

**Science and Mathematics Education Centre**

**Teaching Tools and Techniques for Efficient Teaching and Learning  
of Computer Programming for Beginners Using JAVA**

**Don Nimal Padmasiri Kannangara**

**This thesis is presented for the Degree of  
Doctor of Philosophy  
of  
Curtin University**

**January 2013**

**Declaration**

To the best of my knowledge and belief this thesis contains no material previously published by any other person except where due acknowledgment has been made.

This thesis contains no material which has been accepted for the award of any other degree or diploma in any university.

## **ABSTRACT**

Despite the educational research that has been carried out considering demographic, cognitive and social factors to improve teaching programming in the last decades, finding an effective teaching method is still a debatable issue among Java programming tutors. There are a number of basic concepts to be understood in learning a programming language. The teaching styles to be used to teach different concepts could vary due to the complexity and nature of the concept. This study was aimed at identifying such concepts and the preferred teaching style for teaching such concepts in the Java language. The results of a survey of the students who recently completed introductory level Java programming language revealed such concepts, and also the most preferred teaching style for each concept. This study also investigated the preferred learning styles for learners with artistic abilities and logical abilities. In addition, there have been many research projects based on Cognitive Load Theory (CLT) to investigate better ways of handling germane, intrinsic, and extraneous memory loads on the working memory of learners. The mental modeling technique has been found to be associated with most of the fundamental principles of the Cognitive Load Theory (CLT). This research also included the findings of classroom experiments using activities based on mental modeling, such as analogies, worked examples, and scaffolding, and adhering to the principles of CLT. The context for this research involved teaching Java programming concepts at the introductory level using low cost teaching tools. The study reports on the effects of such activities in teaching Java programming principles.

## **ACKNOWLEDGEMENTS**

I would never have been able to finish this thesis without the guidance of my supervisor, support from the staff and students of Waiariki Institute of Technology, and help from my family and wife.

I would like to express my sincere gratitude to my supervisor, Professor Darrell Fisher, for his admirable guidance, care and patience, and providing me with all the support and advice in conducting this research.

I would like to thank the students who participated in the surveys in both phases of this research. I specially thank the librarian and learning support team at the Waiariki Institute of Technology, especially Mrs Anne-Marie Roux, Mr Graeme Holdaway, Mrs Claire Schnell, Mrs Wendy Monk, and Ms Robin Shirley for helping with proof reading the thesis and the reference pages.

I would also like to thank Mrs Annabel Shuler, the former director of the Department of Computing, Technology, and Communications, who supported me financially for my visit to Curtin University and for purchasing books.

I would also like to thank my mother, brothers and sisters who were always supporting me and encouraging me with their best wishes. Finally, I would like to thank my wife, Deepika Kannangara, and three daughters Jayani, Hasani and Savani who were always there encouraging me and stood by me through the good times and bad.

## TABLE OF CONTENTS

Abstract	iii
Acknowledgements	iv
List of Tables	x
List of Figures	xii
<b>Chapter 1 INTRODUCTION</b>	<b>01</b>
1.1 Background	01
1.2 Java Programming Language	03
1.3 Mental Modeling and Cognitive Load Theory	04
1.4 Overview of Methodology	06
1.5 Significance	07
1.6 Overview of the Thesis	08
<b>Chapter 2 REVIEW OF THE LITERATURE</b>	<b>10</b>
2.1 Introduction	10
2.2 Instructional Models and Learning Theories	14
Behaviourism and Constructivism	14
Kolb Learning Experiential Model	16
Bloom's Taxonomy	17
Felder-Silverman Learning Style Model	18
Past Research on Felder-Silverman Learning Style Model	20
R2D2 Learning Model	22
R2D2 Learning Model for Online Learning	24
Past Research on Learning Styles	25
2.3 Cognitive Load Theory	25
Working Memory Models	27
Advanced Memory Models	28
Atkinson–Shiffrin Memory Model	28
The Baddeley and Hitch Model	29
Kieras Model	31
Working Memory Organisation Approaches/ Hypothesis	32
Total Capacity Approach	32

Task-specific Hypothesis	32
Processing Efficiency Approach	32
Cognitive Load Types	32
Intrinsic Cognitive Load	33
Extraneous Cognitive Load	33
Germane Cognitive Load	34
Interactivity between Memory Loads.	34
Knowledge Representation Theories	35
Cognitive Load Theory as a Pedagogy	37
Past Research based on Cognitive Load Theory	38
2.4 Neurological Aspects of Human Brain	39
Neurological Research on Cerebral Cortex	39
Hemispheric Dominance Theory	40
2.5 Mental Models	44
Past Research on Mental Modeling	45
Constructivism and Mental Models	46
Dual Coding Theory and Relational Organisational Hypothesis	47
Using Visual Tools in Teaching	48
2.6 Instructional Techniques and Tools	49
Scaffolding	49
Use of Anchor Concept Graph in Scaffolding	50
Use of Distributed Scaffolding	50
Collaborative Learning Support using Scaffolding	50
Cognitive Learning Support using Scaffolding	51
Scaffolding using Visual Tools	51
Cognitive Apprenticeship and Metacognition	52
Situated Learning Theory	53
Mind Mapping	54
Use of Mind Mapping in Collaborative Learning	55
2.7 Issues in Teaching and Learning OO Programming	58
Past research on Teaching Issues in OO Programming	58
Cognitive Issues in Teaching	58
Use of Worked Examples in Teaching	59
Use of Cognitive Tools in Teaching	60

Use of Visualising Techniques in Teaching	61
Programming Development Environment Issues	61
Past Research on Teaching Issues in Java Programming	62
Difficult Concepts of the Java Language	62
Pedagogical Issues	62
Conceptual Issues	63
Use of Scaffolding	64
Use of OO-Light Approach	64
Use of Traditional Approach	66
Use of Functional Approach	67
Use of Online Approach	67
Use of Mixed Approach	68
Use of Object First Approach	68
Use of Constructivist Learning Theory	68
Use of Bloom's Taxonomy and Objects-First Approach	69
2.8 Java Development Environments	69
RAPTOR	70
BlueJ	71
Kawa	72
Dr. Java	72
Eclipse	72
Visual J#	73
Borland JBuilder	73
2.9 Summary	74
<b>Chapter 3 METHODOLOGY</b>	77
3.1 Introduction	77
3.2 Research Focus and Significance of the Study	79
3.3 Research Questions	80
Research Questions in Phase One	80
Research Questions in Phase Two	84
3.4 Mind Mapping as a Teaching Tool	84
3.5 Use of Mental Modeling to teach Java Concepts	85
3.6 Use of BlueJ Visual Tool to teach Java Programming	94

3.7	Sampling Technique	96
3.8	Data Collection and Analysis	96
3.9	Assumptions and Limitations	97
3.10	Ethical Considerations	98
3.11	Summary	99
<b>Chapter 4</b>	<b>FINDINGS AND DISCUSSION</b>	<b>100</b>
4.1	Introduction	100
4.2	Findings in Phase One	100
	Difficult Concepts of the Java Language	100
	Correlation between Difficulty Levels and Skills	103
	Types of Learners	105
	Teaching Styles for Different Concepts	106
4.3	Findings in Phase Two	107
	Mini Questionnaire-1 Findings	109
	Findings from the quantitative data	109
	Findings from the qualitative data	111
	Mini Questionnaire-2 Findings	113
	Findings from the quantitative data	113
	Findings from the qualitative data	118
	Mini Questionnaire-3 Findings	120
	Findings from the quantitative data	122
	Findings from the qualitative data	125
	Mini Questionnaire-4 Findings	126
	Findings from the quantitative data	130
	Findings from the qualitative data	133
	Mini Questionnaire-5 Findings	134
	Findings from the quantitative data	136
	Findings from the qualitative data	138
4.4	Performance Improvement of Students	140
4.5	Summary	140



<b>Chapter 5 CONCLUSION</b>	144
5.1 Insights	144
5.2 Significance	147
5.3 Limitations	147
5.4 Implication for Future Research	148
5.5 Recommendations	149
5.6 Final Comment	149
<b>References</b>	150
<i>Appendix A:</i> Code of the STUDENT class	173
<i>Appendix B:</i> Code of the EMPLOYEE class	175
<i>Appendix C:</i> Mini Questionnaire-1	178
<i>Appendix D:</i> Mini Questionnaire-2	181
<i>Appendix E:</i> Mini Questionnaire-3	184
<i>Appendix F:</i> Mini Questionnaire-4	187
<i>Appendix G:</i> Mini Questionnaire-5	191
<i>Appendix H:</i> Participant Information Sheet-1	194
<i>Appendix I:</i> Consent Form	196
<i>Appendix J:</i> Participant Information Sheet-2	197
<i>Appendix K:</i> Ethical Approval	199
<i>Appendix L:</i> Questionnaire – Phase One	200

## LIST OF TABLES

2.1	Comparison of Behavioural and Constructivist Instructional Design Models	15
2.2	Four Quadrants of the Human Brain and Activities	41
4.1`	Difficulty Levels of Java Concepts	102
4.2	Teaching Styles for Different Concepts	106
4.3	Summary of Artistic and Logical Skills of the Participants in Mini Questionnaire-1	109
4.4	Summary of the Most Useful Teaching Tools/Methods Used in Activity-1	109
4.5	Preference of Tools/Methods of Students with Logical and Artistic Abilities in Activity-1	110
4.6	Summary of Tools/Methods Preferred by Students in Activity-1	111
4.7	Summary of Artistic and Logical Skills of the Participants in Mini Quationnaire-2	115
4.8	Summary of the Most Useful Teaching Tools/Methods Used in Activity-2	116
4.9	Preference of Tools/Methods of Students with Logical and Artistic Abilities in Activity-2	117
4.10	Summary of Tools/Methods Preferred by Students in Activity-2	117
4.11	Summary of Artistic and Logical Skills of the Participants in Mini Questionnaire-3	123
4.12	Summary of the Most Useful Tools/Methods Used in Activity-3	123
4.13	Preference of Tools/Methods of Students with Logical and Artistic Abilities in Activity-3	124
4.14	Summary of Tools/Methods Preferred by Students in Activity-3	124
4.15	Summary of Artistic and Logical Skills of the Participants in Mini Questionnaire-4	131
4.16	Summary of the Most Useful Tools/Methods Used in Activity-4	131

4.17	Preference of Tools/Methods of Students with Logical and Artistic Abilities- Activity-4	132
4.18	Summary of Tools/Methods Preferred by Students in Activity-4	133
4.19	Summary of Artistic and Logical Skills of Participants in Mini Questionnaire-5	136
4.20	Summary of the Most Useful Teaching Tools/Methods Used in Activity-5	136
4.21	Preference of Tools/Methods of Students with Logical and Artistic Activity-5	137
4.22	Summary of Tools/Methods Preferred by Students in Activity-5	137
4.23	Students Performance from 2008 to 2012	140

## LIST OF FIGURES

2.1	S-R paradigm	14
2.2	Kolb experiential learning cycle	16
2.3	Bloom's taxonomy	18
2.4	Dimensions of learning and teaching styles	19
2.5	The importance of assessment issues for individual learning styles	21
2.6	The R2D2 Model	23
2.7	R2D2 Model for online learning	24
2.8	Atkinson–Shiffrin memory model	30
2.9	Overview of EPIC architecture	31
2.10	Left hemisphere regions of the brain	39
2.11	Right hemisphere regions of the brain	40
2.12	The four quadrant brain dominance model	42
2.13	Paradigm shift from 1960 to 1990	43
2.14	Brain dominant profile of computer science and engineering students	44
2.15	Sample mind map	54
2.16	Cognitive load relationships in programming	60
3.1	Listed areas and concepts in the Java language	81
3.2	Learning styles explanation	82
3.2	Learning style options for concepts	82
3.4	Questions on artistic and logical abilities	83

3.5	Question on type of learner	83
3.6	Class diagram and mind map of the student class	85
3.7	Star structure	87
3.8	Method returning a value	89
3.9	Java code of main method and getGrade method	90
3.10	Main method using getGrade method	91
3.11	Arrays of objects and primitive types	92
3.12	BlueJ graphical user interface	95
4.1	Scatter diagram of total difficulty level vs. art skills of students	104
4.2	Difficulty levels of learners of different categories	104
4.3	Summary of the types of learners	105
4.4	Activity - 1	108
4.5	Activity - 2	115
4.6	Activity - 3	122
4.7	Activity - 4.1	127
4.8	Pictorial representation of an array of primitive data - 4.1	128
4.9	Activity - 4.2	129
4.10	Pictorial representation of an array of objects - 4.2	130
4.11	Activity - 5	135

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 Background**

An introductory level Java programming course has been taught in the Bachelor of Computing, Communications Technology (BCCT) degree programme at the Waiariki Institute of Technology, Rotorua, New Zealand since 2005. The Dr. Java, an Integrated Development Environment (Allen, Cartwright, & Stoler, 2002), was used as a teaching tool and a recommended text book (Horstmann, 2005) for the course. Teaching materials were based on the content of the textbook and were prepared using PowerPoint slides. The teaching materials contained a broad mix of theory and examples. The order of delivery was based on the chapter sequence of the text book. Examples were used to explain the topics and students tried them out during laboratory sessions. In addition, programming exercises were given to students to apply the knowledge they gained from each lesson. Although there were not sufficient teacher guided laboratory sessions arranged for the students, they were encouraged to do exercises and ask questions. According to Willis's (1995) definition, this instructional method was behavioural.

Unfortunately, more than 50% of the students failed this course and due to the low pass rate there was not a sufficient number of students in the advanced level programming courses. Therefore, it was crucial to improve both teaching and learning to increase pass rates for this course. Being the course coordinator and the key tutor of this course, I began this four year research project to experiment with low cost tools and techniques to investigate the possibility of improving teaching Java for beginners.

To guide this investigation, the following research questions were addressed:

1. What are the difficult Java concepts for the learners and the most suitable teaching styles for teaching such concepts?
2. Is there any relationship between logical and artistic hemispheric dominance factors of students and the difficulty levels of Java concepts?

3. What are the preferred learning styles of learners and the combinations of learning styles?
4. Are the teaching tools based on a combination of Cognitive Load Theory (CLT), the concepts of mental modeling and scaffolding effective in teaching difficult concepts in the Java language?
5. Is there a relationship between students' learning preferences and their logical and artistic hemispheric dominance?

A questionnaire was used to identify the difficult Java concepts for the students who had already done the introductory programming course. This course was also offered at two other polytechnics: UNITEC Institute of Technology and Bay Of Plenty Polytechnic in New Zealand. The three polytechnics used the same syllabus and shared the teaching materials. Therefore, the students in the three polytechnics who completed this course were invited to participate in this survey. Then for the subsequent year, a number of low cost tools and techniques were adopted to teach identified difficult concepts and a series of mini-surveys was conducted to collect both qualitative and quantitative data from students at Waiariki Institute of Technology.

Thus, the participants of the first survey were the students who enrolled in the introductory programming Java course at the three polytechnics in 2008. The participants of the subsequent mini-surveys were the students who enrolled in this course at Waiariki Institute of Technology from 2009 to 2012.

The general belief is that computer programming is difficult due to its logical nature. Although many programming books provide a series of mathematical problem solving exercises, the belief that mathematics knowledge is essential to learning programming is controversial (Hadjerrouit, 1998). According to the comments in the forums available on the internet on this issue, there are some special computer programming applications, such as three dimensional graphics in which mathematics knowledge is essential (Roper, 2012). Some argue that mathematics may help but this is not essential for the learning of computer programming (Mathematics and Computer Programming, 2003). According to Sperry's findings (1981), the intellectual functions of the brain are divided between the left and the right hemispheres of the cortex of the brain (Buzan & Buzan, 2006). As a part of this

study, the Brain Dominance Model (BDM) was studied in detail. According to BDM, the left brain performs logical functions whereas the right brain is for visual functions. In each survey, two questions were asked of students to rate their artistic ability and their logical, analytical and mathematical ability. Each question had five options: Poor, Average, Good, Very Good, and Excellent. Each participant selected only one option for each question. The data collected using the questionnaire in phase one were also analyzed to find out if there was a relationship between students' logical/artistic ability and their preferred learning style.

## **1.2 Java Programming Language**

The Java language was developed as a pure object-oriented (O-O) language in the early 1990s by Sun Microsystems. Many teachers have found Java programming not only hard to learn but also to teach (Kannangara, 2007). The issue of teaching and learning Java programming language has been the focus of most recent researchers who have published hundreds of papers in the last three decades. Therefore, it was vital to find out the inherent difficulty of this language. Prior to the introduction of the object-oriented programming paradigm in 1980s, structured programming was the widely used programming methodology. Norton (1997) argues that "Although object-oriented programming evolved from structured programming, the end-results are revolutionary and rather disorienting" (p. 2). Transition from structured programming to object-oriented languages began three decades ago. Some popular structured programming languages such as C and Basic were further developed with the object-oriented features and introduced to the market as the object oriented version of the languages with names such as C++ and Visual Basic. These hybrid languages comprise of both object-oriented and structured features. Many teachers taught these hybrid languages using structured programming features initially and then the object-oriented concepts were introduced towards the end of the course (Kannangara, 2007). Although this was possible in teaching hybrid computer languages such as C++ and Visual Basic, it was not possible with the Java language due to its pure object-oriented nature. Therefore, a teaching paradigm for Java programming language requires drastic changes to the structured way of teaching (Kannangara, 2007).



The research project began with a search of the literature related to the teaching pedagogies of Java programming language. The literature review revealed that this topic has been investigated by many researchers and an enormous number of research projects have been carried out exploring this issue from different perspectives over the last two decades. Such perspectives include: conceptual, cognitive, paradigm, interactive development environments, pedagogical, instructional design models, teaching tools, and sequence of delivery. Currently, the focus of newly published research is found to be based on the cognitive aspects of teaching.

### **1.3 Mental Modeling and Cognitive Load Theory**

Mental modeling has been historically used for scriptural interpretation in religious places using statues and pictures (Johnson-Laird, Girotto, & Legrenzi, 1998). This concept of mental modeling was postulated by Craik in 1943 as psychological representations of real, hypothetical, or imaginary situations (Johnson-Laird et al., 1998). Some researchers have realized and used these imaginary situations in teaching and understanding concepts. Van Haaster and Hagan (2004) have suggested the possibility of using images in creating mental models to help understand programming concepts. The mental modeling approach has been found to be useful in teaching programming concepts by a number of researchers (Garner, 2009; Ma, Ferguson, Roper, & Wood, 2007; Werhane et al., 2011). Therefore, mental modeling was chosen to be used in the teaching of Java programming concepts in this research study.

Cognitive Load Theory was proposed by Miller (1956) and identified the limitations of the working memory. This theory has been further developed by a number of researchers with working memory models, different cognitive loads, and Schema Model Theory (Arbib, 1992). The new developments describe the process of learning more effectively and ways of utilizing the limited working memory of the learners to improve the learning process (Sweller, 1999). Garner (2002) developed a teaching tool called the Code Restructuring Tool (CORT) and experimented using worked examples to improve the teaching of Visual Basic programming. The worked examples were found to be useful in reducing the intrinsic cognitive load on the working memory of the learner resulting in better performance in learning.

Garner (2009) later suggested the use of mental modeling prior to the use of CORT for better performance. Therefore, the possibility of using cognitive aspects related to Cognitive Load Theory (CLT) was chosen for experimentation with the aim of enhancing the teaching of the Java language at the Waiariki Institute of Technology.

The concept of mental modeling was experimented on with worked examples as teaching tools to teach Java concepts in this research. In addition to these, cognitive aspects of the teaching materials were also considered as a technique to help students better understand the concepts. The teaching materials of the Java programming course were modified according to the principles of the CLT, minimizing extraneous and intrinsic cognitive loads. Scaffolding was used to balance the varied germane cognitive load levels of students. Mental modeling is also found to be related to the Dual Code Theory (DCT), which describes the way human brain processes visual and verbal information (Paivio, 2006). The worked examples, mental modeling, and scaffolding were the three main techniques applied in this study with the aim of improving teaching Java programming for beginners at the second phase of the research.

The object-first model is one of the pedagogical approaches used to teach object-oriented programming (Lister et al., 2006). A slightly modified version of the object-first approach, concentrating more on concepts, was used to teach Java programming concepts in this research. In this approach, the basic concepts such as: creating a class structure, creating objects using a class, manipulating objects, using control structures, using methods, parameter passing, and using arrays were covered first. These concepts were taught using worked examples, mental modeling, and scaffolding. The Integrated Development Environment used was chosen for the teaching of Dr. Java due to its simplicity and its suitability for beginners. The teaching methodology used was constructivism with collaborative learning. The teacher-centred delivery method was chosen due to the small classes of this course. The interactive teaching with hands on sessions was used as the classes were held in computer laboratories. This enabled scaffolding to be used especially for students with difficulties in understanding programming concepts and practical issues. Available teaching tools as well as low cost teaching tools such as images were

adopted in teaching. The consistency of images used for interpretation of different aspects of the Java language was maintained throughout this study.

#### **1.4 Overview of Methodology**

This research was carried out in two phases. The Analyze, Design, Develop, Implement, and Evaluate (ADDIE) model was used as a guideline for instructional design in both the phases (Culatta, 2011). In the first phase, the data were collected from the students who had completed this course prior to the introduction of new teaching tools and methodologies. Using the data collected from students, the difficult concepts in the Java language for the students were identified. In the second phase, mental modeling, worked examples and scaffolding were adopted to teach the difficult concepts identified in the first phase. Mind maps were used as a mental model to study the possibility of enhancing teaching the concept of the class structure of Java programming language. There are some visual programming environments which can be used to create programs without much effort; the novice programming students find these easier to understand. But the critique is that some students tend to be familiar with graphical environment and not the concepts (Pears et al. 2007). These graphical environments do generate some programming code as well. In this research, BlueJ, a graphical program development environment was introduced to students who made a judgment about the tool after learning all the concepts using mental modeling tools and using code based Dr. Java programming environment.

The data collected from the questionnaires were analyzed using statistical software such as NVivo 9 and statistx-8. The use of a combination of cognitive aspects, mental modeling and scaffolding to teach Java at introductory level is unique to this research project. The tools and techniques used are all low cost and affordable for many institutions. Therefore, the findings of this research could benefit many teachers who teach at the introductory level of Java programming. As a result, students would also benefit from the findings. The findings could also open new possibilities for research for those who are involved in research on teaching programming issues.

The majority of participants responding to the surveys were students enrolled in the introductory programming course using the Java language at the Waiariki Institute of Technology. The limited number of students enrolled had an impact on the choice of samples. The highest intake was around 40 students in 2008. In the subsequent years, the number of students enrolled decreased to between 10 to 20 students in a class. Because of this, convenience samples had to be used in the research and consequently the data samples were smaller than expected. This may have affected the accuracy of the findings.

### **1.5 Significance**

The focus on this research has been on the use of low cost teaching tools based on cognition and mental modeling to explore the possibility of enhancing the teaching of Java programming language. Researchers who are currently working on improving the teaching of object-oriented programming will benefit from the findings of this study as it may lead them to further research in this area. Among the wide range of applications of mental modeling, Ma et al. (2007) have suggested the possibility of using viable mental models in teaching Java programming. This research used mental models according to the guidelines of Cognitive Load Theory (CLT) with the intention of maintaining germane and extraneous memory loads at a minimal level. In addition, scaffolding was experimented with using partially-completed Java programming examples for the same purpose as suggested by Garner (2007). Therefore, outcomes of this research will be useful for researchers who are currently involved in research related to Cognitive Load Theory (CLT), mental modeling and scaffolding. This research has also explored the relevance of students' hemispheric dominance to the suitability of a number of teaching tools that were used in this study. Therefore, the findings will benefit those who are interested in Hemispheric Dominance Theory (HDT) and its applications.

The outcomes of this study will also be significant for Java programming teachers as it discloses the difficult concepts and areas for students learning Java programming language and the possibility of applying mental modeling concepts using a unique set of symbols. In addition, teachers will be able to use the successful tools along with the teaching styles discovered in this research project. Furthermore, the Java

teachers also will be exposed to new teaching tools and methods which could increase the effectiveness of teaching.

Overall, the findings of this study will benefit the stake holders, especially researchers, teachers and learners of Java programming language. Low cost tools were used in this project; therefore the successful tools should be affordable for many teachers and organizations who may wish to use them.

## **1.6 Overview of the Thesis**

The first chapter, the introduction, contains the background and the research questions addressed in this research. A brief description of the methodology and the research background and theories based on each tool also are found in Chapter One as are the significance of the findings of the research and an overview of the thesis.

Chapter Two includes a comprehensive literature review which includes: the popular instructional design models and learning theories; in depth coverage of the Cognitive Load Theory (CLT); memory models and knowledge representation theories; and past research on cognition. The neurological aspects of the human brain and Hemispheric Dominance Theory (HDT) are also discussed in detail in Chapter Two. With regard to mental modeling, past research, its relevance to constructivism, Dual Coding Theory (DCT), and Relational Organisational Hypothesis (ROH) are also discussed. A section in Chapter Two includes instructional techniques and tools, such as scaffolding, cognitive apprenticeship, and mind mapping. Past research on teaching object-oriented programming and Java programming is the largest section of the chapter. The programming issues include cognitive, pedagogical, conceptual, the use of visualization, scaffolding, cognitive tools, and a number of approaches to teaching the Java language. The features of seven Integrated Development Environments (IDE) and their pros and cons are also discussed in Chapter Two.

A detailed research methodology is presented in Chapter Three. This chapter covers the background details which led to the choice of the research questions and the methodologies that were used in this research. Such details include conceptual issues in the Java language, mental modeling, balancing cognitive loads, Hemispheric Dominance Theory (HDT) aspects and the use of scaffolding. The methodology of

testing of mind mapping and graphical user interface programming environment (BlueJ) are also included in Chapter Three.

Chapter Four presents the results obtained from the use of the quantitative and qualitative findings of the questionnaire used in phase one of the research and the five questionnaires used in phase two.

Chapter Five identifies insights gained from the findings and includes a discussion. In addition, this chapter covers the significance, limitations, implications, and recommendations.

## **CHAPTER 2**

### **REVIEW OF LITERATURE**

#### **2.1 Introduction**

Teaching and learning programming at an introductory level is known to be challenging for both teachers and students (Van de Ven & Govers, 2007), and Java has a well-deserved reputation as a significantly more complex programming language than comparable procedural languages. This is because Java is a pure object-oriented computer programming language in which archetypical procedural characteristics are almost totally non-existent. Therefore teachers utilising Java need to take special care to introduce concepts in ways that limit complexity and furthermore, such pedagogic methodological considerations should be based upon a theoretical understanding of learning models and approaches. Not surprisingly, there is an enormous amount of published research literature concerning the use of different teaching tools and techniques to enhance and simplify the teaching of programming (Kannangara, 2007). While the matter remains largely unresolved and debatable, nonetheless some clear themes emerge.

Theoreticians, researchers, educators and psychologists have introduced a range of learning models and successfully used them in different areas of teaching. This review includes a general historical framework of such established models, as well as a survey of the latest knowledge representation theories grounded in Cognitive Load Theory (CLT). Established models are based on a few basic theories, including behaviourism, constructivism and cognitivism, or combinations of these, although there are fundamental differences between the first two. Among the popular learning models used by educators are Bloom's Taxonomy, the Felder-Silverman Learning Style Model, and the Recursive, Reflective, Design and Development (R2D2) model. Numerous research publications based on these models have been completed and enhance teaching and learning in different ways.

In the last two decades there have been further research projects which take cognitive aspects into consideration and apply these to enhancing teaching quality. In

particular, Cognitive Load Theory, applied through a range of different cognitive models, is a notable feature of recent research projects (Clark, Nguyen, & Sweller, 2006). One fundamental aspect of this theory, the limitation of Working Memory (WM), is the focus of current research in teaching and learning. There are different definitions of WM, as well as a number of approaches and hypotheses about memory organisation in the use of WM. The literature on different cognitive loads, whether intrinsic, extraneous or germane, and the element of interactivity between them, has been found to be important in preparation of better quality teaching materials and tasks, and many research papers discuss the better utilisation of WM by manipulating the intrinsic, extraneous, and germane factors. The importance attributed to WM as a concept within cognitive aspects of teaching and learning is due to recent neurological research, in which different functional regions of the brain including WM, have been identified (Smith, 2000; Smith & Jonides, 1999). These findings seem to agree with Hemispheric Dominance Theory (HDT), which was introduced by Sperry (Chwif & Barretto, 2003). The review also draws attention to mental modeling as a powerful technique in teaching and learning, and notes that past research on mental modeling has revealed that visualisation can be successful in teaching Java programming concepts. The concept of mental models is based upon social constructivism, a theory advanced by Piaget, Vygotsky and Bruner (Berk, 2003). Related to the concept of mental modeling is the Dual Code Theory (DCT), which also involves visualisation. Despite the long history of DCT, recent research shows that it is still applicable today: it is, for example, the basis for Baddeley's Working Memory Model (Paivio, 2006). Researchers have experimented with visual tools and analysed their relevance to the cerebral and limbic quadrants of the brain. Further details on mental models, visualisation tools and DCT are included in this literature review.

Applications of two tools and techniques, namely scaffolding and mind mapping, appear prominently in the research literature in a range of contexts, including teaching Java programming language. Scaffolding is an essential instructional technique, which can be used with both constructivist and cognitive methods of teaching programming. Cognitive apprenticeships are one kind of scaffolding, related to the social constructivist paradigm (Cognitive apprenticeship, n.d.). Here, situated learning theory is applied to scaffolding within a cultural context. Mind



mapping is another instructional tool that can be effectively used in collaborative learning and is documented in this literature survey.

It is increasingly apparent that the teaching of object-oriented (O-O) programming requires a paradigm shift from structured ways of thinking to an O-O way of thinking. Some issues in imparting programming knowledge to novice learners are conceptual, while the rest are inherent to the programming language used for teaching. Although a number of research projects have been carried out to find ways of improving the teaching and learning of programming, the best method for teachers of students at beginner level is still a highly debatable question (Kannangara, 2007). Most published research has been on the use of different teaching pedagogies and tools, and on the sequence of delivery depending on the nature of the programming language used. As this research is focussed on issues related to OO concepts and the Java programming language, this review includes past research on both areas.

Finally, the choice of a Java development environment in teaching can have a significant impact on learning. The main categories of Java development environments are: professional tools used in the industry, educational tools and tools with graphical user interfaces (GUI). There are pros and cons for each environment category (Brusilovsky, Calabrese, Hvorecky, Kouchnirenko & Miller, 1997). This chapter includes details of popular Java development environments including RAPTOR, BlueJ, Kawa, Dr. Java, Eclipse, Visual J#, and JBuilder.

Section two of this chapter begins with a comprehensive analysis of the features of behaviourist and constructivist theories followed by descriptions and applications of popular instructional design methods such as the Kolb Learning Experiential Model, Bloom's Taxonomy, the Felder-Silverman Learning Style Model and the R2D2 Learning Model. In addition, past research on learning styles is discussed. The third section of this chapter focuses on the fundamentals of the Cognitive Load Theory and Working Memory Models such as Baddeley and Hitch, Atkinson-Shiffrin, and Kieras. This section includes a brief description of different WM definitions such as the Total Capacity Approach, the Task-specific Hypothesis, and the Processing Efficiency Approach. The cognitive load types: intrinsic: extraneous: germane: their interactivity, and the how they affect the limited WM in the learning process are also discussed in section three. In addition, knowledge representation theories on the

Long Term Memory (LTM), Short Term Memory (STM) and schema concept are also discussed. This section also includes the use of Cognitive Load Theory as pedagogy, and past research related to teaching computer programming languages.

Section four of this chapter discusses the recent neurological research findings on the functionalities of the different parts of the human brain. This section also includes a detailed description of the Hemispheric Dominance Theory which describes the functionalities of the four quadrants of the brain. This section also includes details of some research on brain dominance theory and the paradigm shift related to the four quadrants of the Brain Dominance Model.

Section five of this literature review includes the mental modeling concept and its recent applications in teaching introductory programming. This section also contains some research findings on visualisation and conceptual representation using mental modeling. The latest philosophical research outputs in mental modeling with regard to constructivism and social constructivism theory are also included in this section. The use of imagery eventually became pictures. The Dual Coding Theory which describes the way of storing images in the brain is also discussed in section five. Some research findings on the use visual tools in teaching are also included in this section.

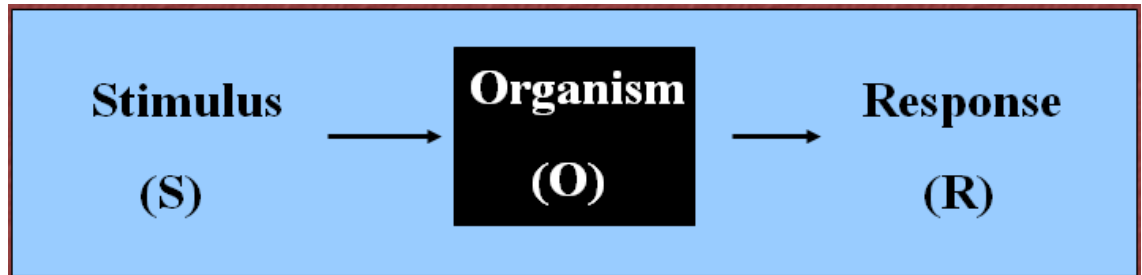
Section six discusses the instructional techniques and tools that are used in teaching. The main focus is on scaffolding and includes the research applications of scaffolding on different aspects of teaching. In addition, more socialised scaffolding techniques, such as apprenticeship and situated techniques, are discussed. Mind mapping usage as a teaching tool and research applications in different areas of teaching are included in the discussion. A number of research applications using mind mapping in collaborative learning Java programming and other programming languages are also discussed in this section.

Section seven of this chapter is the longest section where the research projects related to teaching issues, different findings and other object-oriented languages at beginners' level are discussed. The final section of the chapter provides a summary of the review of the literature and further research opportunities are identified.

## 2.2 Instructional Models and Learning Theories

### 2.2.1 Behaviourism and Constructivism

The traditional instructional models are based on behavioural theories. The S-R paradigm describes the behavioural learning theory. The learner is considered as a black box. Stimuli activate the senses and the overt behaviour of the learner (organism) is the response to the stimuli (see Figure 2.1).



*Figure 2.1. S-R paradigm.*

Willis (1995) has clearly differentiated the characteristics (features) of the behavioural instructional design model and the constructivist instructional design model as in Table 2.1

Table 2.1

*Comparison of Behavioural and Constructivist Instructional Design Models*

Feature	Behavioural design	Constructivist design
Process	Sequential and linear	Collaborative, nonlinear and recursive
Planning	Top down and systematic	Organic, developmental, and reflective
Objectives	Guided development	Emerge from development and design work
Experts	Experts involved in the process	Think that experts do not exist
Instructions	Careful sequencing of teaching with sub-skills	Instruction highlights learning in meaningful contexts
Goal	Delivery of preselected knowledge.	Personal considerate within meaningful contexts
Evaluation	Summative is important	Formative is vital
Data	Objective data are valuable	Subjective data are valued

There are three types of behavioural learning theories, namely, contiguity, operant conditioning, and classical conditioning (Huitt & Hummel, 1999). Willis (1995), describes a behaviourist as a person who assumes language as a theory-neutral medium which could be used to impart meaning of the external world to learners without being influenced or changed. As described by Willis (1995), for a constructivist, language is contextual and it is believed that the meaning of language develops as it is used. This means that behaviourists and constructivists have different perceptions on the role of language and the nature of truth. According to Willis (1995), behaviourists and constructivists have different views and they use different approaches in designing teaching lessons. Although behaviourists carry out scientific research, they believe that knowledge is objective and universal. The constructivists' belief is quite the opposite to this and assumes that knowledge is subjective (Willis, 1995). The behavioural instructional design model is sequential whereas constructivists use a recursive and cyclic process. The behaviourists plan instructions systematically in a top down manner. Constructivists' planning involves collaboration and reflection and is developmental and organic. The objective of behaviourists is guided development whereas the objective emerges from the constructivist's design and development work. In behaviourism, there are experts

who have special knowledge, but there are no experts in constructivism. Careful teaching of sub skills in a sequential manner is important in behaviourism. In constructivism, learning meaningful contexts is emphasised by the instructions. The aim of behavioural instructional design is the delivery of preselected knowledge. Personal understanding within meaningful contexts is expected from constructivist instructional design model. Summative evaluation is critical in behaviourism and formative evaluation is critical in constructivism. Objective data are valuable in behaviourism, but subjective data are most valuable in constructivism (Willis, 1995).

### 2.2.2 Kolb Experiential Learning Model

Kolb, who devised this model, believed that it is “the process whereby knowledge is created through the transformation of experience. Knowledge results from the combination of grasping and transforming experience” (Kolb, 1984, p. 41). It is a holistic model based on experiential learning. The term “experiential” differentiates this model from other cognitive and behavioural theories (Kolb, Boyatzis, & Mainemelis, 1999). The Kolb Learning Cycle and Fekier’s Learning Styles are similar models that describe how students learn (Howard, Carver, & Lane, 1996). This model upholds the reflective constructivist view using experiential learning (Greenaway, 2011).

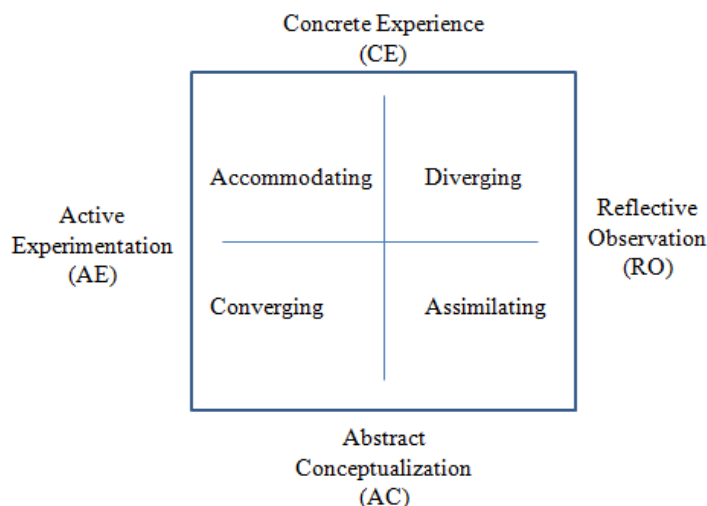


Figure 2.2. Kolb experiential learning cycle.

The Kolb experiential learning cycle has four modes (see Figure 2.2). Two of them, Concrete Experience (CE) and Abstract Conceptualization (AC), are dialectically related for acquiring experience whilst the other two, Reflective Observation (RO) and Active Experimentation (AE), are dialectically related for transforming experience (Kolb et al., 1999). A learner may grasp new information through tangible and concrete experience or using symbolic representation or a virtual simulated environment. According to Kolb et al. (1999), concrete experience is the basis for observations and reflections. Reflections are assimilated into abstract concepts. Such concepts lead the learner to new implications which can be experimented with, and as a result new experiences are gained. As Kolb and Kolb (2005) describes, the theory based on the Kolb experiential learning model is built on propositions: learning is not best envisaged in terms of outcomes but as a process of continuing reconstruction of experience; the process of learning should draw out the students' beliefs and ideas to facilitate learning; conflict, differences, and disagreements drive the learning process; learning is a holistic process of adaptation to the world; learning requires synergetic transactions between the learner and the environment; and in the process of learning, knowledge is created. Kolb also identified and included, in his experiential learning cycle, four different learning styles namely diverging, assimilating, converging, and accommodating (Kolb & Kolb, 2005). Teaching recursion for novice programming learners was the subject of a group of researchers using Kolb's Experiential Learning Model. In this research, the researchers believed that each learner has a unique way of perceiving and processing information which is the most comfortable way of learning (Wu, Dale, & Bethel, 1998).

### **2.2.3 Bloom's Taxonomy**

The meaning of the word taxonomy is classification. Bloom, an educational psychologist, developed this method of classifying intellectual levels of learners in 1956. The model was updated to make it relevant to 21<sup>st</sup> century education by Anderson in 1990 (Overbaugh & Schultz, n.d.)



*Figure 2.3. Bloom's taxonomy.*

This model classifies the forms and the learning levels of a student (Atherton, 2010). The hierarchical levels represent the learner's depth of knowledge in a given subject or cognitive domain. The first layer represents the learner's ability to memorize or recall the facts (Howard et al., 1996). The learner at the second layer level should be able to explain or describe the facts. A learner who achieves the third layer should be able to apply and use the facts. At the analysing level, the learner should be able to compare and contrast the facts. The ability to defend, judge, and evaluate is achieved by the learner at the evaluating level of the Bloom's Taxonomy. The top most level is reached by those learners who can create a new product or point of view (Overbaugh & Schultz, n.d.). The learners may reach a deeper understanding of the subject matter at the highest level of the Bloom's Taxonomy (Howard et al., 1996). Bloom believed that learning is connected to cognitive, affective or psychomotor domain. The cognitive domain involves mental skills, processing information, and knowledge. The affective domain relates to attitudes and feelings and the psychomotor domain to manipulative physical skills (Churches, 2009).

#### **2.2.4 Felder-Silverman Learning Style Model**

Felder and Silverman (1988) carried out some research based on the belief that the learning depends not only students' native ability and prior preparation but also compatibility of the student's style of learning with the instructor's style of teaching. According to Felder and Silverman (1988) learning takes place in two steps. The first step is reception of external information through the senses. The second is the

processing step which involves “simple memorization or inductive or deductive reasoning, reflection or action, and introspection or interaction with others” (Felder & Silverman (1988, p. 674). As a result of the two steps, a student is either learning or not learning. Felder and Silverman (1988) introduced the teaching and learning style model which is comprised of a learning-style model and a teaching-style model. The learning-style model classifies students according to their ways of receiving and processing information. The teaching-style model classifies teaching methods for addressing proposed learning-styles (Felder & Silverman, 1988). Most of the learning styles can be matched to a suitable training style (see Figure 2.4)

<i>Preferred Learning Style</i>		<i>Corresponding Teaching Style</i>	
sensory	} perception	concrete	} content
intuitive		abstract	
visual	} input	visual	} presentation
auditory		verbal	
inductive	} organization	inductive	} organization
deductive		deductive	
active	} processing	active	} student participation
reflective		passive	
sequential	} understanding	sequential	} perspective
global		global	

*Figure 2.4.* Dimensions of learning and teaching styles. Adapted from (Felder & Silverman, 1988, p. 675)

Some learners perceive the world by sensing or intuition. In sensing, a learner receives data through senses. Intuition is a way of perceiving the world through imagination. The second categories of learners, visual/auditory, are divided into three categories namely visual, auditory, and kinaesthetic. This category was renamed as visual/verbal by Felder in 2002. The learners who could remember things better when they learn from visuals and textual representations regardless of whether they are written or spoken fall into this category (Graf, Viola, Kinshuk, & Leo, 2007). Babies learn by observing the world. This learning style is known as induction. The deductive way of learning requires organised materials to be presented to the learner. Felder and Silverman (1988) categorised the third type of



learner as inductive/deductive. Both active and reflective learning involve complex mental processes. Active learners learn by actively engaging in experiments and reflective learners examine and manipulate information when learning. The active/reflective learning category is the fourth in this model (Felder & Silverman, 1988). The fifth category of the model is sequential/global category. Sequential learners learn by using presentation of material in a logically ordered manner. They tend to follow small incremental steps when finding solutions. Global learners expect to be provided with the big picture or the goal of the lesson before being introduced to the steps. Global learners are divergent learners and sequential learners are convergent learners (Felder & Silverman, 1988). Global learners tend to take in learning material almost arbitrarily. When a learner learns enough material, he will be able to get the global picture of the learning outcome (Graf et al., 2007). Global learners are able to find connections between different areas and solve complex problems. Sequential learners expect broad knowledge whereas global learners are interested in overviews (Graf et al., 2007).

#### *2.2.4.1 Past Research on the Felder-Silverman learning style model*

Felder and Silverman's research was carried out with students having an engineering background. The researchers grouped two types of learners by taking their similar characteristics in engineering education into consideration. This model became very popular over a 10-year period and it is now known as the Felder-Silverman Learning Style Model (FSLSM). This model was used by a group of researchers at Athabasca University, Canada, to investigate the possibility of improving their web-based courses. Peer assessment techniques were adopted using Felder-Soloman's Index of Learning Styles (ILS) questionnaire to identify each other's preferred learning styles (Felder & Soloman, 1997). ILS is based on the Felder-Silverman Learning Style Model (Kovacic, Green, & Eves, 2004). According to peer assessments, the students were classified into different groups for adaptive web-based teaching. Two of the four dimensions of teaching styles, active/reflective and sensing/intuitive, were considered with particular interest in this research. The four assessment issues considered in this research were Creativity, Completeness, Execution, and Security (Wen, Graf, Lan, Anderson, & Dickson, 2007).

Issues\LS	Active Sensing	Active Intuitive	Ref. Sensing	Ref. Intuitive
Creativity	Low	High	Low	High
Completeness	High	Low	High	Low
Execution	High	High	Low	Low
Security	High	High	Low	Low

*Figure 2.5. The importance of assessment issues for individual learning styles.*  
Adapted from Wen et al., 2007, p. 12

Active learners usually try out things and work actively with the learning material by applying the content of the material, and by experimenting. Active learners also prefer communicating with others by working in groups participating in discussions. Reflective learners prefer to think and reflect on the learning material. They prefer to work individually or in a small group with a good friend (Graf et al., 2007). Sensing learners are considered to be patient and work slowly and carefully. They usually learn facts and actual learning material and use standard approaches to solve problems. Such learners are considered to be more sensible and realistic. They are more practical and relate the learnt material to the real world (Graf et al., 2007). Intuitive learners enjoy challenges and hence benefit from exercises. Assessment issues in this research include creativity, completeness, execution and security. The research group at Athabasca University concluded that more effective and accurate assessment could be achieved by taking individual learning styles of students into consideration (Wen et al., 2007).

The use of learning styles is becoming popular in technology-enhanced teaching and learning (Graf et al., 2007). According to Graf et al. (2007), this could be achieved by accommodating and integrating learning styles into all aspects of educational technology. Recent investigations on learning styles have revealed the importance of incorporating suitable learning styles to make learning easier. A research study was carried out to identify characteristics of the four dimensions of the FSLSM model at Massey University in New Zealand and Vienna University of Technology in Austria. The four dimensions (groups) of learning styles considered in this research were active/reflective, sensing/intuitive, visual/verbal, and sequential/global. This research used the Index of Learning Styles (ILS) Questionnaire (Felder & Soloman, 1997) to

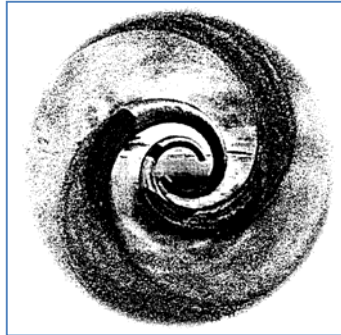
collect data from about 290 undergraduate and postgraduate students at Massey University and Vienna University of Technology. The items in the ILS questionnaire were grouped into four dimensions. The data were analysed in depth using the correlation between learning styles and the dimensional group. The findings tallied with the FSLSM model. This research found a strong correlation between visual learning style and the visual/verbal dimension (Graf et al., 2007).

Litzinger, Lee, Wise and Felder (2007) used the FSLSM model with modified items from the ILS questionnaire, adding five scale options for each question in a research project at Pennsylvania State University. The addition of five scale options improved the reliability of the data collected. The research was focused on investigating the validity of the FSLSM model using the ILS questions having a dichotomous response format aiming to enhance the reliability of the data. The introduction of multi scale options resulted in a reduction of the standard deviations of the scores for all scales and improved consistency and reliability of the four dimensions (Litzinger et al., 2007)

### **2.2.5 R2D2 Learning Model**

Most traditional instructional design models are based on “social science theories from the behavioural family, broadly defined to include information processing and cognitive science theories that break down content to be taught into smaller units which are then taught with direct instruction strategies” (Willis, 1995, p. 5). Willis (1995) introduced an alternative model of instructional design (ID) to the traditional models. The R2D2 model was originally proposed as the Recursive, Reflective Instructional Design Model by Willis in 1995. This model was based on three principles, namely, recursion, reflection, and participation (Willis, 1995). Willis and Wright (2000) say that the R2D2 is one way of implementing the basic principles of constructivist instructional design. Willis (1995) argues that this model differs from other behavioural teaching models as it begins with a team of stakeholders with a general rather than specified aim and also because of its cyclic nature. The team of stakeholders is usually composed of instructors, students, subject matter experts, and instructional designers. The R2D2 design is non-linear and it can be followed in any order and revisited at any time (Chen & Toh, 2005).

According to the R2D2 learning model, the focus is fuzzy at the beginning of learning and as it progresses, it becomes sharper (Willis & Wright, 2000). Three focal points of the model (see Figure 2.6) are Define, Design, and Development. The learning takes place in a cyclic manner and the pattern is unpredictable. The participants will work on three aspects of the design and the focal points are what is important about the work (Willis & Wright, 2000).

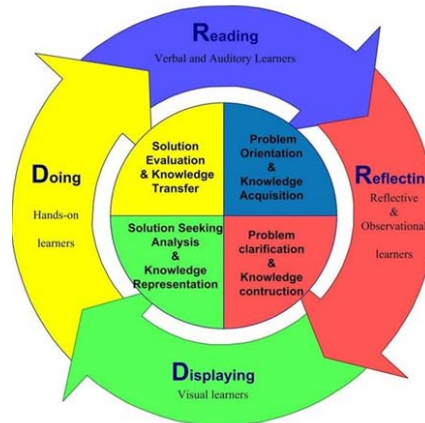


*Figure 2.6. The R2D2 Model. Adapted from (Willis & Wright, 2000, p. 6).*

The three activities at the focus of this model have been defined as creating and supporting a participatory team, progressive problem solution, and developing phronesis or contextual understanding. All the activities are equally important for the entire process (Willis & Wright, 2000). The most difficult task for a constructivist designer is to create a group with supportive, encouraged, and facilitated participation members. One way of creating a good team is to select a small participatory group of members representing each different stakeholder group. An alternative way of forming a good group is to organize a core team with two or three members and then various people will be involved at different points of the process (Willis & Wright, 2000). The R2D2 learning model is one of the constructivist models used in mainly academic and research contexts. Although there is no conclusively proven research evidence that these models are the best instructional designs, selecting a model is rational not empirical (Willis & Wright, 2000). Reigeluth (1996) has indicated changing nine aspects of teaching as a result of a paradigm shift from the industrial age paradigm to the information age paradigm. This indicates the current trend of moving from behaviourism to constructivist instruction design.

### 2.2.6 R2D2 Learning Model for Online Learning

The R2D2 learning model was extended for online learning environments by Bonk and Zhang (2008). This model is cyclic with four phases namely Reading, Reflecting, Displaying, and Doing. The Reading phase is for auditory and verbal learners. The Reflecting phase is for reflective and observational learners. The Displaying phase is for visual learners. The Doing phase is for kinaesthetic learners.



*Figure 2.7. R2D2 Model for online learning. Adapted from (McKinney, 2009, p. 1)*

When a course is delivered online, it is easy to prepare course materials for those who learn by reading. If the online delivery is to be successful, it should also facilitate learning by providing elements of the learning styles favoured by those who learn by hearing, doing, or reflecting critically (Amckinn, 2009). Cartner and Hallas (2009) used this model to teach English online in the English for Academic Study (EAS) programme at Auckland University of Technology (AUT). The blended paper on Listening and Note Taking was delivered using two hours of learning with a computer, two hours face to face learning and six hours of independent study per week. Students were provided with language learning software, Microsoft Word, Microsoft PowerPoint, and four activities facilitated through a website. The students had to learn about the use of: writing skills using Microsoft Word; emails for communication; forums or blogs; wikis for collaborative work; internet for topic research; audio files; podcasts; and online videos. Four activities were completed using the R2D2 cycle (Cartner & Hallas, 2009). A survey was used to gather data about students' assessment of their progress of learning using this model. The finding revealed that the strength of the model lies in the two phases: reflection and

doing, which do not usually happen in blended environments. The student feedback and the facilitator's feedback were useful in assisting task completion and independent and dependent learning (Cartner & Hallas, 2009).

### **2.2.7 Past Research on Learning Styles**

Students' learning styles were studied empirically using Felder-Soloman's Index of Learning Styles (ILS) questionnaire at the Open Polytechnic of New Zealand. The study focused on the learning styles used for teaching computer concepts (Kovacic et al., 2004). Kovacic et al. (2004) argue that both processing (Actively/Reflectively) and perception (Sensing/Intuitive) are two dimensions which are common to FSLSM and Kolb learning models. But new dimensions, Visually/Verbally and understanding with two poles, Sequentially/Globally, were added to FSLSM. The result of the study revealed a significant relationship between learning styles and socio demographic characteristics (Kovacic et al., 2004).

Another study examined the suitability of learning styles for introductory level programming and also the relationship between student learning style and academic performance. In this study, the reflective learners did better than active learners and verbal learners performed better than visual learners in examinations (Thomas, Ratcliffe, Woodbury & Jarman, 2002).

A group of researchers carried out a study to discover the most suitable learning style for teaching mathematics and computer programming. Both quantitative and qualitative data including student comments were collected in this study. The outcome of the survey was that mathematics students prefer sequential, inductive, and deductive learning styles and the programming students prefer sequential, visual, and active learning styles. It was also revealed that learning mathematics requires a strong verbal component while computer programming needs more visual components (Zander et al., 2009).

## **2.3 Cognitive Load Theory**

The Cognitive Load Theory (CLT) explains information processing for cognition using the concept of Working Memory (WM) and Long Term Memory (LTM). According to this theory, human LTM is a huge store of skills and knowledge in the

brain. Miller (1956), who laid the foundations of CLT referred to WM as short-term memory (STM). CLT is based on finding more about the limits of WM in terms of the amount of information it can hold (Van Gerven & Pascal, 2003). According to Miller (1956), the STM of the human brain imposes severe limitations on the amount of information that we are able to receive, process, and remember. Miller (1956) found out that for most people the number of items of information that can be maintained in an active state simultaneously in the STM is seven.

Miller's findings have been developed further over the last 50 years into a comprehensive set of instructional principles called Cognitive Load Theory (Clark et al., 2006). The Cognitive Load Theory was built upon the assumption that people can deal with two or three elements at a time and that the degree of interactivity of such elements could affect the capacity of this WM (Garner, 2002). The CLT describes the way learning takes place within the brain and cognitive loads are imposed by complex cognitive tasks (Paas, Van Gog, & Sweller, 2010). The CLT is proven to result in efficient instructional learning environments as a consequence of leveraging the human cognitive learning process (Clark et al., 2006). Some authors define CLT as a universal set of learning principles which are related to fundamental tools of training such as text, visual and audio (Clark et al., 2006). It has been proven that learning can be improved by minimizing the wasted mental resources described in cognitive load theory. According to the CLT, managing the learner's WM includes limiting the complexity of the work, reducing the degree of mental effort involved and minimising irrelevant information provided by the teacher (Kalyuga, 2006).

Building on Miller's (1956) findings on limitations on STM, Sweller (1988) developed the CLT that consists of schemas, or a combination of elements which describes an individual's knowledge base. It has been the focus of instruction and learning assuming that the limited WM of an individual contributes to limitations on information processing (Sweller, 1999). Chandler and Sweller (1991) describe Cognitive Load Theory (CLT) in this way, "Cognitive Load Theory is concerned with the manner in which cognitive resources are focused and used during learning and problem solving" (p. 2). The WM has limited capacity and duration. New knowledge is generated on the working memory. In the process of creating new

knowledge, the past knowledge related to the new learning stored on the LTM is transferred to the WM. Using this past knowledge and the new visual and audio information on sensory stores, the new knowledge is created on the WM. Finally newly created knowledge is transferred to the LTM (Cooper, 1998). The CLT can also be described as an instructional design theory which could be used by teachers to reduce the load caused by poor design of teaching and learning materials (Pitts, Ginns, & Errey, 2006). Pitts et al. (2006) describe the cognitive load as the total amount of mental activity that the WM has to attend to at a given instant of time.

### **2.3.1 Working Memory Models**

The limitations of the sub systems of the memory were identified by William James in 1890. James (1890) found that for a state of mind to survive on the memory, it should last on the memory for a certain length of time. James (1890) named this cognitive construct that retains the memory as primary memory (James, 1890). Miller (1954) introduced the term immediate memory of the brain which is a measure of the amount of information a person can retain. The concept Working Memory (WM) in these early definitions was based on a single memory store. Later working memory models evolved as one of the multiple cognitive subsystems responsible for different storages and executive control functions of the brain (Yuan, Steedle, Shavelson, Alonso, & Oppezzo, 2006).

According to Baddeley's original WM concept, WM is comprised of three major components (Ashcraft & Radvansky, 2010). These components include two sensory stores (or auxiliary systems) along with an attention control mechanism. The attention control mechanism is also known as a central processor or central executive (Kalyuga, 2006). According to Baddeley's model, the temporary storage of information, manipulation of information and executive control are the three major functional aspects of the WM (Clark et al., 2006).

The model proposed by Akinson and Shriffrin was slightly different from Baddeley's model (Yuan et al., 2006). This model included Short Term Memory (STM), Long Term Memory (LTM) and Sensory Store. The STM is viewed as a capacity limited, temporary memory store which is used for information processing. According to this model, incoming information is stored in the sensory store and the



information attended to is passed onto the STM while the rest will be lost. The information not rehearsed is decayed on the STM. The rehearsed information on the WM is encoded and saved on the LTM which has an enormous capacity and is long lasting (Yuan et al., 2006).

In the early models of Working Memory, it was considered as a single memory store. But in later models such as the Baddeley and Hitch model (1974), WM is considered as a multi-component system (Yuan et al., 2006). Despite the long history of the concept of the Working Memory (WM), or as some refer to it, the Short Term Memory (STM), researchers have not yet come to a unanimous agreement (Kyllonen, 2002).

### **2.3.2 Advanced Memory Models**

In advanced models, LTM and STM are considered as separate memory stores or single memory stores. These models suggest different modes of activation for both Long Term and Short Term Memory stores (Kalyuga, 2006). Baddeley and Hitch (1974) improved the Modal Model further by adding functional importance in cognitive processing, and replacing the term Short Term Memory (STM) by Working Memory (WM). The WM concept was introduced to account for processing units of information. Baddeley (1986, p. 34) describes the WM as “a system for the temporary holding and manipulation of information during the performance of a range of cognitive tasks”. Logie (1999, p. 174) describes WM as a “desktop of the brain that keeps track of what we are doing or where we are from moment to moment that holds information long enough to make a decision, to dial a telephone number or to repeat a strange foreign word that we have just heard”. The focus of these models is that the WM and the STM have been tested using concurrent processing of several tasks. Such tests are more complex and involve meaningful cognitive operations (Clark et al., 2006).

#### ***2.3.2.1 Atkinson–Shiffrin Memory Model***

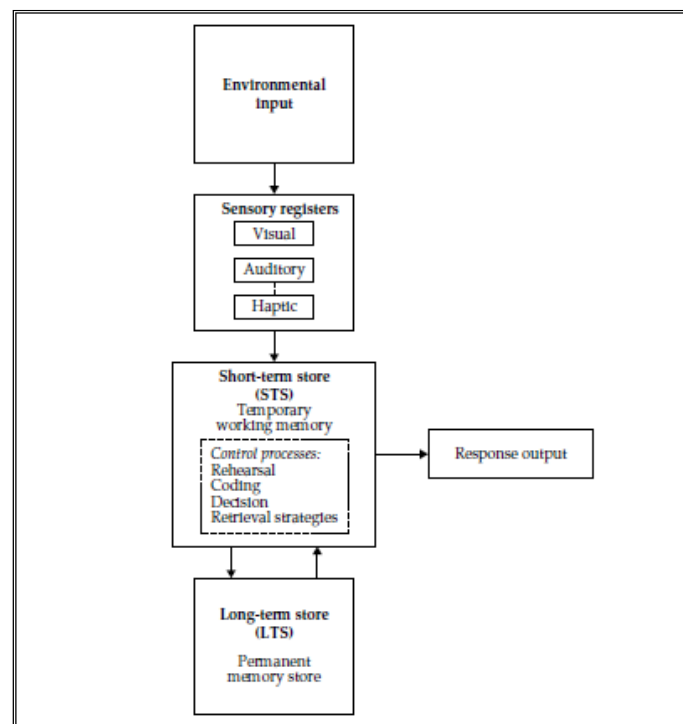
This model was introduced in 1968 by Atkinson and Shiffrin (1968). It is also known as the multi-store model due to the multiple components of the memory. The Atkinson–Shiffrin memory model proposed three memory components: sensory memory, STM, and LTM (Ashcraft & Radvansky, 2010). The sensory memory

component was left out in the original model (Lynch, 2011). The sensory memory receives enormous information from different senses. The types of sensory memory include visual, auditory, and haptic. This is also known as iconic sensory memory which holds visual information. Auditory or echoic sensory memory holds auditory information. The haptic memory holds sensory memory for touch (Ashcraft & Radvansky, 2010). Most of such information stays on sensory memory for a short period of time and cannot be processed due to the limitations of our memory. If attention is paid to any received information on any of the sensory memory, then it is transferred to the STM. The remembering process begins at this point. Then this information is rehearsed repeatedly and finally it is transferred to LTM (Ashcraft & Radvansky, 2010). The STM has limited capacity and it refers to the information retained on our senses long enough to be used. This model suggests that processed information is transferred to LTM which has unlimited capacity and is long lasting (Lynch, 2011). The iconic (visual) input lasts less than half a second but echoic (auditory) input lasts about three to four seconds on the sensory memory (Lynch, 2011).

#### *2.3.2.2 The Baddeley and Hitch Model*

The Baddeley and Hitch model of Working Memory is a multiple component system (Yuan et al., 2006). According to this model, the WM consists of a sensory store and central executive. One component of the sensory store is used as a temporary storage for acoustic and verbal information and is called an articulatory or phonological loop. This is also referred to as an inner voice. The other component is used as a temporary storage for visual and spatial information and is called a visuospatial sketch pad or inner eye (Clark et al., 2006). The information on sensory store may fade away quickly if the attention is diverted or WM capacity is overloaded (Kalyuga, 2006). Sensory memory is stimulated and processed through our senses such as sight, sound, smell, touch, and taste. This sensory information is constantly overwritten by new inputs unless it is refreshed. Unless this sensory information is attended to, visual information will stay on WM for half a second and auditory information for three seconds (Pitts et al., 2006). In Baddeley's view these two sensory components have a specific set of responsibilities supporting central executive with lower-level processing involved in a task (Ashcraft & Radvansky, 2010). According to Baddeley (2001), WM consists of buffers which are used for

storing coded information. One buffer is responsible for storing verbal information and another is responsible for storing visual and spatial information (Baddley, 2001). The third buffer, called episodic, was introduced recently (Baddley, 2001). The Working or Short Term Memory can be compared with the random access memory (RAM) of the computer. The information is stored as chunks in the WM. These chunks of information could be simple character, numerals, or even complex abstracts and images (Pitts et al., 2006). The episodic buffer integrates information already in WM with information retrieved from LTM. This part of the WM binds different types of information together to form a complete memory (Ashcraft & Radvansky, 2010).



*Figure 2.8.* Atkinson–Shiffrin memory model. Adapted from Ashcraft and Radvansky, 2010, p. 38.

The rehearsed information is transferred from the STM to the LTM. The information on the STM is forgotten or lost through the processes of displacement or decay if rehearsal does not happen (McLeod, 2007). This model is considered to be due to its suggestion that both STM and LTM operate in a similar fashion, but it influenced researchers to carry out further studies into memory models (McLeod, 2007). Realising that the STM is more complicated and instead of considering it as a unitary

store, Baddeley and Hitch (1974) developed this model by further identifying different components of the STM such as the central executive, visuospatial sketch pad, and articulatory loop (McLeod, 2007).

### 2.3.2.3 Kieras Model

There are diverse opinions about what components should be included in the WM (Yuan et al., 2006). For example, in the Kieras model there are four components. The four components are visual, auditory, tactical, and kinaesthetic. (Yuan et al., 2006). Kieras used the Executive Process/Interactive-Control (EPIC) architecture, a software simulator for modeling cognition and action issues about WM. The EPIC included the auditory processor, visual processor, ocular motor processor, vocal motor processor, tactile processor, and manual motor processor as inputs (see Figure 2.9) to the WM (Kieras, Meyer, Mueller, & Seymour, 1999)

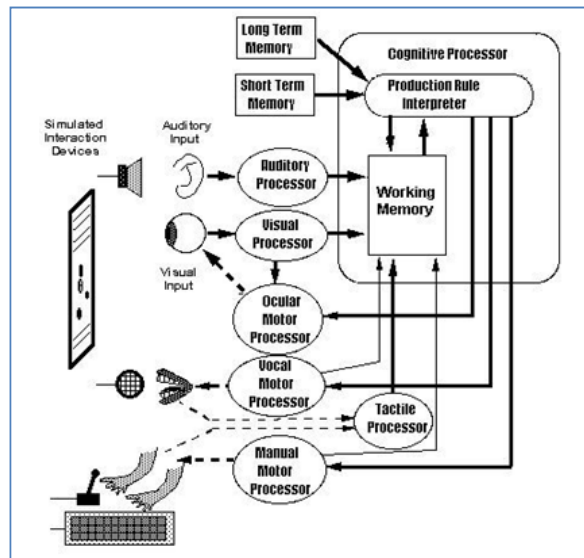


Figure 2.9. Overview of EPIC architecture. Adapted from Meyer, 2011, p. 1.

According to simulated results in EPIC, the mean duration of the auditory WM was two to four greater than the two seconds previously claimed in other models (Kieras et al., 1999).

### **2.3.3 Working Memory Organisation Approaches/Hypotheses**

There are a number of hypotheses that describe the differences in WM and the effect on the performance of individuals (Clark et al., 2006).

#### *2.3.3.1 Total Capacity Approach*

In the total capacity approach, all the cognitive processes get resources from a fixed pool (Clark et al., 2006). It also states that when there is more storage of resources on the WM, it will result in a decline in the processing capabilities of an individual (Clark et al., 2006).

#### *2.3.3.2 Task Specific Hypothesis*

This hypothesis was introduced by Daneman and Carpenter (1980). According to this hypothesis, the WM capacity is specific to a given task when that task is performed (Daneman & Carpenter, 1980). A person with efficient processing skills will have more WM to store processing products. The processing efficiency for a particular task could be achieved by training.

#### *2.3.3.3 Processing Efficiency Approach*

According to the processing efficiency approach, a single central system is responsible for processing and storage. If an individual's processing is inefficient, it takes up more WM, leaving less capacity for storing information (Daneman & Tardif, 1987).

### **2.3.4 Cognitive Load Types**

There are three main types of cognitive loads namely intrinsic, germane, and extraneous which have been identified in cognitive load theory. Mental capacity is limited, therefore, it is important to balance these three cognitive loads to maximise the efficiency of teaching (Clark et al., 2006).

#### *2.3.4.1 Intrinsic Cognitive Load*

The intrinsic cognitive load is the inherent difficulty or the natural complexity associated with the information to be understood and learnt (Chandler & Sweller, 1991). It is a measure of the learner's mental work needed due to the complexity of the lesson (Clark et al., 2006). The intrinsic load is fixed for a given task and knowledge level. It cannot be altered unless the basic task is changed or the knowledge level is altered (Sweller, 2010). Although the intrinsic cognitive load is considered to be immutable, some techniques can be applied to manage complexity by segmenting and sequencing complex material (Sweller, Van Merriënboer, & Paas, 1998). The intrinsic cognitive load depends on the level of element interactivity. The element interactivity is the coordination of several knowledge elements in the memory to accomplish a particular task (Clark et al., 2006). Some learning tasks are learnt in serial fashion while other tasks require coordination. The serial fashion tasks are related to low element interactivity whereas the coordinated tasks require high element interactivity (Clark et al., 2006). An example of an element could be a concept or procedure that has to be learnt (Sweller, 2010). Low interactivity tasks require low WM load due to limited element interactivity (Sweller, 2010). For example, learning to create and use a primitive variable in the Java language requires lower element interactivity than learning to create and use an object, an activity which should impose a much higher intrinsic load on the WM. The intrinsic cognitive load depends on the knowledge skills associated with the instructional objectives needed to teach a particular lesson. Although intrinsic load cannot be altered, it is possible for a teacher to decompose a complex lesson with a high element of interactivity into a series of prerequisite tasks and distribute the supporting knowledge over a series of sub-topics or lessons (Clark et al., 2006). Therefore, experienced professional instructors usually manage intrinsic cognitive load by arranging and sequencing content into a series of instructional events (Clark et al., 2006).

#### *2.3.4.2 Extraneous Cognitive Load*

According to Clark et al. (2006), extraneous cognitive load imposes irrelevant mental work on the learning task to be achieved and is a waste of mental resources. Optimal instruction procedures should not cause extraneous cognitive load for the

WM of the learner. One of the useful applications of Cognitive Load Theory is to use techniques to reduce extraneous load (Sweller, 2004). Sweller (2010) argues that although there is a reasonably clear pattern to the generation of identified cognitive effects, little attempt has been made to identify the cause of extraneous cognitive load. It might have a common underlying cause.

#### *2.3.4.3 Germane Cognitive Load*

The germane load was first described by Sweller, Van Merriënboer and Paas (1998). Both intrinsic and extraneous loads are caused by the characteristics of the learning material. However, the germane cognitive load is caused by the element of interactivity applied to individual learners (Beckmann, 2010). Thus, the germane cognitive load is concerned with the characteristics of the learner and not the learning materials. This means that for the same learning material, a learner with low knowledge levels will have more interactivity resulting in a higher germane load than that of a learner with higher knowledge levels (Sweller, 2010). The germane cognitive load is the load applied while processing, constructing and automating schemas (Sweller et al., 1998).

#### *2.3.4.4 Interactivity between Working Memory Loads*

According to Clark et al. (2006), complex topics are associated with large complex schemas. Such tasks can be broken down into subtasks so that each subtask will be associated with a subschema. Thus, one way of reducing this load is to teach subtasks in isolation and bring them back together as a combined task (Clark et al., 2006).

As Beckmann (2010) suggests, the element interactivity contributes to both extraneous and intrinsic cognitive load. If the element of interactivity can be reduced without altering what is to be learned, then there will be less extraneous cognitive load (Beckmann, 2010). Some information may impose not only intrinsic but also extraneous cognitive load on the learners (Sweller, 2010). The sources of extraneous cognitive loads include those that lead to goal-free, worked example, split attention, and redundancy effects (Sweller, 2010). In instructional procedures that facilitate learning, the number of elements that are to be simultaneously processed by learners should sometimes be reduced. A lesson with high intrinsic cognitive load and low

extraneous cognitive load will have a high germane cognitive load because of the need for the learner's interactivity dealing with essential learning materials (Sweller, 2010). The germane cognitive load takes up a portion of the WM resources for interacting elements which determine the intrinsic cognitive load (Sweller, 2010). Both intrinsic and extraneous cognitive loads constitute an independent source of memory. But, the germane load which is generated due to the interactivity associated with the intrinsic load is not independent and uses available memory resources of the WM (Sweller, 2010). In order to maximise learning, lesson instructions should be organised to let the WM resources deal with the elements related to intrinsic and germane cognitive loads. The effectiveness of learning will be affected if the learner has to deal with elements imposed by extraneous cognitive load (Sweller, 2010).

When more WM is taken up for extraneous cognitive load, there will be less memory available to deal with the intrinsic cognitive load and also the germane cognitive load. When the extraneous cognitive load is decreased, the germane cognitive load will be increased and as a result more memory resources will be available for the intrinsic cognitive load (Sweller, 2010).

### **2.3.5 Knowledge Representation Theories**

There are many theories to describe the way knowledge is represented in the LTM. Such theories explain how knowledge is stored in the brain and how prior knowledge is used later on to acquire and store new knowledge. This concept was first introduced by the psychologist Frederic Bartlett in 1930. The term schema was not new in cognition and was used by Jean Piaget in his theory in 1926 (Pitts et al., 2006). Piaget used the word schema for both a category of knowledge and the process of obtaining that knowledge (Cherry, 2011). In his Cognitive Development Theory, Piaget used the term schema for the organisational structures that manage the sense of experience in children's brains. According to Piaget, such schemas do change with age (Berk, 2003). Bartlett studied human memory using an experimental psychology method. This method was based on using folktales, ordinary prose, and pictures to study the human memory storage of meaningful material. The participants of the survey studied the material for a period of time and were required to recall it several times. This experiment proved that the human memory for meaningful material is not reproductive and but is rather a reconstructive memory. The process



of reconstructing memory includes combining elements from the original material together with existing knowledge (Ashcraft & Radvansky, 2010). Some argue that schema theory was based on Ausubel's (1976) assimilation theory (Mead et al., 2006).

The concept of schema is used in psychology and education to describe knowledge representation in the brain. A schema is a model or hypothetical structure that organises knowledge (Pitts et al., 2006). Schema integrates and stores meaning and the relationships of individual experiences in the form of knowledge. Schema is an abstraction of a collection of learners' past experiences and can be applied later on as new, in related contexts. Schemas are individual and can be encoded differently by individuals even if it is created as a result of a shared experience with a group of people (Mead et al., 2006). According to Driscoll (2000), there are three ways that a new experience affects the creation of a new schema and fits into the hierarchy of existing schemas. The first way, accretion, where the new experience fits well to existing schemas and is remembered by the learner with no significant alterations. The second way, tuning, is where the learner's new experience cannot be fully understood in the context of existing schemas and as a result a new schema evolves to accommodate new experience. The third way, restructuring, is where the learner's new experience is quite different from existing schemas and tuning an existing schema is not viable (Mead et al., 2006). The learning process naturally invokes the formation of a new schema which is the key to the development of expertise and the problem solving ability of a learner. Unlike a novice, the expert in a particular subject has relevant schemas which facilitate the processing of information for learning. An expert can easily recognize and use relevant past experiences stored in the brain in the form of schemas to select a suitable problem solving strategy (Mead et al., 2006). A novice in a subject area lacks relevant and useful knowledge in schemas to be used in problem solving (Mead et al., 2006).

Caspersen and Bennedsen (2007) used the word "pattern" for concrete representation of schema. According to this definition, schemas could be chunks, plans, templates, or idioms. The schema representation in program design will be design patterns. For the algorithm design domain, it will be elementary patterns and algorithmic patterns (Caspersen & Bennedsen, 2007). The processing and encoding takes place on the

limited WM of the learner. A schema is capable of holding huge amounts of information. It is important that this be treated as one element of information. When the schema becomes complex, the processing also becomes advanced on the WM. When the processing is completed, the encoded schema is stored on the LTM which is considered to be unlimited (Caspersen & Bennedsen, 2007).

According to Newell and Simon (1972) knowledge is represented by a set of conditional rules. The production rules are stored on the LTM and loaded to the WM to be used whenever conditions of a rule occur. It then triggers action which could change the contents of the WM (Kalyuga, 2006).

Another theory explains the way that knowledge representation is based on production rules. This theory is called ACT-R (Adaptive Control of Thought-Rational), and was introduced by Anderson (1993). ACT-R theory is based on findings in cognitive neuroscience. According to this theory, memory organises individual processing modules to produce cognition (Anderson, 1993). ACT\_R theory suggests two kinds of memory modules called declarative and procedural. The declarative modules consist of the facts such as propositions, images, other experiences of facts and experiences (Kalyuga, 2006). The procedural memories are in the form of production rules which contain skills and knowledge (Kalyuga, 2006).

### **2.3.6 Cognitive Load Theory as a Pedagogy**

In cognitive pedagogy, it is important to keep all three categories of cognitive loads low so that it does not hinder learning and information of schemas. The intrinsic cognitive load is usually an immutable characteristic of the topic to be taught. The teacher can reduce the cognitive load by adjusting germane and extraneous cognitive loads. The extraneous cognitive load can be reduced by improving the quality of teaching materials and examples (Paas, Renkl, & Sweller, 2003). Scaffolding is another useful technique to reduce the germane cognitive load in particular. Thus, the new concepts to be taught could be introduced to students in a sequential and timely manner to reduce both germane and extraneous cognitive loads (Paas et al., 2003).

Muller (2005) used Pattern-Oriented Instruction (POI) to reduce cognitive load in problem solving in computer programming. This involves locating areas with high

element connectivity resulting in high intrinsic cognitive load. Such areas require particular attention to ensure that cognitive load is minimized and as a result learning becomes effective. In POI, different patterns such as idioms, programming patterns, algorithmic patterns and design patterns are identified (Muller, 2005).

The human knowledge base is retained in the form of schemas, therefore, problem solving ability has to be developed by constructing cognitive schemas. The POI approach was aimed at enhancing the “development of algorithmic problem-solving competence through the construction of an effective knowledge base” (Muller, 2005, p. 65). The identified patterns could be re-used in developing algorithmic solutions (Muller, 2005). Muller (2005) argues that learners who understand a problem comprehensively as a whole, perform well in problem solving. Apparently, such learners tend to load more relevant schemas and less irrelevant schemas to the WM while solving problems (Muller, 2005).

### **2.3.7 Past Research based on Cognitive Load Theory**

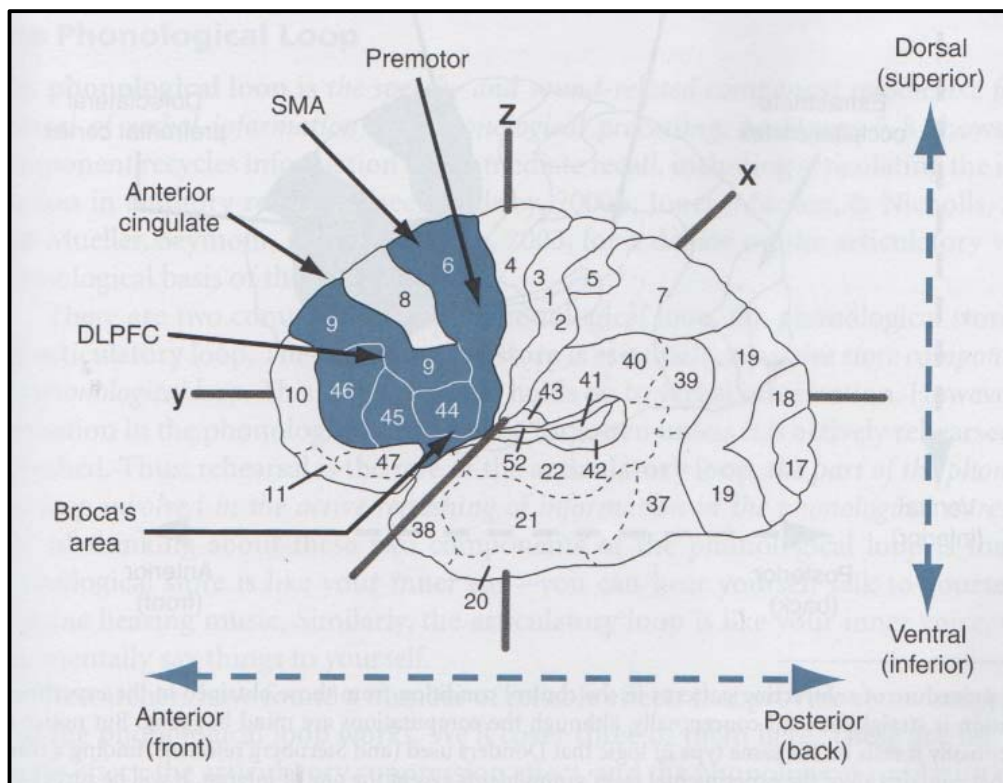
Just and Carpenter (1992) explain individual differences of WM capacity of people. Their findings were based on a study of reading comprehension related to WM. This study uncovered individual differences on “the amount of activation they have available for meeting the computational and storage demands of language processing” (Just & Carpenter, 1992, p. 124). The study revealed the existence of qualitative differences of reading, speed and accuracy of individuals.

Garner (2002), has suggested that the Cognitive Load Theory needs to be carefully taken into account when designing instruction material for teaching computer programming. According to Garner (2002), programming has a very high intrinsic cognitive load which takes up a considerable amount of the limited WM of the learners. A preliminary study using a teaching tool CORT for teaching Visual basic languages, suggested that CORT has reduced extraneous load and has great potential to provide necessary amount of the germane cognitive load to help students develop required programming schemata (Garner, 2002).

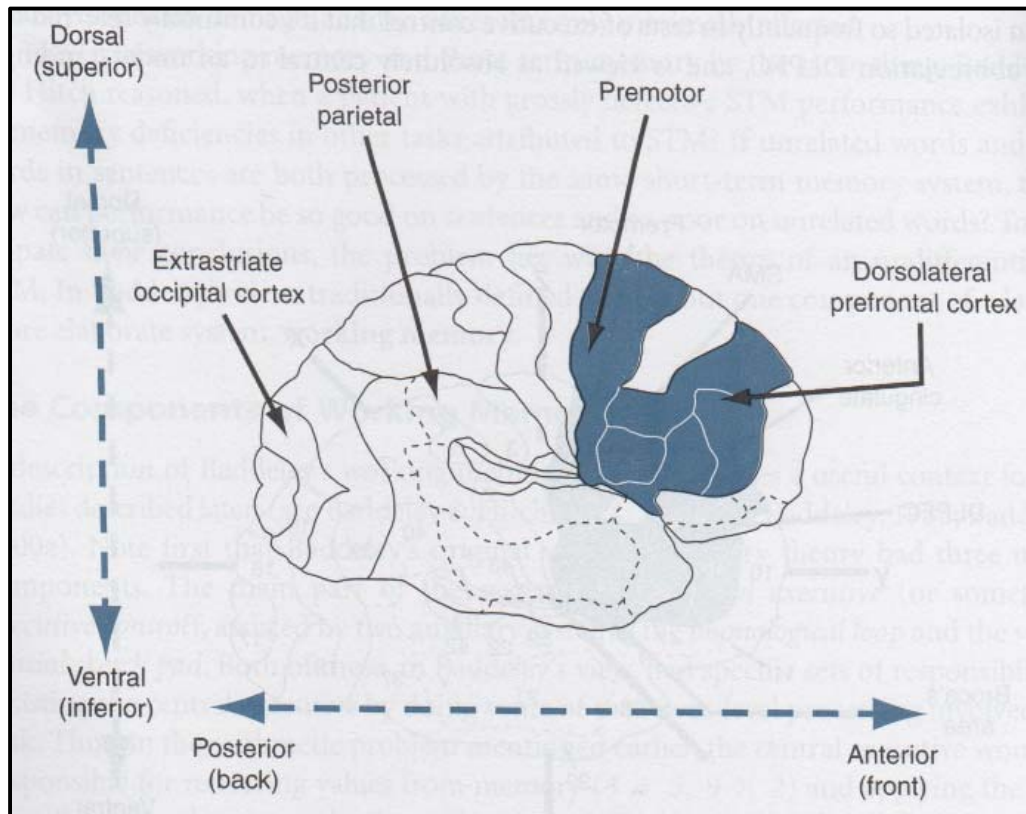
## 2.4 Neurological Aspects of the Human Brain

### 2.4.1 Neurological Research on Cerebral Cortex

Neurological findings on Working Memory (WM) identified different functional regions of the brain. Smith (2000) and Smith and Jonides (1999) used brain imaging techniques to identify regions of the brain where various activities were taking place. The frontal cortex is about 33% of the human brain. It engages in higher cognitive functions of the STM. The executive processes taking place in a part of this WM. This part is “mediated in part by the prefrontal cortex (PFC)” of the left hemisphere of the brain (Smith & Jonides, 1999, p. 1657). The activated areas of the brain for verbal and special activities have been investigated using neurological experiments. The findings of such studies have revealed that the storage of verbal material activates Broca’s area (see Figure 2.10) of the left hemisphere of the brain and storage of special information activates the premotor (see Figure 2.11) area of the right hemisphere of the brain (Ashcraft & Radvansky, 2010; Smith, 2000; Smith & Jonides, 1999).



*Figure 2.10. Left hemisphere regions of the brain. Adapted from Ashcraft & Radvansky 2010, p. 163.*



*Figure 2.11. Right hemisphere regions of the brain. Adapted from Ashcraft & Radvansky, 2010, p. 164.*

#### **2.4.2 Hemispheric Dominance Theory**

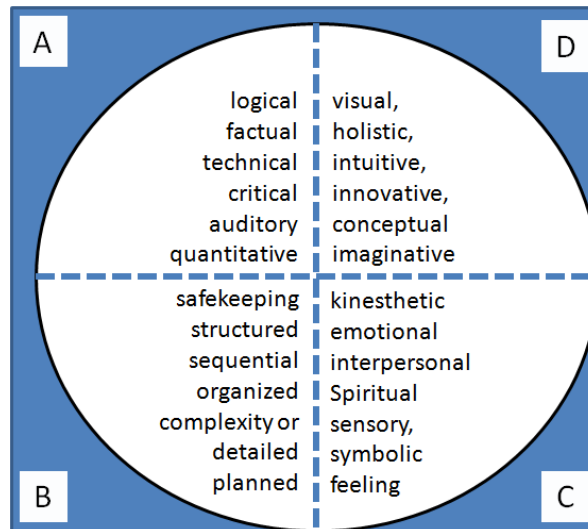
Sperry was awarded the Nobel Prize for his research findings on the cerebral cortex of the human brain (Buzan & Buzan, 2006). According to his findings, the major intellectual functions are divided between the left and the right hemispheres of the cortex of the brain. The right hemisphere is said to be dominant in the areas such as rhythm, spatial awareness, gestalt (wholeness), imagination, day dreaming, colour and dimension. The dominant areas of the left hemisphere include words, logic, numbers, sequence, linearity, analysis and lists (Buzan & Buzan, 2006).

The Herrmann Brain Dominance Instrument (HBDI) is a way of assessing thinking styles of people using 120 questions (HBDI, 2001). The questions are targeted for measuring individual preferences. There are no correct or wrong answers to these questions (HBDI, 2001). This system was developed by William Ned Herrmann. This brain dominance model classifies thinking into four different quadrants (Chwif & Barretto, 2003). The activities of each quadrant are summarised in Table 2.2.

Table 2.2

*Four Quadrants of the Human Brain and Activities. Adapted from Chwif and Barretto, 2003, p. 1995)*

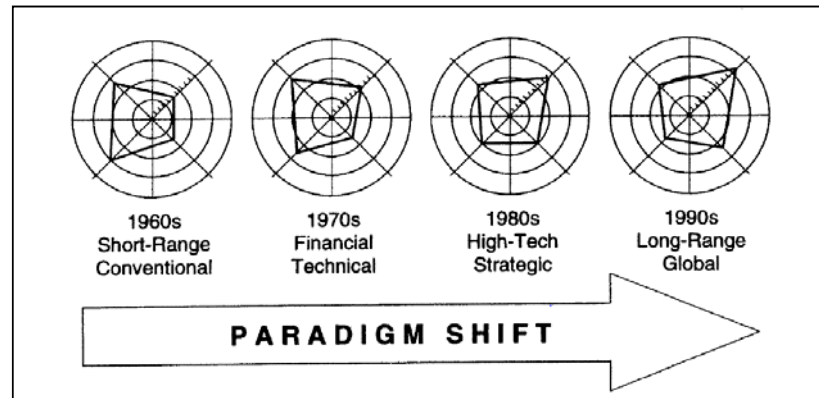
Quadrant	Key Words	Preferred activities
Quadrant A (left brain, cerebral). Analytical thinking  Logical, analytical, quantitative, factual, critical;	logical, factual, technical, critical, auditory, quantitative	understand how things work, collect data, analyse data, judge ideas built on facts, criteria and reason out logically
Quadrant B (left brain, limbic). Sequential thinking	safekeeping structured sequential organized complexity detailed planned	problem solve following directions, detailed orientation of work, organize and implement
Quadrant C (right brain, limbic). Interpersonal thinking	kinaesthetic emotional interpersonal spiritual sensory symbolic feeling	listen and express ideas, look for personal meaning, input sensory, and interact as a group
Quadrant D (right brain, cerebral). Imaginative thinking	visual holistic intuitive innovative conceptual imaginative	look at the big picture, initiative, challenging assumptions, visuals, thinking, creative and problem solving, thinking is long term.



*Figure 2.12.* The four quadrant brain dominance model.

In Herrmann's model, both left brain and right brain are divided into two parts namely cerebral and limbic. Cerebral is the upper part of the brain whereas limbic refers to the lower part of the brain (Lumsdaine & Lumsdaine, 1995).

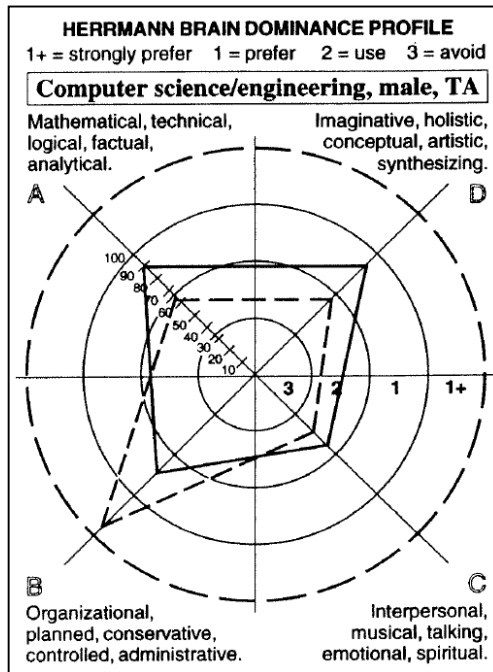
A paradigm shift is a change from one way of thinking to another. Paradigm shifts in education have happened in the past. As an example, a paradigm shift in thinking skills required for success in studies occurred between 1960 and 1990 (Lumsdaine & Lumsdaine, 1995). Figure 2.13 shows that the paradigm had been shifted from Quadrant B to Quadrant D during this period (Lumsdaine & Lumsdaine, 1995). Sung (2010) argued that in the 21<sup>st</sup> century a paradigm shift is to use active learning with collaboration using integrated teaching models. In the 21<sup>st</sup> century, teachers are not the sole provider of knowledge as students have access to information available on the internet (Larson, Miller, & Ribble, 2009). This will lead to a huge paradigm shift in education with the teacher playing a facilitator role rather than teaching in the traditional sense. When there is a paradigm shift in any subject area, long lasting education reforms will be required (Ferrero, 2005). The education system shifted from teacher centered to student centered in the last two decades due to increasing focus on teaching practice (Pears et al., 2007).



*Figure 2.13. Paradigm shift from 1960 to 1990. Adapted from Lumsdaine and Lumsdaine, 1995, p. 195.*

The paradigm shift of students from the beginning of the course to the end of the course was studied at Michigan University (Lumsdaine & Lumsdaine, 1995). The Herrmann Brain Dominance Instrument (HBDI) was used for students in different engineering faculties. The different results were found in different programmes offered at Michigan University. The quadrilateral with the dotted line in Figure 2.14 shows the brain dominance profile of computer engineering students at the beginning of the course and the quadrilateral with solid line shows their brain dominance profile in the final year of the computer science and engineering students (Lumsdaine & Lumsdaine, 1995). According to this result (see Figure 2.14), the students who studied computer science engineering have become more left-brained oriented thinkers at the end of the four year course. This was achieved by getting students involved in additional creative problem solving activities throughout the course (Lumsdaine & Lumsdaine, 1995).





*Figure 2.14.* Brain dominant profile of computer science and engineering students.  
 Adapted from Lumsdaine and Lumsdaine, 1995, p. 201.

Logical, analytical skills, which are sequential in nature, are thought to be required for computing. McCluskey and Parish (1993) tested the effect of learning HyperCard by right-brain dominant, left-brain dominant, and mixed-brain dominant students. The findings were quite the opposite to the expected notion that the left-brain students do cognitively-oriented computer tasks better than those with right brain or mixed-brain (McCluskey & Parish, 1993). When this experiment was done, the eyedness was the measure used to categorise students' brain dominance.

## 2.5 Mental Models

Mental models were introduced by the Scottish psychologist Craik in 1943. Craik proposed three distinct processes involved in reasoning in the human brain. The first process is the translation of external processes into an internal representation in terms of words, numbers and figures. The second process is the derivation of other symbols from them by some sort of inferential process. The third process is the retranslation of the symbols into actions or recognition of the correspondence between these symbols and external events (Johnson-Laird, 1983).

### **2.5.1 Past Research on Mental Modeling**

People usually use visual images, pictures or scenes when they are using their imagination. These are psychological representations of real, hypothetical, or imaginary situations (Johnson-Laird et al., 1998) and are internal constructs that are stored on our memory. The mental models work conscientiously or unconsciously in a human brain when we think and respond (Carlson, 2007). Once the mental models are accepted as reality, then they are powerful and influence the way humans interact with others. It is suggested that such models work effectively in churches and temples and help to retain scriptural interpretation and historical traditions. Some operations can be carried out on imaginary models and they can be related to corresponding conceptual processes (Johnson-Laird et al., 1998).

Many researchers have recently experimented with the possibility of using mental modeling with students to enhance teaching. Ma et al. (2007) used a questionnaire with open ended questions where novice Java programming students were asked to describe the execution of a small program using texts or diagrams. These questions were unstructured and were aimed at getting unanticipated information of each participant's mental models. The questionnaire contained some multiple choice questions which predicted the answer of a given set of small programs with pre-defined answer options. The data collected from the multiple choice questions were mapped to possible mental models. In this survey, the most novice of the programming students had non-viable mental models on the use of reference variable and a few had non-viable mental models on the use of assignments of values to variables. It was concluded that the reason that many failed the introductory Java programming course was their lack of understanding of the reference variable. In this research, the researchers have highlighted the importance of helping students to develop viable mental models (Ma et al., 2007).

Visualisation tools are useful in creating mental models and help students to understand concepts of software development. For example, visualisation could be useful to help students to understand the concept of an object state and object identity versus equality in OO programming (Van Haaster & Hagan, 2004). According to Ben-Ari (2001b), when a teacher uses a visualisation tool to teach abstract concepts, there are intrinsic difficulties in synchronising the mental models

of the student with those of the teacher. Therefore, a common starting point is required to develop a shared mental model of a concept for both the teacher and the students (Norman, 1990). Some researchers have used visualisation software tools such as Alice to successfully teach introduction to objects, methods, decision statements, loops and recursion (Dann, Cooper, & Pausch, 2001).

Mental models represent an abstract concept. The conceptual representation is unique to an individual and provides predictive and explanatory powers in understanding the concept (Wu et al., 1998). Teachers define the conceptual model, which should represent the topic to be taught, as the sense of being accurate and complete in a consistent way (Norman, 1983). If the learner has an accurate mental model of what has been learnt it means that the learner has understood it and learning has been successfully completed (Wu et al., 1998). Wu et al. (1998) argue that the teacher needs to develop a conceptual model which facilitates students in developing their own mental models.

### **2.5.2 Constructivism and Mental Models.**

Constructivism is a way of constructing human learning. The learners build new knowledge upon the foundations of previous knowledge. The key thinkers of constructivism were Piaget, Vygotsky, and Bruner (MacNaughton, 2003). The Theory of Social Constructivism is a dominant philosophical theory introduced by Kant. Social constructivism is based on a mental modeling concept. Kant concluded that humans construct knowledge by organising and sequencing the experiences that they gain from the outside world (Werhane et al. 2011).

Constructivism is an educational concept which helps learners to construct mental models (Lui, Kwan, Poon, & Cheung, 2004). Lui et al. (2004) used cognitive science to construct mental models of programming elements in the mind of learners. The learners who managed to construct viable mental models which matched the design model understood the correct concepts and became successful in learning programming. According to Ben-Ari (2001a), learning in a constructivist way is effective and demands the construction of viable models. The construction of mental models is a recursive process in which new models are constructed and existing ones

are adjusted or dropped. Five hazards that could hinder learning computer programming have been suggested by Lui et al. (2004).

They are:

1. presentation of high fidelity programming interfaces on test books and lecture notes;
2. abstract symbols and implicit concepts in languages such as C and Java are difficult for novice learners;
3. frustration of novices due to time spent on editing, compiling, and running to check each mental model;
4. lack of prior knowledge or correctly constructed knowledge on which to construct new knowledge; and
5. having unsuitable pieces of knowledge as the basis to construct new knowledge.

Hazards four and five are related to the construction of new mental models by students. Lui et al. (2004) suggest that weak students are less tolerable of these hazards. The learners get access to mentally designed models using the interface provided by the language, programming environment, lecture notes or lecture contents. The learners interact with the interface through which they have to probe the actual model in order to construct and test the model (Lui et al., 2004).

### **2.5.3 Dual Coding Theory and Relational-Organisational Hypothesis**

Yates (1966) argues that imagery was used and applied in a broader sense with the aim of accelerating the acquisition of knowledge even before Christ (as cited in Paivio, 2006). With the increase of language emphasis in education, imagery was eventually externalized as pictures (Paivio, 2006). Dual coding theory and its educational implications further enhance the historical evidence of centralisation of knowledge using imagery and pictures (Paivio, 2006). The use of images in cognitive processing has been investigated in the last three decades (Ryu, Lai, Colaric, Cawley, & Aldag, 2000).

Recent research led to the revival of the use of imagery in education and the formulation of the Dual Code Theory (DCT). The Dual Coding Theory (DCT) is

based on the assumption that visual and verbal information is processed, encoded, stored, and retrieved for subsequent use by different channels of the brain (Paivio, 2006). Although verbal memory and image memory store and function independently, both work interactively (Thomas, 2010). Verbal information contains the item's linguistic meaning and the visual images represent what the item looks like (Liu, 2011). Paivio, an emeritus professor of psychology at the University of Western Ontario, experimented by giving pairs of words to a group of people and checking how they could recall them. Baddeley appreciated and adopted DCT in his WM model. In Baddeley's Working Memory Model, the two memory stores were named visuospatial sketchpad and phonological loop.

There are some other controversial theories that have been proposed by others (Thomas, 2010). The Relational-Organisational hypothesis is an alternative to Dual Coding Theory. According to this hypothesis, humans create a number of links or hooks between the items of pairs which are required to remember paired-associated information. Bower (1970, as cited in Liu, 2011) researched this with three groups of participants each with a different set of instructions in a pairs associated task. For group one, two items were rehearsed aloud. The second group was asked to construct two images which were not interactive and the two items were separated in a marginal space. The third group was asked to construct an interactive scene with two images which were interactive. Group three managed to recall 53% of the paired associates and group two was able to recall only 27% of the paired associates. Bower argued that if DCT was true, both group two and three would have performed equally. Based on these findings, Bower concluded that interacting images create more links between target information and other information, making it easier to retrieve (Liu, 2011).

#### **2.5.4 Using Visual Tools in Teaching**

Critical thinking skills require logical thinking and reasoning which includes sequencing, classification, deductive and inductive reasoning, comparison, hypothesizing, cause/effect, patterning, webbing, analogies, forecasting, planning, and critiquing (Dunbar, 1997). Critical thinking is considered as a functionality of the left-brain (Ursyn & Scott, 2007). Images and animations are visual learning tools that can be used to enhance learning at any level. Visual representations help to

bridge language barriers (Willis & Miertschin, 2005). Ursyn and Scott (2007) state that visual ways of thinking related to simulation and visualisation give rise to the ability to perceive complex systems. Communication through visual symbols is nonlinear and quite different from communicating using verbal symbols. Such nonlinear processing involves cognitions and produces personal referents and insights. Hence, visual symbols help to develop creative thinking (Ursyn & Scott, 2007). Many researchers believe in using visual tools for reducing the complexity of an intellectual task. Visual tools help the learner to understand abstract ideas and understand processes, concepts, misconceptions and tasks in their own way (Krajcik, Czerniak, & Berger, 2003). Most programming concepts are abstract with no graphical form. Therefore, teachers tend to use visual representation of the structure and operation of programs and algorithms to make them easier to understand for novices in programming. However, “Students may look at dynamic visualizations without understanding the context or deeper meaning” (Pears et al. 2007, p. 209).

## **2.6 Instructional Techniques and Tools**

### **2.6.1 Scaffolding**

Scaffolding for supporting learning was introduced by Wood, Bruner and Ross (1976) as a metaphor to explain the one-to-one assistance that teachers provide to learners. In scaffolding, the teacher provides assistance only if the required skill is beyond the learner’s capability (Lipscomb, Swansonm, & West, 2008). Educators widely use scaffolding to assist students’ learning. Educators get students engaged in a collaborative manner by providing scaffolding with structures for learning. These support structures are necessary to complete tasks and to develop the knowledge structures for the students who need support. In scaffolding, teachers provide clear directions, use methods to keep the students on task, and do sporadic assessment with feedback (Mead et al., 2006). Six categories of scaffolding namely: instructing, questioning, modeling, feeding back, cognitive structuring, and contingency management were identified by Gallimore and Tharp (1990). Krajcik et al. (2003) describe scaffolding as the process of providing support directions by a more knowledgeable person for an intellectual task which is beyond the learner’s capacity at the beginning. The scaffolding concept is that the support given to a learner needs to be gradually reduced as the learner internalises the knowledge and

skills. In this process, the responsibility for completing the task is transferred from the teacher to the learner (Puntambekar & Hubscher, 2005).

#### *2.6.1.1 Use of Anchor Concept Graph in Scaffolding*

The Anchor Concept Graph is a useful structure for educators to use in scaffolding as it shows the required building blocks needed to understand a concept. A node of a concept graph represents the knowledge structure to be developed by a student. Students must traverse a path of the graph through intermediate nodes to reach the goal node. The purpose and the graph direction must be clear and well defined. There could be more than one path on the graph to reach a goal node. Teachers use the graph structure to keep track of each student's traversing to make sure that students reach the final goal (Mead et al., 2006). Mead et al. (2006) suggest intermediate nodes as ideal locations for intermittent assessments in scaffolding.

#### *2.6.1.2 Use of Distributed Scaffolding*

The distributed scaffolding concept is usually applied when different ways of meeting development needs are required. It may involve a different type of knowledge, communication and a large assortment of learning or support. Tabak (2004) argues that the tasks need to be extended over a long period of time due to their complexity. When scaffolding takes place over a period of time, the student's scaffolding needs may change. Therefore it is necessary to meet the changing needs of the student accordingly (Tabak, 2004). Redundant scaffolding is a distributed scaffolding pattern in which a learner receives multiple supports of different types (Tabak, 2004). The Zone of Proximal Development (ZPD) was defined by Vygotsky as the gap between the learner's independent problem solving level and the level which they are expected to attain (Wertsch, 1985). Tabak (2004) argues that redundant scaffolding is the solution to students with varied ZPD within the same class.

#### *2.6.1.3 Collaborative Learning Support using Scaffolding*

According to Stahl (2006), mind tools can be used for scaffolding with shared cognitive processes for a group of learners in a collaborative manner. Using mind tools for a group of people could be more complex than using them with individuals

due to multi-interactivity and reinterpretation of the meaning among the members of the group. Stahl (2006) used chat tools for communication along with a whiteboard to collaborate ideas. Information and Communication Technology will continue to be used as a mind tool in collaborative learning and scaffolding (Kirschner & Erkens, 2006).

#### *2.6.1.4 Cognitive Learning Support using Scaffolding*

From the pedagogical view of cognitive load theory, it is important to adjust the germane cognitive load while maintaining extraneous cognitive load at a minimal level in teaching. This can be achieved by “modifying teaching materials and examples, by using scaffolding, and by carefully sequencing and timing the introduction of new concepts” (Mead et al., 2006, p. 186). A research paper written by Caspersen & Bennedsen (2007) was based on the use of cognitive load theory, cognitive apprenticeship, and worked examples to improve the teaching of computer programming. Scaffolding was used to support students as they continued with the tasks. As the learning progresses, fading was eventually applied to hand over the responsibility for performing the task to the learner. This instructional design has been used successfully at University of Aarhus in Denmark with more than 400 students enrolled in introductory level programming (Caspersen & Bennedsen, 2007).

#### *2.6.1.5 Scaffolding using Visual Tools*

Some intelligent software tools have been used to provide scaffolding for learners. Garner (2002), used a software tool called the Code Restructuring Tool (CORT) for scaffolding at Edith Cowan University, Australia. The students were provided with partially complete solutions of assignments and the CORT provided online scaffolding while completing the assessments. CORT provided four categories of support types, namely, syntactical, semantic, structural, and algorithmic. The CORT determined the degree of assistance required and provided strong scaffolding for student learning. According to Garner (2007), it is important to design partially-completed problems in such a way that scaffolding is reduced gradually in order to maintain the cognitive load at a moderate level.



At the University of Houston in the USA, mind mapping was used as a visual tool in team collaborative learning (Willis & Miertschin, 2006). The process consisted of three-stage scaffolding. In the first stage, scaffolding was provided to individual students who created mind maps independently. In the second stage, mind maps were exchanged between team members and for peer review. Scaffolding was also provided to the team members who had developed a new knowledge structure collaboratively at the second stage. In the last stage the students had to show the proficiency of the applied contents and how concepts were integrated. At this stage scaffolding was provided to the students. The scaffolding at each stage was provided by means of the assessment tasks given to students for completion (Willis & Miertschin, 2006).

Mind tools and cognitive tools transform information into knowledge. These tools can be simple or complicated: ranging from email to visualization systems. They are used to “engage in, and facilitate, critical thinking and higher order learning” (Kirschner & Erkens, 2006, p. 199). Students use mind tools to represent what they know in different meaningful ways. Therefore a teacher can use mind tools for scaffolding. The messages between the teacher and learner also provide a scaffolding structure when specific kinds of responses are used. This form of scaffolded conversation results in more consistent and convincing conversations (Jonassen, Carr, & Yueh, 1998).

### **2.6.2 Cognitive Apprenticeship and Metacognition**

The members of communities which are bound by a shared set of interests usually follow the cognitive apprentice model of learning (Dennen, 2008). The cognitive apprenticeship is part of the social constructivist paradigm. In cognitive apprenticeship, the principles of ZPD are applied with tasks that require scaffolding (Cognitive apprenticeship, n.d.). As in traditional apprenticeships, cognitive apprenticeship students are expected to demonstrate skills with assistance and coaching. A cognitive model which could be used to develop reasoning abilities of the learners by making expert thinking in a visible subject area is a cognitive apprenticeship model (Collins, Brown, & Holum, 1991). Some researchers use the word metacognition, which was introduced by Flavell (1976), to describe cognitive apprenticeship. It refers to a person’s knowledge concerning his own cognitive

process or anything related to it (Flavell, 1976). Collins, Brown, and Newman (1989, p. 456) describe cognitive apprenticeship as “learning through guided experience on cognitive and metacognitive, rather than physical, skills and processes”. Another definition which is more related to traditional apprenticeship is that “Cognitive apprenticeship is a process by which learners learn from a more experienced person by way of cognitive and metacognitive skills and processes” (Dennen & Burner, 2007, p. 427). In the Cognitive Apprentice Learning Model, cognitive skills are developed through interactions using activities such as modeling, coaching, reflection, articulation, and exploration (Collins et al., 1989; Dennen & Burner, 2007; Seel, 2001). Cognitive apprenticeship could be provided to learners on computers creating simulated apprenticeships in a multimedia environment. According to Reeves (1993), it is a major benefit to have well designed interactive simulated apprenticeships in the classroom environment.

### **2.6.3 Situated Learning Theory**

Situated Learning Theory (SLT) was proposed by Lave and Wenger. According to SLT, learning takes place within activity, context and culture. According to Lave and Wenger (1991), this theory is related to Vygotsky’s social development (Learning Theories Knowledge base, 2011). Brown et al. (1989) describe the context of situated learning and the way that it can be applied to real life and for meaningful learning; it is required to embed the social and physical context within which it can be used. Lave (1988) identified three categories of learners: Activities of students, practitioners and Just Plain Folks (JPF) and studied the learning patterns of each category (as cited in Brown et al. 1989). Lave (1988) focussed on JPFs and found that the ways they learn were quite distinct from what others do. They can acculturate through apprenticeship or qualitative change as others have in a conventional way. Lave (1988) found that JPFs acculturate into different communities (as cited in Brown et al., 1989). Brown et al. (1989) believe that JPF behaviour should be discouraged in schools. Although Situated Learning Theory is logical and easily explained, it is difficult to implement such ideas practically in instructional settings (Herrington & Oliver, 1995). The biggest challenge in the situated learning model is the observation and identification of the community of practice of learners (Lave & Wenger, 1991). When computer applications are used as teaching tools, it takes learners away from the real life work situations. Therefore, it

becomes a new learning environment away from authentic situations and hence leads to a setback in situation learning context (Hummel, 1993). Hummel (1993) suggests virtual reality and hypermedia be used in computer learning applications to simulate real life work situations.

#### 2.6.4 Mind Mapping

In the late 1970s, Buzan (2011) defined mind mapping as “a powerful technique which provides a universal key to unlock the potential of the brain” (p. 1). In an educational context, Martin (2007) described mind mapping as a technique which combines graphical and textual components for studying and planning. It ties together a range of cortical skills with word, image, number, logic, rhythm, colour, and spatial awareness in a powerful manner. Unlike most other tools, mind mapping allows the user to freely associate with new ideas of his brain. The open ended nature of the mind map allows the user to make new connections (Buzan, 1991). This tool is currently used by millions of people in the world (Buzan, 2011).

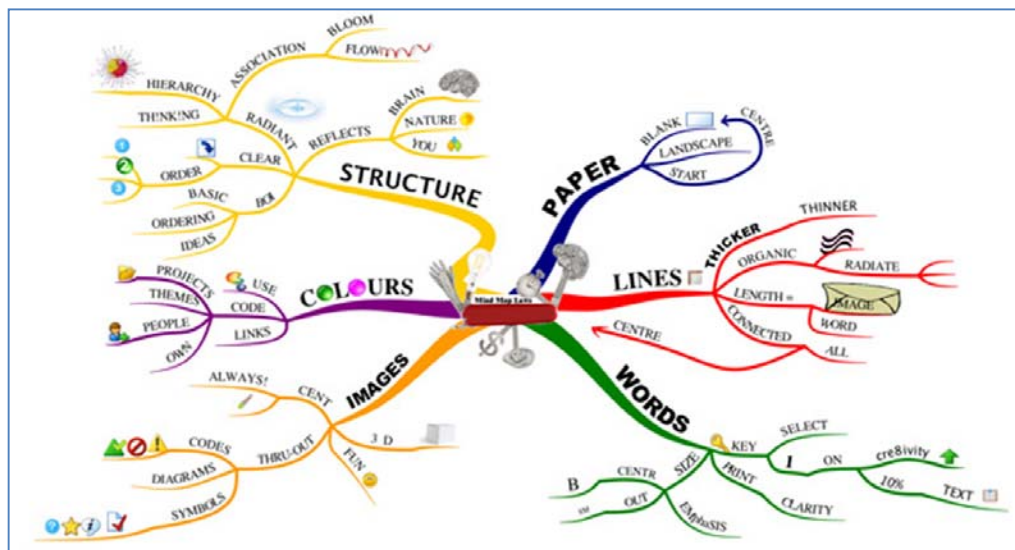


Figure 2.15. Sample mind map. Adapted from Buzan, 2011, p. 1.

According to the guidelines given on Buzan’s website, mind maps should start in the middle of a blank page, allowing the brain to express freely and naturally, allowing ideas to spread out in all directions across the page. A central coloured image or picture may be used to help focus and concentrate the user’s imagination. Colours should be used throughout as they help creative thinking and add extra vibrancy to

the brain. The main branches should be connected to a central image and the second and third level branches all connect to the first level. This helps the user's brain as it works by association. The connections help the creator to understand and remember the contents a lot more easily. The branches should be curved as straight lines could be boring for the brain. Single key words should be used per line to add more power and flexibility to the mind map. Images or pictures should be used throughout the mind map (Buzan, 2011). It is also important to use colours throughout the mind map as colours enhance memory, delight the eye and stimulate the right cortical brain. In addition, it is important to use single headed or multi-headed arrows to show forward and backward directions of the contents of the mind map. Symbols such as asterisks, exclamation marks, crosses, and questions marks are usually used next to words to show connections and clarity. Geometrical shapes such as squares, circles, oblongs, and ellipses are used to mark areas or words which are similar in nature. They are also used to show and classify the sequence of importance. Triangles can be used to indicate possible solutions on the mind map (Buzan, 2006). Historically, mind mapping was popular among researchers for generating ideas in loosely structured brainstorming sessions. In such sessions, ideas could be categorised informally in the branches of a mind map. At the end of the brainstorming session, ideas from different branches of the mind map were reviewed to create important ideas (Millen, Schrieffer, Lehder, & Dray, 1997).

#### *2.6.4.1 Use of Mind Mapping in Collaborative Learning*

Mind Mapping has been successfully used in computer-supported collaborative learning (CSCL) at Saint Petersburg State University, Russia. Now mind mapping is a widely known learning tool especially in CSCL (Koznov & Pliskin, 2008). Willis and Miertschin (2006) used mind mapping to experiment on collaborative group learning. Tablet PCs (TCP) were used to investigate the possibility of incorporating mind mapping activities in order to improve the critical thinking of Information Systems Technology students (Willis & Miertschin, 2006). The use of mind mapping in enhancing peer interaction aiming at collaborative learning was studied at the Saint Petersburg State University, Russia using 200 undergraduate level student participants. In this study, students firstly created individual mind maps to develop the understanding of their module topics, and secondly exchanged their mind maps with the team members for peer review with comments which aided the refining of

their understandings. At the end of this study, the researchers noticed “booming efficiency of collaborative learning processes and student activity, and the entire education process seems to have become more creative and interesting” (Koznov & Pliskin, 2008, p. 488). Some researchers consider Mind Mapping as a tool for organising meanings. Learners analyse and organize what they know or what they are learning using semantic organization tools. Mind Mapping is the best known semantic organization tool (Jonassen et al., 1998). There are newly developed simple teaching tools with embedded free hand manipulation features. GroupScribbles (GS) is one such design tool for teachers to support student knowledge building using collaborative learning (Tan, Chen, & Looi, 2009). Jakovljevic (2003) believes that current methods for teaching programming fail to give in-depth understanding of programming concepts and the use of mind tools has not properly been investigated in the present programming classrooms.

The Vodafone research and development group discovered the benefits of using mind mapping use-cases opposed to the linear use-cases. Despite the surprising similarities between two use-casings, the group found a number of benefits in using mind map use-cases. One advantage was the ability to include vague information at an early stage in the project. In Mind Mapping no information gets lost during extensive and/or confusing use case elaboration. The Mind Map use-cases indicate a specific style of thinking. Therefore it can be leveraged in other phases of requirements elicitation. The fast swap between big picture specification and every single detail is another advantage in mind map use-cases. It was also found that the packages could be easily rearranged and package alternatives were simple and able to be identified and adjusted. Mind map use-cases enhance hithero use-case writing practises (Holtel, 2005).

Scribbles, a software package, was used at the School of Computing at the University of Dundee for object-oriented design. This software package is capable of recognising hand drawn shapes and enables the use of freehand manipulation of a hybrid mind map. The mind map provides a platform for brainstorming and eventually that platform unlocks the capacity to explore ideas across a number of dimensions and levels of ideas. It is also possible to store valuable ideas and documentation in the early stages of program design using mind maps. This stored

information would be useful at the design stage of the software project. This is a very lightweight modeling environment ideally suited to introducing students to object-oriented design (Martin, 2007). Martin (2007) prefers this light weight tool to Computer Aided Software Engineering (CASE) tools due to its hybrid nature with mind maps and unlike CASE tools it allows free hand manipulation of mind maps (Martin, 2007).

The Computer Science Department of University of Wales, Aberystwyth (UWA) embarked upon a project called MindMapX. This product involves a real time multiuser mind mapping application that could not only be used in software engineering but also in other disciplines. This product enables students to collectively plan, learn and develop their ideas in software engineering which includes object-oriented programming (Davis, 2005).

Jonassen et al. (1998) categorised mind tools into several classes; namely semantic organization, dynamic modeling, information interpretation, knowledge construction, and conversation and collaboration. Mind mapping comes under the category of semantic organization tools. Semantic networking tools contain visual screen tools to produce concept maps. Concept mapping is a learning strategy in which visuals map of concepts are drawn and then connect to each other by using lines. As Jonassen, Beissner and Yacci (1993) describe, this structural knowledge is stored in the memory as spatial representations of ideas and their interrelationships. It is more beneficial to use computers as mind tools using appropriate software as knowledge representation formalisms rather than using computer-based instructions (Jonassen et al., 1998).

One singular advantage of mind mapping is that having the main topic at the centre, it becomes more clearly defined. The more important ideas are closer to the centre and less important ideas are near the edge of a branch. The links between key concepts are easily recognisable on a mind map. The structure of the mind map helps recall and review contents more effectively. The structure of the mind map allows new additions without much effort and each mind map is unique.

## **2.7 Issues in Teaching and Learning OO Programming**

### **2.7.1 Past Research on Teaching Issues in OO Programming**

#### *2.7.1.1 Cognitive Issues in Teaching*

According to Winslow (1996), the dropout rate was highest in programming courses at the University of Dayton, Ohio. Such courses were regarded as difficult. Winslow (1996) argues that novice programmers face a wide range of difficulties and deficits and they need at least 10 years of experience to become an expert programmer. Winslow (1996) believes that a novice programmer's knowledge is limited to surface and superficially organised knowledge. Winslow (1996) found that novices were lacking detailed mental models which are useful program chunks or structures in programming. Robins, Rountree, and Rountree (2003) believe that many of the strengths and abilities of programming experts are due to their ability to recognise and adapt patterns or schemas. Therefore learning programming requires not only knowledge about the language, but also the ability to comprehend programs and to generate programs (Robins et al., 2003).

At the Open University of Hong Kong, it was found that about 10 to 20 percent of the introductory level programming students were unable to understand the fundamental programming concepts and as a result they didn't go past the first assignment every year (Lui et al., 2004). After a few years of experience, teachers realised that the reason for failures was due to their incorrect understanding of the programming concepts. A group of researchers developed courseware for such students using an approach called Perform Approach, based on cognitive science theories and constructivism. In the perform approach, weak students were provided with different support schemes for concept construction. This rigorous process of learning helped to avoid weak students constructing incorrect concepts (Lui et al., 2004).

Van de Ven and Govers (2007) conducted a survey at the Eastern Institute of Technology in Napier, New Zealand to find out ways by which the effectiveness of teaching and learning could be improved at an introductory level in computer programming. It was aimed at finding the most difficult parts in teaching programming, in order to identify factors which could overcome difficulties. The

result of the survey revealed that most students became frustrated and lost their motivation as they did not have the background to understand the high level of new content presented at the start of the course.

White and Silvitanides (2002) argue that students require a formal operational level of cognitive development in order to learn a programming language (as cited in van de Ven & Govers, 2007). According to White and Silvitanides, students formulate different programming environments and require different levels of cognitive load for different languages (as cited in Van de Ven & Govers, 2007). The literature review carried out by Prasad and Fielden (2003) came to the conclusion that adapting the teaching to the different cognitive styles of the students may have the greatest chance of success in computer programming.

#### *2.7.1.2 Use of Worked Examples in Teaching*

Garner (2002) believes that the nature of computer programming results in a high intrinsic cognitive load on the learner's memory and emphasises the need for reducing the intrinsic cognitive load of learners. Garner has suggested taking Cognitive Load Theory into account when developing instruction materials for programming students (Garner, 2002). According to Garner (2002), novice programming students could be provided with complete worked examples in order to reduce extraneous cognitive load. Afterwards, when students are given incomplete assignments to complete, they will abstract and use the relevant schemas from the LTM which were created while studying worked examples. Past research suggests that incomplete, but well-structured and understandable examples be given to students to generate missing codes or to complete examples. Such incomplete examples should be carefully designed to include sufficient clues to guide students to complete the work. The aim of this careful design of the examples should be to minimise the germane cognitive load of the task imposed on the learner's WM. In addition, it is also important to make sure that the blueprints are found for mapping to a new problem situation and it forces mindful acquisition of relevant schemas to the new problem when students start working on the example (Garner, 2002).



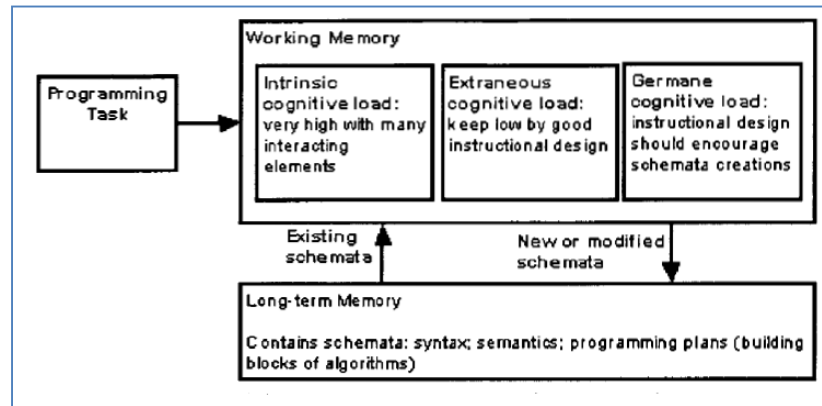


Figure 2.16. Cognitive load relationships in programming. Adapted from Garner, 2002, p. 4.

An instructional format using worked examples reduces unnecessary cognitive load imposed in conventional instructional format. It has been proven to be effective for novice learners (Van Gog, Paas, & Sweller, 2010). According to Koedinger and Alevan (2007), learners receive enormous amounts of instructional guidance from worked examples compared to problem solving exercises. The amount of guidance to be added to the worked examples was studied by Wittwer and Renkl (2010) who found that it had little effect on gaining procedural knowledge but was useful in understanding conceptual knowledge. The worked example itself contains the solution procedure for the learner to study.

### 2.7.1.3 Use of Cognitive Tools in Teaching

Garner (2002) experimented with a teaching tool called the Code Restructuring Tool (CORT) to improve teaching Visual Basic programming at introductory level students. The CORT was based upon using partly completed solutions of programs in teaching. The CORT had three levels of using code which provided different degrees of the germane cognitive loads to students. As discussed earlier, the CORT was tested at Edith Cowan University in Australia. The researcher designed teaching materials taking the principles of the Cognitive Load Theory into consideration. Due to the high intrinsic cognitive load in programming topics, it was realised that the need for lowering the extraneous cognitive load was important. The partly completed work should make students think, apply germane cognitive load and finally create a new schema in the LTM as new knowledge. This process had been incorporated into the software tool CORT. There are different views and suggestions

for using partly completed programs in teaching. Van Merriënboer and Paas, (1990) suggest using understandable and well-structured program examples with sufficient clues so that students understand and complete the work. This would enable students to acquire schemas to be used for mapping new problem situations. Another suggestion from Lieberman (1986) is to give worked examples to students and ask them to annotate with the detailed functionalities of the code. After using CORT for a couple of years, Garner (2009) confirmed “the possibility of utilising a technology supported part-complete solution method, in the form of the CORT system, with students in introductory programming classes” (p. 308). Garner (2009) suggests development of mental models for students prior to the use of the CORT system.

#### *2.7.1.4 Use of Visualizing Techniques in Teaching*

Youssoof, Sapiyan and Kamaluddin (2006) proposed visualization as a technique to reduce cognitive load in programming. In visualising, the computer program run time behaviour is visually displayed on the screen. This framework integrates learning support and enables students to totally concentrate on the learning with no redundancy or split attention and as a result the WM could be fully utilised. This also enables the proper use of visual memory received from visual sensors. The visual metaphor can retain and process faster than the verbal metaphor. Therefore, better utilisation of WM was expected from the proposed visualised framework (Youssoof et al., 2006). The proposed framework consists of concept maps to visualise the various aspects of a concept and also relationships between various aspects. These concept maps are expected to help build schema structure in the LTM.

#### *2.7.1.5 Programming Development Environment Issues*

Programming development environments or interfaces have been designed for professional programmers. Such environments aren't suitable for novice programmers (Reis & Cartwright, 2004). Vogts, Calitz, and Greyling (2010) argue that novice programmers tend to focus more on getting syntax right rather than with pedagogical aims in mind when they use programming development environments. Despite the availability of suitable programming environments for students, some tertiary institutions tend to use a professional programming environment due to external pressure to have real world programming experience for the students (Vogts et al., 2010). A comprehensive study was conducted on this issue by three

researchers at the Nelson Mandela Metropolitan University on the perception of the two environments, academic performance and programming behaviour of novice programmers. The findings of this study include: motivation and self-belief of students are important and shouldn't be ignored: appropriate program development environments should be used in teaching (Vogts et al., 2010).

## **2.7.2 Past Research on Teaching Issues in Java Programming**

### *2.7.2.1 Difficult Concepts of the Java Language*

Garner, Haden and Robins (2005) studied and analysed problems that students encounter while studying Java as the first programming language at the University of Otago, New Zealand. This study was carried out keeping track of the questions asked by students during Java practical laboratory sessions conducted by instructors. The most frequently asked questions were on the use of arrays followed by data flow and headers. Garner et al. (2005) noted that the students had more issues associated with procedural or algorithmic aspects than with OO aspects. The problems associated with the procedural nature were on control flow and data flow. In OO, the main issue the students had was with constructors and when objects should be used in a program (Garner et al., 2005).

### *2.7.2.2 Pedagogical Issues*

Teachers at Agder College in Norway, used Simula and C++ computer language for teaching programming courses at the introductory level. Due to better features and advantages available in the Java language, the Agder College switched over to the Java for teaching the first programming language students in 1996. Hadjerrouit (1998) evaluated the suitability of the Java language, taking three years of experience using Java as the first language into consideration. This research revealed the importance of getting students' mind-set prepared in an object-oriented way from the beginning of the Java programming course. It was a difficult task for beginners until significant programming experience is gained. Hadjerrouit (1998) argues that many Java programming textbooks contains unnecessary illustrations of Java program development environment and some specific applications such as using applets, using pictures and sounds. Unfortunately, such books do not sufficiently cover topics, such as algorithmic thinking, structured programming, and object-

oriented design (Hadjerrouit, 1998). In this critical evaluation, Java was found to be a relatively difficult language for students with no programming background, and also a language suitable for teaching students with some programming knowledge. But the researcher, Hadjerrouit, had identified the fact that teaching Java was not just a problem of technology, but a pedagogical problem. Teaching and learning Java needs new ways of thinking and in more depth in order to grasp, and this is more challenging (Hadjerrouit, 1998).

### *2.7.2.3 Conceptual Issues*

With the popularity and the demand for the Java language at the end of the 20<sup>th</sup> century, there was some uncertainty about the suitability of using Java computer programming language to teach introductory level courses at Australian universities. There were some debates over this issue and three researchers, Clark, MacNish, and Royle from two leading Australian universities: the University of Canberra and the University of Western Australia embarked upon a research project centred on this issue. They explored the fact that OO languages such as Java impose an immediate conceptual load on students who are new to programming. They identified educational overheads inherent in Java programming language. The researchers also found a need to introduce the concept of class and object at early stages of the course. They also found the need for teachers to be explicit about the fact that the instances of the primitive data types contain values and the instances of objects contains the references to objects (Clark, MacNish, & Royle, 1998). Terms such as instance, encapsulation and hierarchy that are used to explain the concept of class and objects were found to be hard especially for many students for whom English was not their first language (Clark et al., 1998). Another difficulty in teaching the Java language is the need to expose students to exception handling too early. Finally, the researchers suggested object-oriented concepts such as classes, objects, creation of objects, and methods of a class are fundamentals and should be taught from the beginning of the introductory programming course. The teacher could use the advantage of corresponding software objects to real world objects when introducing to the concepts in the Java language (Clark et al., 1998).

#### *2.7.2.4 Use of Scaffolding*

Butler and Morgan (2007) investigated the academic problems faced by novice object-oriented programming students who use Java computer programming language at Monash University, Australia. About 150 novice programming students at Monash University participated in a survey which was aimed at finding the problems they faced in learning Java programming. The findings of the survey indicated that students were not receiving adequate feedback from lecturers on conceptually difficult issues such as OOP principles and efficient program design (Butler & Morgan, 2007). This course begins with basic programming paradigms and spans to high levels of conceptual complexity. The students had indicated Algorithms, Methods, OO concepts, and OO design as difficult areas to understand and implement. The researchers have suggested the need for using further surveys with the aim of finding out exactly why students find these conceptually difficult. It was also concluded that there was a need for scaffolding student learning in the difficult areas identified in this research (Butler & Morgan, 2007).

#### *2.7.2.5 Use of OO-Light Approach*

Lunney, McCullagh, and Lundy (2003) experimented using an approach called OO-light teaching approach to teach Java as the first programming language in graduate courses at the University of Ulster, UK. The students who enrolled in this course had already completed a primary degree in diverse disciplines such as science, engineering, arts, and management. The majority of the students were mature and highly motivated towards career progression. According to a survey conducted with the students who had enrolled in this course in 2002, 90% said that the programming course was the most difficult one of the graduate courses. Thus, Lunney et al. (2003) started teaching Java using applets and graphical components of Abstract Windowing Toolkit (AWT) and Swing components. Kawa, a graphical user interface tool, was used for program development in this course. The OO-light approach differs from the traditional approach in which a good foundation of Java concepts is given to students at the beginning of the course. The researchers involved in the OO-light approach realised the need for basic understanding about interactive objects to understand graphical interface programs using applets. Therefore, the applets were used in a parrot-like fashion, not going into deeper learning at the beginning of the

course. This was followed by a traditional approach to teach basic concepts until students grasp the concept of class. The researchers used non-object-oriented aspects of teaching basic concepts such as main method, using primitive variable, selection statements, repetitive statements and static methods and delayed teaching novel features such as arrays associated with loops until object-oriented features were introduced to the students (Lunney et al., 2003).

Collins (2002) experienced a similar problem of failing students in Java programming in undergraduate and postgraduate level degree programs at the University of Keele, UK. Despite the use of various approaches such as object first and object last in teaching Java, students were still not fully satisfied at the University of Keele (Collins, 2002). Collins (2002) believes that students require a clear understanding of the programming concepts of Java, before the development of programs. The OO-light approach could ease pedagogical shortcomings and initial overhead of the Java language. It also could motivate and appeal more to graduate level students as they begin learning using applets in web based applications (Lunney et al., 2003).

Object-oriented Programming (OOP) has become a dominant paradigm today due to its complexity and better organisation. Therefore, it is important to expose students at early stages to the OOP paradigm using computer languages such as Java (Georgantaki & Retalis, 2007). Grey and Miles (2002) consider the Java language as an ideal choice due to the fact that it is fully object-oriented and its wide range of libraries will enable students to use the knowledge in more specialised projects in the future. However, most students, including those who already had prior experience with procedural languages, found it difficult to understand even a simple Java program due to the need for understanding new concepts in an object-oriented paradigm at early stages (Grey & Miles, 2002).

Java programming was taught to first year students as a procedural language and later as an object-oriented language in the subsequent year at the University of Hull. This approach was changed in 1999 and the Java language was taught using object-oriented features at early stages with an interactive learning package on a CD\_ROM to support the course (Grey & Miles, 2002). Novice programming learners need to

understand the classes, static methods and return types of methods to write the simplest program. Input value from the keyboard is complex in the Java language, as it requires the knowledge of I/O streams and exception handling. Such expectation is beyond the comprehension of novice programming students at early stages (Grey & Miles, 2002). Grey and Miles (2002) realised that many students used such concepts blindly without clear understanding. According to Grey and Miles (2002), there are two ways of addressing this issue. One way is to ask students to ignore many important aspects of the Java language and the other is to hide some of the complex issues. The teaching paradigm used at Hull University was not to hide any of Java features, but to sequence teaching in such a way that it minimises the new concepts encountered at any one time. In addition, real world objects using pre-written classes were used in practical exercises. This course was interactive and partially completed programs were given to novice programming learners. In this course, learning from other people's codes was considered a useful learning technique. Story telling techniques were used to introduce the sequence of a number of themes and express issues directly related to the experience of the student (Grey & Miles, 2002). Such novel techniques can enrich the teaching material of a computer based course. Such material could incorporate audio/visual presentations. Although there was some interactivity in teaching, Grey and Miles (2002) included text and graphics and omitted audio and video as programming is not a visual activity.

#### *2.7.2.6 Use of Traditional Approach*

The fundamental topics to be taught in computer programming include variable types, methods, parameters, return types, local variables, and conditional statements. This list of topics was extended further in teaching Java programming with the topics such as private and public access specifiers, classes, objects, and state of objects. The challenge of teaching Java programming to beginners at Radford University, Virginia was teaching all of these topics at early stages of the course and also sequencing the content to make sense to the students (Barland, 2008). The `println ()` method could be used at early stages in programming to perform different calculations and display the result on console applications. This function-first approach was useful for learning passing arguments and returning values. It was also useful for them to be familiar with the syntax of the Java language. Students used private and public keywords without clearly understanding the meaning. As a result,

their first impression of programming became esoteric and unintuitive. With the experiences of teaching introductory level Java programming at Radford University, it was realised that proper sequencing of the topics reduces the unnecessary confusion and stress of learners (Barland, 2008). Barland (2008) found out that by reorganising the topics and minimising the emphasis on syntax of the Java language in the early stages, teaching could be improved. In addition, it is important to emphasize connections to high school algebra in Java programmes used at early stages of programming.

#### *2.7.2.7 Use of Functional Approach*

Bloch (2009) found students at beginner's level spending a lot of time, and struggling with the Java language. As a solution to this dilemma, Java was introduced using a simple limited concept called subset and then eventually guided them to advance to a more complex concept. In addition, the students were exposed to step by step design recipes for software development (Bloch, 2009). This process included concrete questions, and products enabling the students to know what they have achieved so far and the next step to be followed. According to Bloch (2009), students should be exposed to functional programming prior to imperative, sequential, or procedural programming. Such functional programming includes programs with simpler semantics or familiar models with algebraic expression evaluations (Bloch, 2009).

#### *2.7.2.8 Use of Online Approach*

According to Hadjerrouit (2007), programming is a difficult subject because it is a skill rather than a body of knowledge. There are a few online programming learning facilities and web-based programming tutors available today. Although, appropriate feedback is given to the online learner, most of such systems focus on technological application features rather than pedagogical aspects based on learning pedagogies. Hadjerrouit (2007) experimented teaching Java programming to novice students using blended approach. In this study, design-based research with a feedback loop was adopted to explore the possibility of improving students' ability to acquire basic Java programming concepts online (Hadjerrouit, 2007).



#### *2.7.2.9 Use of Mixed Approach*

Caspersen and Bennedsen (2007) proposed a teaching model based on three leaning theories to improve teaching programming at introductory level. The three theories used were: Cognitive Load Theory, Cognitive Apprenticeship, and Worked Examples. In addition, a pattern-based approach emphasizing program design and general problem-solving skills were applied to aid schema creation and improve learning (Caspersen & Bennedsen, 2007). Cognitive apprenticeship was applied to complex tasks with conceptual and factual knowledge.

#### *2.7.2.10 Use of Object First Approach*

The issue of teaching programming using object-first or imperative-first was addressed by the members of the Special Interest Group on Computer Science Education (SIGCSE) by email and the findings were published as a research output (Lister et al., 2006). In the object-first model, teachers focus on the principles of object-oriented programming and design with exposure to inheritance at the beginning of the course. In the latter part of the course, traditional control structures are taught within the OOP context. Some programming teachers believe in using a procedural paradigm first and then moving onto an object-oriented paradigm. They argue that fundamental knowledge of algorithms, structured programming, procedures, and historical development, is needed for students before introducing object-oriented programming (Lister et al., 2006). Burton, and Bruhn, (2003) support this idea and consider OOP as an extension to algorithmic thinking but accept OOP as a new paradigm. Culwin (1999) argues that most people who decide on undergraduate curriculum development have learned the procedural paradigm first and then moved on to the OOP paradigm, hence they have the conception that the old paradigm is a pre-requisite for the new paradigm. The programming-first approach could be painful for some learners who are expecting industry relevant teaching due to the wide usage of object-oriented programming in the industry.

#### *2.7.2.11 Use of Constructivist Learning Theory*

A number of researchers have suggested using Constructivist Learning Theory to overcome difficulties in learning Java concepts. (Hadjerrouit, 1999; Lui et al., 2004; Mead et al., 2006). These researchers argue that students are required to construct a

valid model in learning programming and that constructivist learning strategies are yet to be used in Java programming. According to Mead et al. (2006), we have to consider three basic components: curriculum, pedagogy, and assessment in teaching computer programming languages. According to past research documents, there has been much emphasis on the programming abilities of students. Some research documents consider the learning difficulties from cognitive scientists, learning theorists, and computer scientists point of views. Unfortunately, no attempt has been made on the impact of text books and curricular structures on learning (Mead et al., 2006). Mead et al. (2006) argue that novice students must construct a valid model of a computer in order to deal with the difficulties of learning programming. Moreover, proficiency in programming requires the acquisition of higher-order thinking skills, such as analysis, design, analogical thinking, reuse, evaluation, and reflection. Currently, however, few educators systematically apply constructivism to computer science (Berglund, Daniels & Pears, 2006), and constructivist learning strategies are only beginning to emerge.

#### *2.7.2.12 Use of Bloom's Taxonomy and Objects-First approach*

Machanick (2007) researched the use of the object-first approach to teach Java programming language adopting the Bloom's Taxonomy to design the course delivery at the University of Queensland, Australia. In this research, factual contents were introduced to introductory level students at the beginning and subsequently the higher level cognitive skills and design skills were taught. Bloom's taxonomy enabled classification of concepts to be taught and skills to be acquired. It helped to decide on suitable tasks at a given skill level. Bloom's Taxonomy was applied as a basis for sequencing the teaching concepts to be taught in the course (Machanick, 2007). Despite the strong motivation students reported in the course appraisal, researchers were unable to find sufficient evidence to come to a strong conclusion about the suitability of an Object-First approach in teaching the Java language (Machanick, 2007).

## **2.8 Java Development Environments**

There is a wide range of Java development environments available. Some of these are professional tools to be used for industrial applications developments whilst others are developed and designed to be used in teaching environments. The use of

educational tools could be very useful in overcoming students' difficulties and also in achieving teaching and learning objectives (Brusilovsky, et al., 1997). Contrary to the above argument is that some teaching tools which are too simple, too complicated, or inappropriate could not serve the purpose and may cause problems for students. Some students could get confused in professional program development environments containing huge set of interface components and functionalities (Kölling, Quig, Patterson & Rosenberg, 2003). Some Graphical User Interface (GUI) environments which are used for OO programming development could give a distorted picture of programming concepts to the learners. According to Kölling et al. (2003), students who use Graphical User Interface (GUI) development environment, concentrate more on visual aspects rather than programming concepts. Kölling et al. (2003) describe such environments as traps. Kölling et al. (2003) describe such environments as traps. GUI environments are suitable for program builders, not for the learners. Therefore, it is important to understand the distinction between teaching tools and professional tools in Java programming (Georgantaki & Retalis, 2007). As Georgantaki and Retalis (2007) suggested, it is appropriate to teach Java programming to novices using Java teaching tools and then introduce professional tools to students at a later stage.

### **2.8.1 RAPTOR**

RAPTOR is a visual programming design tool which generates the C++ and Java code to a certain extent. It was developed by the department of computer science at the US Air Force Academy (Welcome to the RAPTOR home page, 2011). It is freely distributed to the computer science education community. Carlisle (2009) used this tool to teach Java programming language and presented the outcomes in a research paper. Carlisle (2009) found that RAPTOR reduces the time student spend on dealing with complex syntaxes and also helps them to visualise classes. According to the findings of a survey conducted by Fowler, Allen, Armarego and Mackenzie (2000), 70% – 83% of the students were visual learners. Quoting this finding, Carlisle (2009) argues that the “textual nature of most programming environments works against the learning style of the majority of the students” (p. 276). UML designer allows the user to design classes graphically including comments, specifying Java access modifiers. It is also possible to specify inheritance, associations, nesting, aggregation, and dependency. Interface

implementation is possible in RAPTOR. The RAPTOR class editor is useful for students to create Java code for Instance variables, Constructors, and Methods using GUI interface without much effort (Carlisle, 2009). Carlisle (2009) found this tool a simple environment in which to experiment with OO programming and useful for visualising complex concepts such as recursion, heaps, and stacks.

### **2.8.2 BlueJ**

BlueJ is a visual Integrated Development Environment (IDE) for the Java language. It was designed and implemented by Kölling and Rosenberg (1996) to improve the teaching and learning of introductory programming using object-oriented style in the Java language. Despite its limited features, it is useful software for beginners in Java programming. The features of BlueJ include graphical representation classes and objects, simplicity, and inspect features to see the values of the properties of an object. BlueJ provides software project structure graphically in UML like diagrams (Kölling & Rosenberg, 1996). According to Kouznetsova (2007), BlueJ helps beginners to grasp difficult Java programming concepts easily. BlueJ also helps students to generate and edit Java code and enables students to use graphical images without prior knowledge of Java graphics in Swing libraries (Kouznetsova, 2007).

The students at Sam Houston State University had difficulties in grasping object-oriented concepts in Java programming. The Bluej was experimented as a teaching tool to help students understand Java concepts better in game development applications. This experiment revealed that BlueJ was very helpful and it increased their level of engagement in programming. (Kouznetsova, 2007). Assignments with incorporated graphics are crucial in teaching Java programming language as graphical representation enhances student engagement. Students can easily create graphics and incorporate them into applications using BlueJ (Kouznetsova, 2007; Van Haaster & Hagan, 2004).

BlueJ facilitates the object-first teaching approach in which students are able to experience the interaction with objects before being confronted with the Java concepts and syntax (Kölling & Rosenberg, 1996). Hagan and Markham (2000) published a research paper on their experiences using BlueJ to teach introductory level programming using the Java language. The analysis of the data collected from

students on their backgrounds, perceptions and attitudes towards BlueJ was found to be very positive, and towards the end of the course, from the answers students had given to examination questions and in assignment interviews, the researchers were convinced that they had a good grasp of object-oriented concepts (Hagan & Markham, 2000).

### **2.8.3 Kawa**

Kawa is a software tool that can be used for managing and creating Java programs. Kawa provides an integrated development environment (IDE) which consists of an editor, compiler, window for library browsing and output window (Introduction to Kawa 3.13, n.d). Kawa was adopted at the University of Ulster in the UK to teach Java programming at an introductory level (Lunney et al., 2003). Lunney et al. (2003) consider Kawa to be the best IDE environment to teach a first programming language. The major concern with Kawa is that it has no new developments.

### **2.8.4 Dr. Java**

Dr. Java is an open source Integrated Development Environment (IDE) for Java programming language. It runs on multiple platforms such as Windows, Linux, and Macintosh (Olan, 2004). It was developed at Rice University in the USA. Dr. Java is a pedagogic programming environment with a simple interface which enables students to focus on programming rather than spending time on learning the programming environment (Allen, et al., 2002). Dr. Java does not provide graphical representation of classes and objects as in BlueJ (Georgantaki & Retalis, 2007). Dr. Java provides an interactive pane with a feature named Read-Eval-Print-Loop (REPL). This feature allows the evaluation of Java expressions and statements without running the whole program and as a result facilitates incremental program development. REPL is useful for introductory level Java programming learners to test methods with parameters and return values. Dr. Java is provided with a tool for generating Javadoc documentation for classes.

### **2.8.5 Eclipse**

Eclipse is a multi-language open source software development environment (IDE) which could be used for developing Java programs (Moyer, 2010). Among the other open source Java programming environments, Eclipse is considered a professional

program development environment (Chen & Marx, 2005). Chen and Marx (2005) argue that most Java IDEs have been designed for pedagogical purposes and fail to expose students to real world environments. One attractive feature of Eclipse is that it has a very simple editor and students do not need to spend more time with instructors to be familiar with the editor. The advantage of using Eclipse is that its IDE environment provides professional industry level experience for the students. The wizards in Eclipse save time for the users as they generate codes for classes, methods, and constructors (Chen & Marx, 2005). The facility to generate Javadoc standard documents is also an attractive feature in Eclipse (Olan, 2004). There are a number of plug-ins available for Eclipse. For example JScoper is a plug-in using a graphical call graph browser that can be used to convert Java code to Real-Time Specification for Java (RTSJ) (Ferrari, Garbervetsky, Braberman, Listingart, & Yovine, 2005). Other useful plug-ins include LaTeX, CVS, and UML diagrams (Moyer, 2010). The cheat sheet functionality of Eclipse enables users to create tutorials which are interactive with Eclipse's User Interface to support features. However, extensive effort and specific skills are required to create cheat sheets (Ying, Gang, Nuyun & Hong, 2009).

### **2.8.6 Visual J#**

Visual Studio .NET provides a multi-language environment which supports Visual Basic .NET, Visual C++ .NET, Visual C# .NET, and Visual J# .NET. Mixed language solutions can be developed in Visual Studio by the Microsoft.NET environment. Unlike most of the other Java applications, J# programs do not run on a Java Virtual Machine. J# programs run only on .NET Framework (Introducing Visual Studio .NET, 2011). Due to the declining usage of J#, on January 10, 2007 Microsoft announced the retirement of J# and that J# language and JLCA tool will not be available in future versions of Visual Studio (Product Announcement, 2011). This is a drawback for Microsoft which pushed big telecoms and financial institutions from Java to .NET (Cantù, 2007).

### **2.8.7 Borland JBuilder**

JBuilder is a full-fledged IDE environment for Java programming language (Kouznetsova, 2007). It is a professional IDE with Facilities for developing, testing, debugging and deploying J2ME applications (Utting, 2006). It was developed by the

Borland Software Corporation and sold to Embarcadero Technologies in 2008. Embarcadero Technologies released the latest version of JBuilder 2008 in 2008 (Embarcadero customer support, 2010). Some research articles using JBuilder as a development tool have been published. In one of the students' projects, JBuilder was used on Extreme Programming (XP) as a tool at Brighton University. Two other tools: Castor for object relational mapping and MySQL were used along with JBuilder. The students found the tools quite complex and the debugging Java Servlets and Java Server Pages (JSP) very slow even on fast computers (Lappo, 2002). In another research application which was aimed at students exploring cognitive difficulties in maintaining an unfamiliar object-oriented system, JBuilder was chosen (Karahasanović & Thomas, 2007). In a survey conducted by Haaster and Hagan (2004), students had commented that JBuilder was quite complex and not suitable for novices in OO programming.

## **2.9 Summary**

The literature review underpinning the research described in this thesis includes a detailed coverage of most of the popular instructional design methods and learning theories. The report began by collecting details of various learning models, such as behavioural, constructivist, experiential and cognitive. Learning models such as the Kolb Experiential Learning Model, Bloom's Taxonomy, the Felder-Silverman Learning Style Model and the R2D2 Learning Model were discussed in detail. As more recent research studies were explored, it was found that research on teaching and learning had been focused on cognitive learning models. In particular, the emphasis has been on managing limited WM by carefully manipulating germane, extraneous, and intrinsic cognitive loads. The use of worked examples in teaching programming have been successfully used in reducing extraneous cognitive load in past research applications (Caspersen & Bennedsen, 2007; Garner, 2002, 2007, 2009; Koedinger & Aleven, 2007; Van Gog et al., 2010; Wittwer & Renkl, 2010). Garner (2009) used worked examples in Visual Basic programming language using a teaching tool called CORT. Garner (2009) has suggested the development of mental models prior to the use of worked examples as a future research prospect. Wittwer and Renkl (2010) found worked examples to be useful for understanding conceptual knowledge. These findings suggested future research areas which are yet to be experimented with in teaching Java programming for beginners. One such future

research prospect was identified to study the possibility of using worked examples to impart the conceptual knowledge of Java programming concepts at a beginner's level using consistent mental modeling techniques.

Despite the long history of using mental modeling to retain scriptural interpretation and historical traditions (Johnson-Laird et al., 1998), past research proved them to be useful in conceptually representing abstract concepts (Wu et al., 1998). A number of researchers have emphasised the importance of using visual images, pictures, or visualisation tools to improve teaching (Ben-Ari, 2001b; Carlson, 2007; Johnson-Laird et al., 1998; Ma et al., 2007; Roper & Wood, 2007; Van Haaster & Hagan, 2004; Wu et al., 1998). A survey conducted by Ma et al. (2007) found the need to help students to develop viable mental models in order to understand reference variables at the very beginning. The issue of the need to synchronise the mental models of the teacher and the students or having a shared mental model of a concept has been highlighted by some researchers (Ben-Ari, 2001b; Norman, 1990). This literature opened up new research opportunities to experiment with the use of shared mental modeling techniques in explaining Java programming concepts. Past research also suggests the use of visualisation tools, pictures and images for mental modeling.

Hemispheric Dominance Theory, widely known as the right brain-left brain theory, was suggested by Sperry in 1960. This theory divides major intellectual functions into the left and the right hemispheres of the brain. Recent neurological research has identified the regions of the brain where different functionalities take place (Smith, 2000; Smith & Jonides, 1999). These findings led to an opportunity for this research to discover if there is a correlation between brain dominance and the ability to understand Java programming.

Instructional techniques and visual tools were studied in detail in this literature survey. It was found that although mind mapping is considered a powerful visual and collaborative teaching tool, it has never been used in teaching Java programming concepts. Therefore, this is another area that could be investigated.

Some Integrated Development Environments (IDEs) have been designed for professional Java programmers while others have been developed for Java programming language learners. The use of appropriate program IDE for teaching



Java programming at a beginner's level has been a debatable issue among researchers (Vogts et al., 2010). The Dr. Java IDE was used due for two main reasons. First, this course belonged to UNITEC Institute of Technology, Auckland. It was delivered at the Waiariki Institute of Technology under the guidance of UNITEC and Dr. Java matched the UNITEC prescription. The second reason was the simplicity of the IDE environment of Dr. Java for beginners in Java language.

This chapter has demonstrated that considerable research has been completed and now this research can build on the theoretical framework provided by these studies. At the same time, this research makes an original and unique contribution to the efficient teaching and learning of computer programming for beginners using JAVA.

## **CHAPTER 3**

### **METHODOLOGY**

#### **3.1 Introduction**

This chapter provides details of the methodology used to identify problem areas in the teaching of programming using the Java language at the introductory level and how instructional problems were addressed in an attempt to improve this teaching. The initial task was to find an acceptable instructional design model to meet the expectations of the research. The popular Analyze, Design, Develop, Implement, and Evaluate model (ADDIE) was chosen to provide guidelines for the instructional design process. According to the ADDIE instructional design model, it is required to identify instructional problems, instructional goals and objectives, delivery options, pedagogical issues, learning environment and learners' existing knowledge and skills (Culatta, 2011).

At an early stage in this research, it was necessary to identify instructional problems and delivery options of the Java language for beginners. Therefore, in the first phase of this research, a questionnaire was used to determine from the students who had completed this course, what they considered to be the difficult areas and concepts in learning Java programming language. Furthermore, they were asked to consider suitable teaching styles for each concept.

In addition, an extensive literature survey was carried out which provided ample information about instructional problems related to teaching and learning Java programming and also object-oriented programming. According to the findings of the literature survey, researchers have identified instructional problems as pedagogical (Hadjerrouit, 1998), cognitive (Garner, 2002; Yousoof et al., 2006) and conceptual (Clark et al., 1998). In particular, Yousoof et al. (2006) has proposed a model for using visualization techniques to deal with cognitive aspects of learning programming. The need for detailed mental models as program chunks or structures was identified by Winslow (1996) nearly two decades ago.

Recent developments on working memory models such as Baddeley's have led to the formulation of Dual Code Theory (DCT). According to DCT, visual and verbal information are handled by different channels of the brain (Paivio, 2006). As suggested by Paivio (2006), mental modeling using imagery and pictures were adopted for teaching in phase two of the research described in this thesis.

Garner (2009) experimented with a cognitive tool called CORT in which worked examples were used based on the principles of the Cognitive Load Theory in teaching programming. Many other researchers have identified teaching programming as a formal operational level cognitive development (Ven de Ven & Govers, 2007). It was interesting to find that so many different approaches had been tested by teachers in different parts of the world when teaching this language. Such approaches include using the Object First Approach, Bloom's Taxonomy, Constructivist Learning Theory, OO-Light Approach, Scaffolding, Traditional Approach, Functional Approach, Online Approach, and a Mixed Approach. It is noteworthy that approaches such as the Traditional Approach and the Object First Approach were found to be opposed to each other.

Taking the findings of the literature survey and the data collected from the students in the first phase of this research into consideration, instructional goals and objectives were determined. It was decided to use the concept first approach as the Java language has been developed on the principles of object-oriented concepts. The teaching materials were changed to enable the introduction of mental modeling with the goal of giving students a thorough knowledge of programming concepts. In addition, recent findings in cognitive load theory, which are related to management of cognitive loads, were applied in the preparation of teaching materials and examples. After teaching the concepts with the new teaching tools and techniques, the success of teaching each concept was evaluated by the students using a Mini-questionnaire during the second phase.

Suitable delivery options and pedagogical issues were also to be decided for this course. As proposed by Clark et al. (1998), it was decided to introduce the concept of class and objects and the manipulation of objects at the beginning of the course. The development of Cognitive Load Theory and its applications to the efficient usage of working memory is a recent advance in pedagogical research in teaching. These

research projects have given particular emphasis to the use of worked examples to reduce unnecessary cognitive loads imposed by conventional instructional materials (Van Gog et al., 2010). The partially completed assignments were used to build the schemas with new knowledge using the prior knowledge available on schemas in the long term memory of the learners (Garner, 2002). The element interactivity contributes to both extraneous and intrinsic cognitive load (Beckmann, 2010). Therefore, the topics were delivered using the teaching materials produced with reduced intrinsic and extraneous cognitive loads. The germane cognitive load for individual students was to be kept at a minimal level by using scaffolding in a collaborative manner.

The other teaching tools used include FreeMind 0.8.0, Dr. Java, and BlueJ. The techniques used include graphics for mental modeling, collaborative learning and scaffolding. The tools and techniques used to teach identified concepts were based on the principles of Cognitive Load Theory with the aim of managing the learner's working memory effectively and also using the concept of mental modeling to help with the understanding of programming concepts. In addition, Hemispheric Dominance Theory (HDT) and its relevance to their learning styles were investigated in this research.

### **3.2 Research Focus and Significance of the Study**

The research project examines the teaching tools and techniques that could be used to effectively teach Java programming to beginners. Any teaching tool or technique is usually based on a theory or hypothesis. Therefore, it was necessary to apply, and experiment with, new instructional design models and learning theories in classes. Consequently, the emphasis in this research was placed more on cognitive learning concepts due to evidence of its proven applications in teaching (Clark et al., 2006).

The literature study revealed that the methodology to be used in teaching Java was still a debatable issue among programming teachers and researchers (Kannangara, 2007). The object-first approach (Lister et al., 2006) was adopted as a teaching methodology with more emphasis placed on the Java concepts, and classes and objects were introduced at the early stages of teaching. A selected set of low cost teaching tools was used to impart the knowledge of object-oriented programming

principles and concepts. The teaching tools and techniques used were based on the principles of Cognitive Load Theory (CLT) combined with mental modeling and scaffolding. Hence, the outcome of this project should reveal the effect of using a combination of the principles of CLT and mental modeling in teaching object-oriented programming principles using the Java language. The findings of this research will benefit not only teachers but also learners due to the low cost teaching tools used in teaching difficult Java programming concepts.

### **3.3 Research Questions**

The following research questions were addressed in the Phase One of this research:

1. What are the difficult Java concepts for the learners and the most suitable teaching styles for teaching such concepts?
2. Is there any correlation between logical and artistic hemispheric dominance factors of students and the difficulty levels of Java concepts?
3. What are the preferred of learning styles of learners and the combinations of learning styles?

#### **3.3.1 Research Questions in Phase One**

The main aim of this phase was to identify the difficult areas and concepts of the Java programming language for the students. A questionnaire was used to collect data from students who had just finished the introductory level Java programming course. This course was designed and developed at the UNITEC Institute of Technology, Auckland in New Zealand. It was delivered at the Waiariki Institute of Technology and Bay of Plenty Polytechnic under the license of Bachelor of Computer Systems (BCS) programme at the UNITEC Institute of Technology. The questionnaire was web based and made available on the internet to the participants in the three institutions. With the researcher's past experience in teaching the Java language, areas and concepts which were considered to be important were listed in the questionnaire (see Figure 3.1). Each concept had five difficulty levels listed on radio buttons as "too difficult", "very difficult", "difficult", "not difficult", and "very easy". One option of the difficulty levels for each concept was chosen by the participant. There was sufficient space available on the questionnaire for the students

to add any other difficult areas or concepts which were not listed and could be important in the Java language. They were also asked to indicate the difficulty level.

8) Please select one of the difficulty levels for each of the concepts or areas listed below with the experience you had while learning Java language at the introductory level.

	1 too difficult	2 very difficult	3 difficult	4 not difficult	5 very easy
Variable types (int, char, & double etc.)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Variable categories (local, parameter & instance)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Using conditional statements (if ..then ..else)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Using repetitive statements (loops)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Understanding of the concept of classes and objects	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Creation of an object using a Class	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Creation of a template for a class	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Use of parameter variables (arguments) in a method	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Returning a value from a method	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Testing and debugging	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Preparation of test data for a program	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Using arrays for primitive data types	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Using arrays for objects of a class	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
String manipulation in Java (using methods in String class)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Using text files for input and output	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Logic depiction methods (structure diagrams etc.)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Inheritance and polymorphism	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

*Figure 3.1.* Listed areas and concepts in the Java language.

The meanings of the learning styles; auditory, kinesthetic, and visual were included in the questionnaire to make them clear to the participants (see Figure 3.2). Another question related to the above was to suggest the most suitable learning style for each concept or area listed on the survey. The three learning style options given were auditory, kinesthetic, and visual (see Figure 3.3). A participant could select one of the three options on the radio buttons and could also indicate their second choice of learning style in the space provided on the text box (see Figure 3.3).

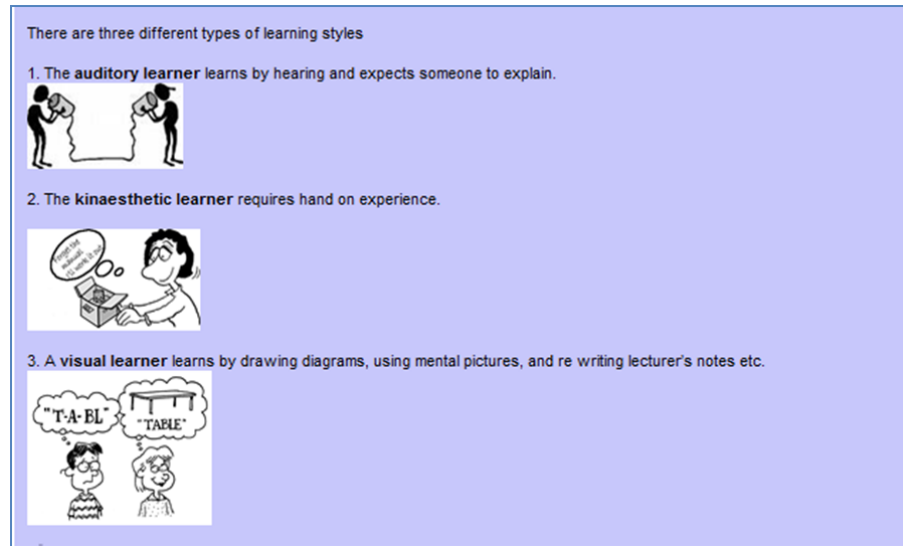


Figure 3.2. Learning styles explanation.

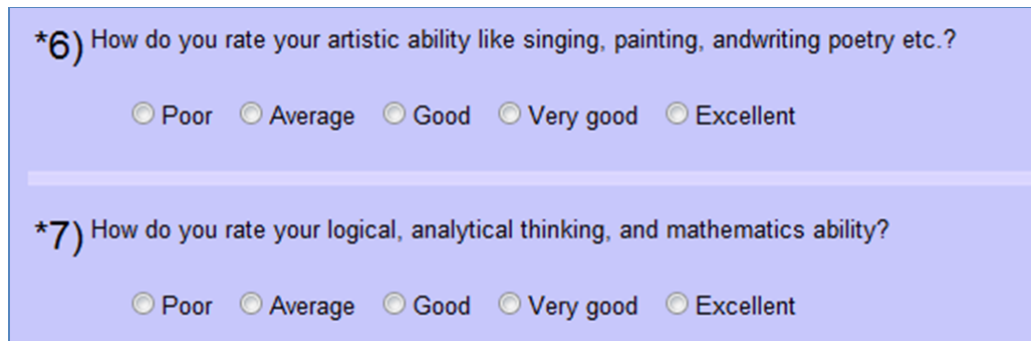
12) Which one of the following teaching styles would be the most suitable to teach each of the following areas? Please indicate your second choice (if any) on the text box.

	Auditory	Kinaesthetic	Visual	Your Second Choice
Variable types (int, char, & double etc.)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="text"/>
Variable categories (local, parameter & instance)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="text"/>
Using conditional statements (if ..then ..else)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="text"/>
Using repetitive statements (loops)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="text"/>
Understanding of the concept of classes and objects	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="text"/>
Creation of an object using a Class	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="text"/>
Creation of a template for a class	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="text"/>
Use of parameter variables (arguments) in a method	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="text"/>
Returning a value from a method	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="text"/>
Testing and debugging	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="text"/>
Preparation of test data for a program	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="text"/>
Using arrays for primitive data types	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="text"/>
Using arrays for objects of a class	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="text"/>
String manipulation in Java (using methods in String class)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="text"/>
Using text files for input and output	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="text"/>
Logic depiction methods (structure diagrams etc.)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="text"/>
Inheritance and polymorphism	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="text"/>

Figure 3.3. Learning style options for concepts.

A question was included to indicate the participant's judgment on his/her artistic ability such as singing, painting, and writing poetry. A similar question was used to indicate his/her logical, analytical thinking, and mathematics ability. Each question had five options: Poor, Average, Good, Very good, and Excellent with five radio buttons (see Figure 3.4). A participant could select only one option in each question.

These two questions were included to investigate the correlation between students' total difficulty level and the logical and artistic brain dominance profile.



\*6) How do you rate your artistic ability like singing, painting, and writing poetry etc.?

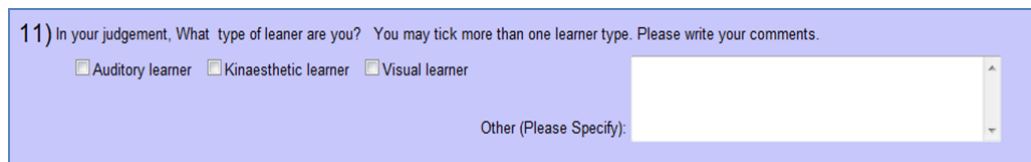
☐ Poor ☐ Average ☐ Good ☐ Very good ☐ Excellent

\*7) How do you rate your logical, analytical thinking, and mathematics ability?

☐ Poor ☐ Average ☐ Good ☐ Very good ☐ Excellent

*Figure 3.4. Questions on artistic and logical abilities.*

A question, What type of learner are you?, was included in the questionnaire with three options: Auditory learner, Kinaesthetic learner, and Visual learner using check boxes (see Figure 3.5). The participants could tick more than one option for this question. This question was used to explore the distribution of students by preferred leaning types and to analyse the combinations of learning types of the data sample chosen.



11) In your judgement, What type of learner are you? You may tick more than one learner type. Please write your comments.

☐ Auditory learner ☐ Kinaesthetic learner ☐ Visual learner

Other (Please Specify):

*Figure 3.5. Question on type of learner.*

In addition, questions related to demographics were included to collect information on students' gender, age, work experience, and their highest academic qualification. It was hoped to discover some useful information on the different types of learners.



### **3.3.2 Research Questions in Phase Two**

The following research questions were addressed in Phase Two of this research:

4. Can the teaching tools based on a combination of Cognitive Load Theory (CLT), the concepts of Mental Modeling and scaffolding be effective in teaching difficult concepts in the Java language?
5. Can there be a relationship between students' learning preference and their logical and artistic hemispheric dominance?

### **3.4 Mind Mapping as a Teaching Tool**

The creation of a template for a class is a fundamental concept in Java programming language to be understood by students but according to the survey given in Phase One, 27% of the student population found this concept difficult to understand. Since Dr. Java is not a visual programming IDE, the functionality of the components of the class template were taught using textual programming code. Therefore, mind mapping was introduced as a tool along with the class diagram and textual code in phase two of the research. The aim of this was to explore the possibility of using mind mapping as a teaching tool for this concept. Mind mapping was chosen as it is renowned as an excellent visual tool in collaborative learning (Willis & Miertschin, 2006). Also, mind mapping is a low cost tool. As suggested by Koznov and Pliskin (2008), use of a combination of both graphical and textual components could help learning. There are many free mind mapping software tools available today. The FreeMind 0.8.0 mind mapping is one of them and was used in this research (FreeMind – free mind mapping software, 2011). The students had the option of using hand drawn mind maps or the software.

The students were asked to produce a mind map diagram for a given class template. Three main branches: Instance variables, Constructors, and Methods were drawn in three different colours (see Figure 3.6). Each instance variable and the variable type were listed at the second level branches under instance variables (see Figure 3.6). The Methods were divided into two main branches to elaborate the two categories: Accessor and Mutator. The Accessor Methods were listed at the third level branches under Accessor branch and Mutator Methods were listed at the third level branches under Mutator branch. All the Constructors: default; and alternatives; were placed in a separate branch and each branch represented a Constructor.

The mind map was adapted as a more elaborate version of the class diagram (see Figure 3.6). A teaching activity using mind mapping was used to explain the concept of the class template in the second and third weeks of the course. The lesson was introduced using a worked example. The worked example was a complete Java code of a class and consisted of a couple of instance variables, constructors, and methods (see Appendix A).

As the first part of the second phase of the research, mind mapping was experimented with as a tool to explain the concept of the class template. After having taught students this concept using a worked example and a mind map, the Mini-questionnaire -1 (see Appendix G) was used to get students' comments about the effectiveness of using mind mapping and the worked example to teach this concept.

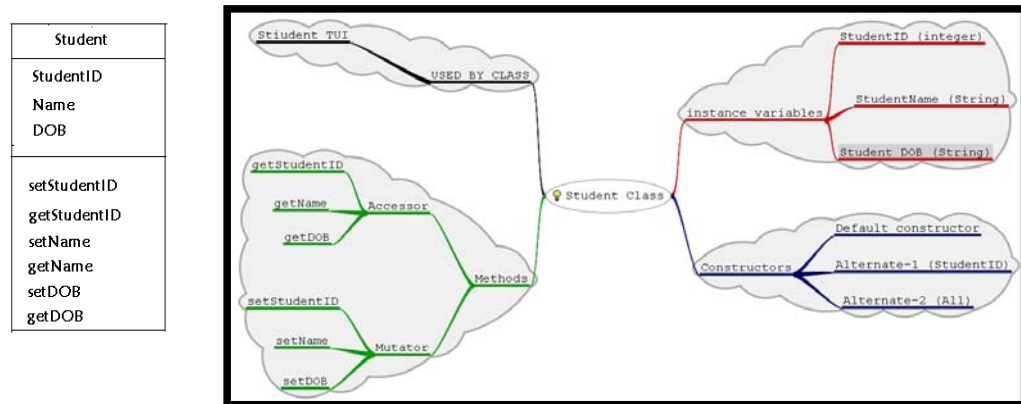


Figure 3.6. Class diagram and mind map of the student class.

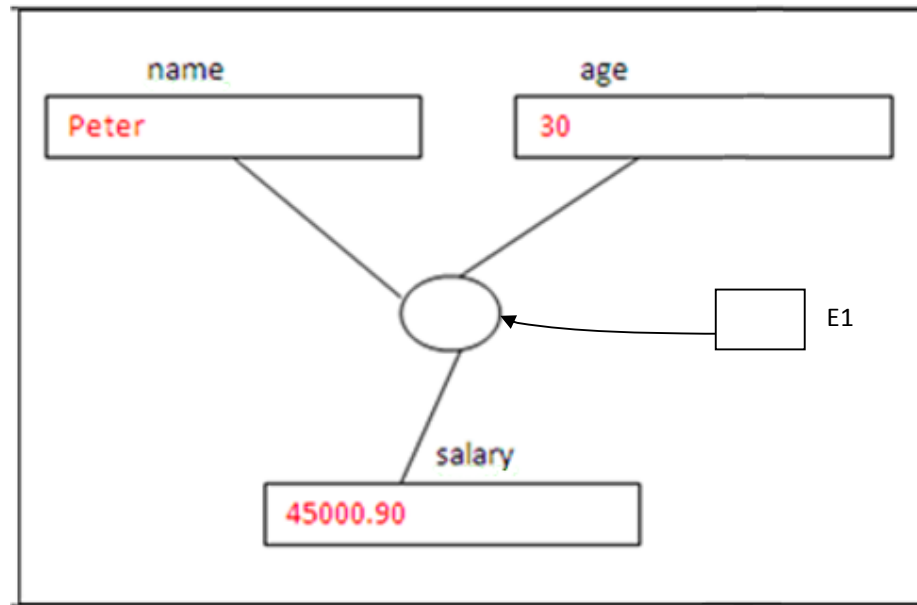
### 3.5 Use of Mental Modeling to teach Java Concepts

Mental modeling is based upon social constructivism. It was historically used to represent an external process internally in the human brain in terms of words, numbers and figures (Johnson-Laird, 1983). It can be psychological representations of real, hypothetical, or imaginary situations (Johnson-Laird et al., 1998). Recent research suggests visualization as a way of understanding object oriented programming concepts (Van Haaster & Hagan, 2004). According to Carlson (2007), knowledge is stored in the brain as internal constructs or images. These mental models activate when we retrieve or use such knowledge. Some researchers have experimented with visual software development environments such as Alice to teach

object oriented concepts (Dann et al., 2001). But, Pears et al. (2007) argue that in such software environments, students tend to visualize dynamically and do not understand the deeper meaning or context of the concept. There are situations where the teacher's mental model and the student's mental model do not synchronize (Ben-Ari, 2001b; Norman, 1990). Therefore teachers should help students to build mental models while teaching concepts (Ma et al., 2007; Wu et al., 1998).

In the second phase of this research, the possibility of using images to create mental models to enhance teaching Java programming at beginner's level was investigated. Some of the programming concepts were taught using a carefully designed teaching activity (see Figure 4.5) with the intention of helping students to create mental models and reduce cognitive loads. The success of this was tested using a Mini-questionnaire at the end of the course.

A visual representation of an object was designed with the aim of creating a mental model to help in the understanding of a number of concepts. Such concepts include creation of an object using a class, variable categories, and arrays of objects. A set of images was used consistently throughout the course so that students became familiar with and gave meaning to images. For example, one of the images used to visualize methods was the stick man. The Star Structure (see Figure 3.7) was used to visualize objects and the variables were represented using a rectangle or box. The state change of objects was explained visualizing the change with the value assigned to properties.



*Figure 3.7. Star structure.*

Some students had commented about the difficulty in understanding the concept of reference variables in Java programming language in the questionnaire used in Phase One of this research. In Figure 3.7, E1 is shown as a rectangle as it is a reference variable. The instance variables are also shown as rectangles. The object on Figure 3.7 has no static name and it has to be accessed through the reference variable E1. It is shown using an arrow.

A worked example with a class template was used to create an object on the computer's Random Access Memory (RAM) and also to manipulate the object using Mutator and Accessor Methods of the class. Many researchers have used worked examples to help learners understand difficult concepts. They have also discovered that such examples help to reduce both intrinsic and extraneous cognitive loads (Garner, 2002; Van Gog et al., 2010) and support creation of new schema (Caspersen & Bennedsen, 2007).

Figure 3.7 is an example of a Star Structure which was used to illustrate the creation and manipulation of an object using an Employee Class Template (see Appendix B) with three instance variables: name; age; and salary. Many semantics have been embedded into this image. The embedded semantics including the fact that instance variables are containers of data and an object is comprised of many instance

variables. The data are shown in red colour on the Star Structure. The Star Structure of the object in Figure 3.7 has three instance variables which are shown as rectangles. When the Java code is taught to create an object, this figure could help students to imagine the state of the object visually with no values in instance variable rectangles.

The Java code given below creates a reference variable E1 which contains the reference to the object created by the default constructor.

```
Employee E1 = new Employee();
```

A relevant picture similar to Figure 3.7 is usually shown to students until they establish a mental model to visualize the concept. When teaching the use of a mutator method to assign a value to an instance variable of an object command and the state of the object on the image are shown to students with the value added to the box of that instance variable in red colour. The following is an example of a mutator method used to add a value to an instance variable of an object.

```
E1.setName("Peter");
```

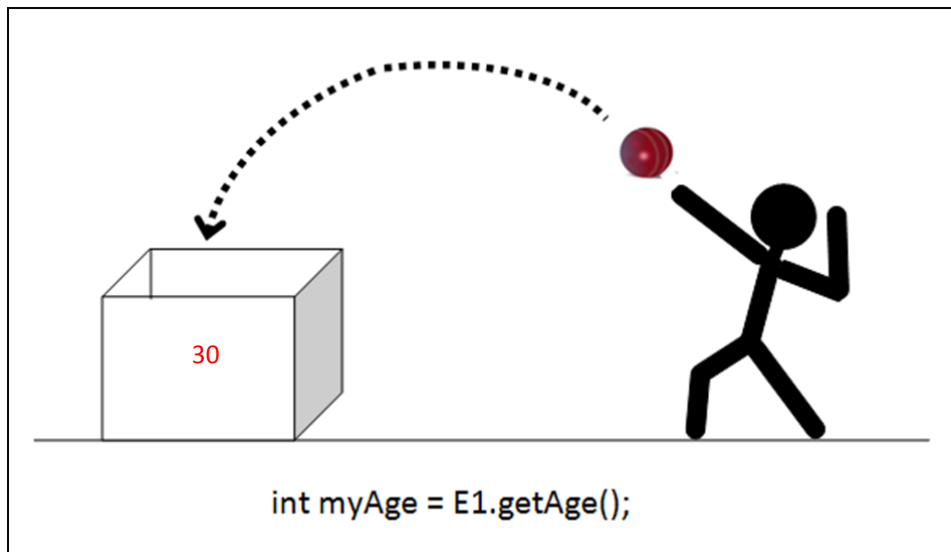
In the second part of the Phase Two, Star Structure was used to visualize object components, and the state of an object after each operation, with the aim of enhancing teaching. This visual structure was used throughout the course. The Mini-Questionnaire-2 (see Appendix D) was used to get students' comments about the effectiveness of using Star Structure and the worked examples to teach the creation and manipulation of objects.

The use of an accessor method on an object to retrieve data from an object is more complicated than the use of a mutator method due to its return type and the value. According to the results of the survey conducted in phase one, 15% of students had difficulties in understanding this concept. Therefore, Figure 3.8 was introduced to help students understand the concept of returning a value using a picture of a stick man returning a ball. This image was used while explaining the Java code that retrieves the value of an instance variable of an object using a method that returns a

value. The Java code shown below is an example of an accessor method which returns a value. The returned value is assigned to myAge variable.

```
int myAge = E1.getAge();
```

Figure 3.8 was used as a tool for mental modeling this concept. The stick man represents a method and the box represents the variable. This image was created with the intention of helping schema building for the concept of returning a value. As per Garner's (2002) findings, the relevant existing schemas in the long term memory are used in creating new schemas in the working memory and the new knowledge is stored as a new schema in the long term memory. The learner's existing schema on a human throwing a ball into a box is used to create a new schema with the concept of a Java method returning a value and the value returned is assigned a variable.



*Figure 3.8. Method returning a value.*

The survey results in Phase One, indicated that using parameters in a method was found to be difficult for about 23% of the student population. The image in Figure 3.10 was used with the aim of creating a mental model using an analogy to make this concept easier to understand. This image was used while teaching this concept of passing parameters and returning a value back to the calling program using Java code (see Figure 3.9). The main method was run a couple of times with different sets of values to show the reusability of the getGrade method.

```

Public void main()
{
    char answer = getGrade (50000.0);
}

public char getGrade(double salary)
{
    char ch=' ';
    if (salary >= 60000)
        ch= 'A';
    else if ((salary < 60000) && salary > 40000)
        ch = 'B';
    else if ((salary <=40000) && salary > 0)
        ch = 'C';
    return ch; }

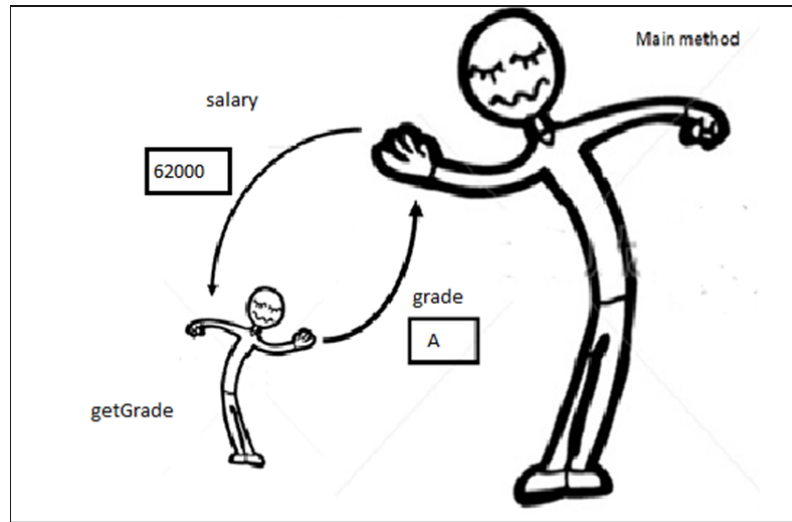
```

*Figure 3.9. Java code of main method and getGrade method.*

In Figure 3.10, the getGrade method has one parameter. This parameter is used to pass a salary value into the getGrade method. The getGrade method decides the grade for the salary and returns the appropriate grade back to the main method. In Figure 3.10, the two methods, getGrade and main are visualized as two stick men. Figure 3.10 provides an analogy symbolizing two methods as two stick men, one assigning work to the other. In this example, getGrade method can be reused. This analogy describes the functionality of the getGrade method. In designing this teaching tool, the cognitive aspects of building new schema using existing schemas in the long term memory was applied. It was assumed that most students are familiar with this analogy used in the real world and so it could be used to build knowledge related to the Java method as a new schema using the existing schema (Driscoll, 2000; Mead et al., 2006).

The parameter variables and the local variable to the main method are displayed as rectangles, with the passing and returning values.

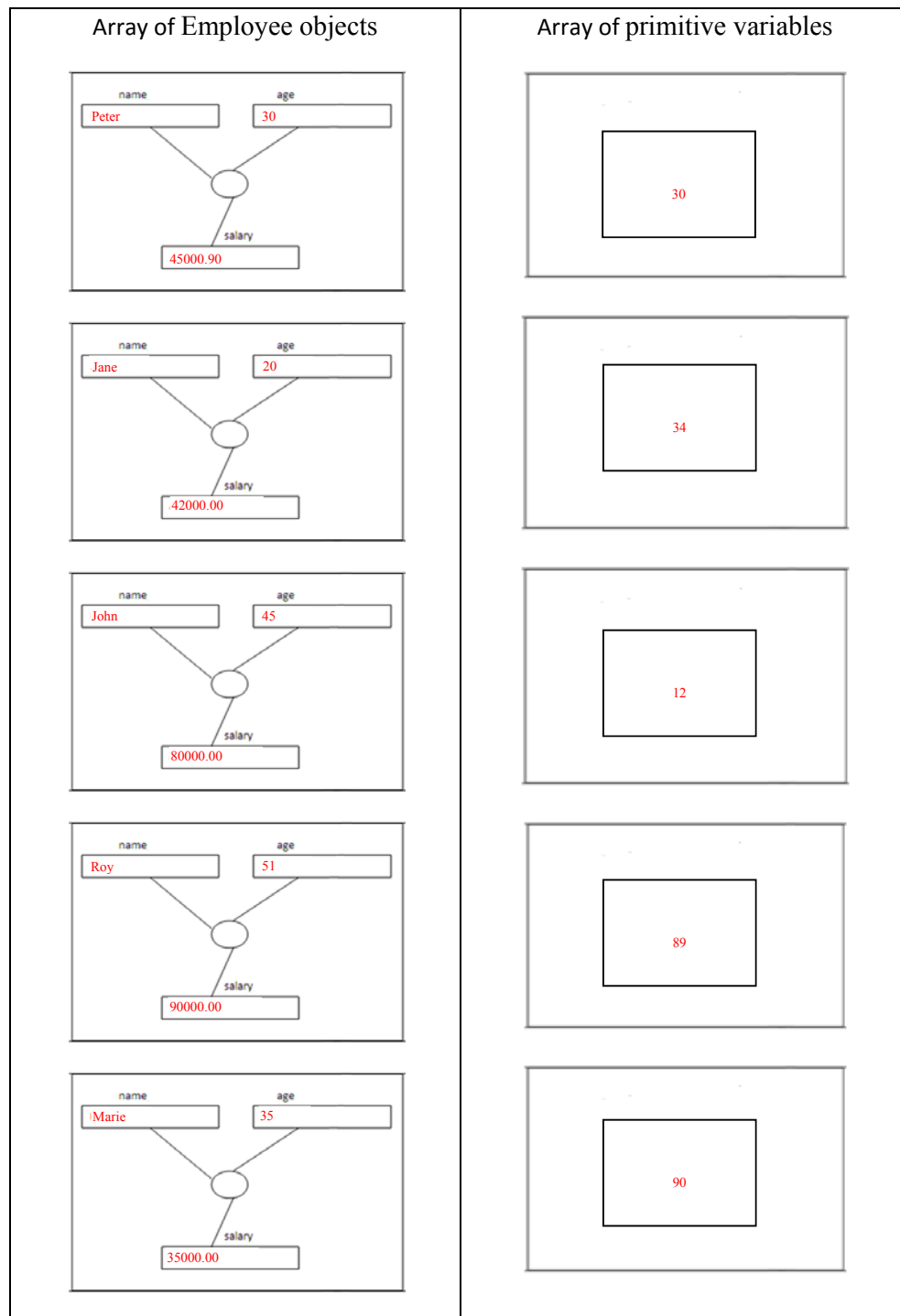
The success of the use of this image in teaching the concept of passing parameters was evaluated by students answering the Mini-questionnaire-3 (see Appendix E) at the end of the course.



*Figure 3.10.* Main method using `getGrade` method.

According to the survey in Phase One, using arrays for primitive was one of the difficult Java concepts for many students. Many participants in the survey had found using an array for objects was even harder than using an array for primitive variables.





*Figure 3.11. Arrays of objects and primitive types.*

The teaching tool used to improve teaching arrays was the use of worked Java programming examples, along with pictorial representation of an array (see Figure 3.11). The filing cabinet was used as an analogy to provide a pictorial representation of the array structure as shown in Figure 3.11. A complete Java program with an

array of five integer variables was provided to students as a worked example. Students experimented running the program with integer type inputs on an array. The students were asked to run the program a couple of times with different sets of values. The second step of learning was to modify the Java program code to change the length of the array and run the program with different sets of values. The third step of learning was to modify the data type of the array to other types such as double and String. The students were asked to modify the pictorial representation (see Figure 3.11) according to the changes on the program.

The teaching of the use of an array for objects was delayed until the students became familiar with arrays using primitive variables. This concept was found to be difficult for 41% of the students. A worked example was used to explain the concept of using an array of objects as was done with primitive variables. The concept was taught using the Kolb Experiential Learning Model with scaffolding. The students used the worked example and assimilated new knowledge by experimenting and modifying the code of the program using guidelines and scaffolding that were provided. The students were provided with the code of the Employee class (see Appendix B) and the code of the main method that creates the array of five Employee type objects. The complicated and high element of interactivity of the code in the main method such as the creation of an array for objects, use of loops to create objects on the array and input data into the created objects could result in a high germane cognitive load for many students (Beckmann, 2010). Therefore, in an attempt to ease the high germane cognitive load, the whole process was explained as three sets using three pictorial representations to show the structures created on the computer memory and their content after each set of code.

When the code in the main method was run, there was no output displayed on the computer screen. The students were asked to modify the code to print the contents of the five objects using a loop. This task may not impose high intrinsic cognitive load as the loop has been already used twice on the worked example. Due to their varied germane levels, some students needed teacher guidance to continue with the tasks assigned with the worked example. The students were also asked to modify the code by changing the length of the array to accommodate more employee record. The students were then asked to add the fourth instance variable: address to the

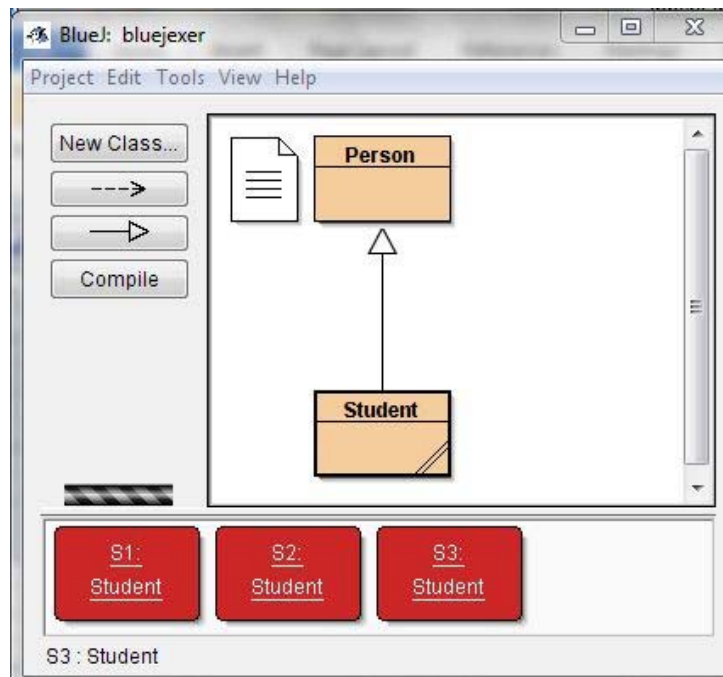
Employee class so that the address of each employee could be stored on the array. This task involved adding code to the Employee class, and the Main method. The students were expected to change the pictorial representations of the three sets before or after the modifications on the Java code. Finally, at the end of the class, the students were given the questionnaire-4 (see Appendix F) to evaluate the success of the pictorial tool and the techniques used in teaching this concept.

### **3.6 Use of BlueJ Visual Tool to teach Java Programming**

BlueJ is a graphical IDE environment developed as a learning environment for the Java programming language. BlueJ generates Java code and the graphical user interface (GUI) can be used by any user without prior knowledge of the Java language (Kouznetsova, 2007; Van Haaster & Hagan, 2004). According to Kouznetsova (2007) BlueJ helps beginners to grasp difficult Java programming concepts easily. Those who do not agree with providing a GUI learning environment for beginners in programming say that it is not a tool for teaching programming concepts due to students' familiarization with dragging and dropping buttons and not concentrating on concepts (Georgantaki & Retalis, 2007).

Inheritance was found to be the most difficult concept for students to understand. The main focus in teaching Java programming concepts in this research has been on managing intrinsic and germane cognitive load with teacher assistance and using mental modeling with pictorial representations. The visual aspect of BlueJ was used for mental modeling classes and inheritance between them. Hands-on sessions supported by the teacher were applied in an activity to ease the germane cognitive load of some learners.

Towards the end of the course, BlueJ was introduced. The students were given the task of creating the Person class and Student sub class using the code generation feature of BlueJ (see Figure 3.12).



*Figure 3.12. BlueJ graphical user interface.*

Students were given a set of tasks to be carried out interactively in the computer room after an introductory lesson on BlueJ (see Figure 4.11). In addition, the BlueJ user manual was provided to the students. The teacher guidance was provided to students who required support to carry out the tasks given to them. The first task was to write the code for the two classes and adding inheritance feature to make the Person class the super class and the Student class the sub class. Then the students experienced creating three objects of Student class using default and alternative constructors. Then they used Mutator methods to set values to objects and Accessor methods to access the contents of objects. Students were asked to identify the methods of the sub class and the methods inherited from the super class. The inspect feature was also used to see the contents of each object.

At the end of the session, Mini-questionnaire-5 (see Appendix G) was used to get student comments on the success of using BlueJ to teach the inheritance concept in Java programming language.

Two questions were included in all the five Mini-questionnaires to find out students' artistic and logical abilities. The aim of having these two questions was to investigate the relationship between brain dominance and mental modeling using

images and worked examples. Each of those questions had five options: Poor; Average; Good; Very good; and Excellent. The participant could select only one option.

### **3.7 Sampling Technique**

Although students at three polytechnics were involved in the first phase, the second phase of the research was carried out with the students who had enrolled in the Introductory Java programming course in the last four years at the Waiariki Institute of Technology. The research in the second phase was based on experimenting on teaching tools and concepts in a class room environment. For this reason, convenience sampling was the only available option to be used in this research. The convenience sampling technique is also known as grab or opportunity sampling in which the sample population available for the research is selected due to its availability. Participation in the survey was entirely optional for the students; however, the percentage of participation in the second phase was around 90% of the class.

### **3.8 Data Collection and Analysis**

Web based questionnaires were used to collect data from students in both phases of the research. An account at <http://freeonlinesurveys.com> was used to create the questionnaires. An email was sent to all students in the class to inform them about the survey and to provide the web address to access the questionnaire. In the email, it was stated that participation in the survey was voluntary. An information page was included in the webpage of the questionnaire. The consent to participate in the survey was indicated by ticking a check box. The participant could not answer the questions on the survey without ticking the consent question. The questionnaire used in phase one was completed by thirty three students. A three week period was allowed for the students to complete the questionnaire.

There were five Mini-questionnaires used in Phase Two. Mini-questionnaire-1 (see Appendix C) was used to evaluate the use of mind-mapping as a teaching tool. Mini-questionnaire-2 (see Appendix D) was used to evaluate the use of mental modeling in teaching class and object Java concept. Mini-questionnaire-3 (see Appendix E) was used to evaluate the use of mental modeling in teaching parameter passing Java

concept. Mini-questionnaire-4 (see Appendix F) was used to evaluate the use of mental modeling in teaching an array concept in the Java language. Mini questionnaire -5 (see Appendix G) was used to evaluate using the BlueJ Visual tool to teach Java programming.

The data collected from the questionnaires were both qualitative and quantitative. Despite some summative information provided by the website itself, data had to be exported to other packages for analysis. The quantitative data were analyzed using Excel and Statistix-7 and the qualitative data were exported to Nvivo9 for processing. Open-ended questions, such as commenting on the usefulness of a tool in understanding a programming concept, were included and categorized as positive and negative comments using the Nvivo9 statistical package. Student achievement results before and after introducing the new teaching tools were also used as a measure of impact on student learning.

### **3.9 Assumptions and Limitations**

The data sample in Phase One included the participants from three polytechnics in New Zealand. Although a criterion for participant selection was not applied, the student participation depended on their decision to participate in the survey. The convenience sampling method was applied in phase two of the research in which data were collected from students at the Waiariki Institute of Technology. Although a convenience sample is not the best sample of the population, it was used because of the availability of the data sample at the Waiariki Institute of Technology. The maximum number of students enrolled in the introductory programming course using the Java language was 43 in year 2008 and the course was offered once a year. The annual enrolment of students for the introductory programming course at the Waiariki Institute of Technology has dropped significantly in subsequent years. Participation in the study was entirely voluntary and not all students were expected to complete the questionnaires used in this research. Random sampling was not possible due to the limited number of students enrolled in this course.

There were some limitations due to financial constraints and availability of limited tools that could be used to teach Java programming concepts in this study. The low cost options such as pictures were used to aid in mental modeling. The data were

processed using the statistical software available at the Waiariki Institute of Technology.

### **3.10 Ethical Considerations**

As a number a surveys were used in this research, ethics approval was obtained from the Human Research Ethics Committee at Curtin University, Australia prior to conducting the surveys (see Appendix-P). A copy of the approval from Curtin University and the details of the data to be collected and the copies of questionnaires were supplied to the Research Committee of the Waiariki Institute of Technology.

Students who had completed an introductory level of programming at three polytechnics were chosen to participate in phase one of the research. They were requested to take part in this research by email. The questionnaire was available on the internet at <http://freeonlinesurveys.com> site. In addition, a printed information sheet was handed over to students by a staff member of the department requesting students to participate on the survey. Students had the option of completing the survey on a hard copy or online. Participants were given an opportunity to ask any question with regards to the research by email or verbally. The stored data at this site have been protected by a username and a password. The purpose of the research was clearly stated on the information sheet and was available on the internet webpage (see Appendix H) for the participant to read before taking part in the survey. Students were assured their rights to withdraw from the research at any time without prejudice or any negative consequences. In case of withdrawal, they were assured that the data collected prior to their withdrawal would not be processed and would be destroyed. Participants were asked not to include their personal details such as name or identification codes on the questionnaire. The consent was obtained by using a check button on the webpage. The data fields were not highlighted for data entry until the check button for the consent was ticked. Students who completed the hardcopy of the questionnaire were given a hardcopy of the consent form (see Appendix I) to be signed.

In Phase Two of the research, data were collected using five Mini questionnaires. As in Phase One, the Mini-questionnaires were available online and in paper form. The information sheet-2 (see Appendix J) was made available to students on the web

pages and also on printed form for those who chose to complete the questionnaire on paper. As in information sheet-1, the information sheet-2 contained the purpose of the research, students' rights to withdraw from the research at any point of time, confidentiality of data, the fact that the participation is voluntary, and the need of their consent to participate in the survey. Participants had to sign the consent form prior to filling in the Mini-questionnaire. As in Phase One, a printed consent form was made available for those who completed the Mini-questionnaire on hardcopy.

### **3.11 Summary**

Chapter Three has covered the methodology used to address the research questions outlined in Chapter One. The instructional design model ADDIE was followed throughout the research. A summary of the findings of the literature survey related to this methodology was provided in the introduction section. Such literature included: the recent applications of Cognitive Load Theory (CLT) with the emphasis on efficient use of working memory and lowering cognitive loads using worked examples; use of mental modeling in teaching and Hemispheric Dominance Theory (HDT). In addition, a brief description of the existing teaching approaches and the reason for using concept first approach was justified. The research focus and the theoretical and practical aspects of the significance of this research were discussed in section two. The research questions targeted in the two phases of the research were discussed in section three. The findings in Phase One of this research were used in deciding the concepts to be used and the ways of delivering them were included in phase two. The findings in phase one revealed that the majority of the student population (76%) were kinaesthetic learners followed by visual learners (56%). Therefore, teaching tools which combined visual and kinaesthetic learning were chosen for the experiment in Phase Two. In addition, every effort was made to reduce cognitive loads to maximise the use of the working memory of the learner. In particular, partly completed work assignments and scaffolding were used throughout this research. A set of pictures and analogies were introduced and used them consistently in work assignments to help create mental models for the students in the second phase. Five Mini-questionnaires were used to collect data. The analysis of the collected data and consequent discussion follow in Chapter Four.



## **CHAPTER 4**

### **FINDINGS AND DISCUSSION**

#### **4.1 Introduction**

In this chapter, data collected in the two phases of the research are analysed and discussed. This chapter also includes how the findings in Phase One contributed to the design and development of activities to be experimented with in Phase Two. The identification of difficult Java programming concepts and preferred teaching styles from the students' point of view were the most important findings in Phase One. In addition, the learners were categorised into three groups: auditory, kinaesthetic and visual. The percentages of students in each category were calculated and proportionately catered for in the teaching activities designed for Phase Two. The summarised details of the difficult concepts and teaching styles, and type of learners are presented in tables and charts in this chapter.

In Phase Two, five teaching activities were developed and used to teach the difficult areas identified in Phase One. Each activity was evaluated by the students using a mini questionnaire. The visual tools such as Mind mapping and BlueJ were experimented with in two activities and findings of their usefulness have been included in the chapter. In addition, as suggested by Paivio (2006) and Van Gog et al. (2010), the use of pictures and analogies for mental modeling and schema building were used. The findings of the usefulness of mental modeling used in five activities are discussed in this chapter. Worked examples and teacher guidance were used in activities in an attempt to reduce germane and intrinsic cognitive loads as suggested by Beckmann (2010). The findings from the feedback from students on the effectiveness of such techniques are also included in this chapter.

#### **4.2 Findings in Phase One**

##### **4.2.1 Difficult Concepts of the Java Language**

Thirty three students from the three polytechnics participated in the survey conducted in Phase One of the research. Each participant was asked to indicate the

level of difficulty of each Java programming concept listed in the questionnaire. The five difficulty level options for a concept were numerically coded as: too difficult (5), very difficult (4), difficult (3), not difficult (2), and very easy (1). The data were analysed using Microsoft Excel and Statistix7 statistical software packages. The mean value ( $\mu$ ) and the standard deviation ( $\sigma$ ) for each concept were calculated (see Table 4.1). The standard score (Z Score) was calculated using the formula:

$$z = \frac{x - \mu_x}{\sigma_x}$$

The probability of the percentage of the student population  $p$  ( $3 < x < 5$ ) with difficulty in level 3, 4 and 5 of each concept was calculated using Z-table. The last column of the Table 4.1 shows the percentage of the students who found each concept difficult, very difficult or too difficult.

According to the results shown in Table 4.1, inheritance was the most difficult concept with 44.8% of the students finding it difficult to understand. The second most difficult concept was the use of arrays for objects with 41.7% of the student population finding it difficult. Using arrays for primitive variables was found to be difficult for 37.4% of the student population. The use of text files was also found to be hard for 38.6% of the student population. Of the student sample, 27.6% had difficulties in understanding the class and object concept (see Table 4.1). New teaching tools were devised in the second phase of the research to improve the teaching of most of the concepts identified as difficult in Phase One.

**Table 4.1**  
*Difficulty Levels of Java Concepts*

<b>Concept (variable – x)</b>	<b>Mean (<math>\mu</math>)</b>	<b>Standard Deviation (<math>\sigma</math>)</b>	<b>Z Score</b>	<b>% found difficult in population  (<math>p(x \geq 3)</math>)</b>
Variable types	1.58	0.65	2.15	1.6%
Variable categories (local, instance, parameter)	2.27	1.01	0.72	23.8%
Conditional statements (if then else)	2.12	0.96	0.92	17.8%
Repetitive statements	2.18	0.98	0.83	20.3%
Class & Object concept	2.24	1.00	0.76	27.6%
Create an object using a class	2.09	0.80	1.13	12.9%
Class Template	2.30	0.92	0.76	27.6%
Returning a value from a method	2.03	0.95	1.02	15.4%
Testing and debugging	2.33	0.96	0.70	24.2%
Preparation of test data	2.33	0.92	0.72	23.6%
Using arrays for primitive variables	2.67	1.05	0.32	37.4%
Using arrays for objects	2.79	0.99	0.21	41.7%
String manipulation	2.21	0.86	0.92	17.9%
Text files	2.67	1.14	0.29	38.6%
Logic depiction methods	2.45	0.90	0.60	27.4%
Inheritance	2.88	0.96	0.13	44.8%

Prasad and Li (2004) conducted a similar survey to find difficult areas in C++ programming for the introductory level students and found the use of arrays as the most difficult area. But, their study was focussed on the structural aspects of the difficult levels rather than on conceptual aspects.

#### 4.2.2 Correlation between Difficulty Levels and Skills

The total difficulty levels for each participant, for all the concepts were calculated by adding quantified values of the difficult levels together. This total difficulty level of the participants was used as one variable of the correlation. The logical and artistic skill levels were numerically coded: “excellent” as 5, “very good” as 4, “good” as 3, “average” as 2 and “poor” as 1. The total skills of both logical and artistic were calculated by adding the two numerical values as both\_skills. The other three variables used in the correlation were the logical skill, artistic skill and both\_skills score of the participants.

The Pearson's formula to calculate correlation of a paired  $(x_i, y_i)$  data sample is:

$$r = \frac{\sum_{i=1}^n (x_i - \mu_x)(y_i - \mu_y)}{\sqrt{\sum_{i=1}^n (x_i - \mu_x)^2} \sqrt{\sum_{i=1}^n (y_i - \mu_y)^2}}$$

where  $\mu_x$  and  $\mu_y$  are mean values of the two variables x and y.

The Statistix7 which supports Pearson's correlation coefficient was used to calculate the correlation coefficient of the variables described above. The correlation coefficient ( $r$ ) between total difficulty levels and the logical skill variables was found to be -0.14. This figure indicates that there is zero or insignificant correlation between students' total difficulty level and their logical skill. But the correlation coefficient between students' total difficulty level and their artistic skill was found to be 0.24 ( $p = 0.19$ ). This indicates a weak positive correlation between artistic skill and the total difficulty level of Java programming concepts (see Figure 4.1).

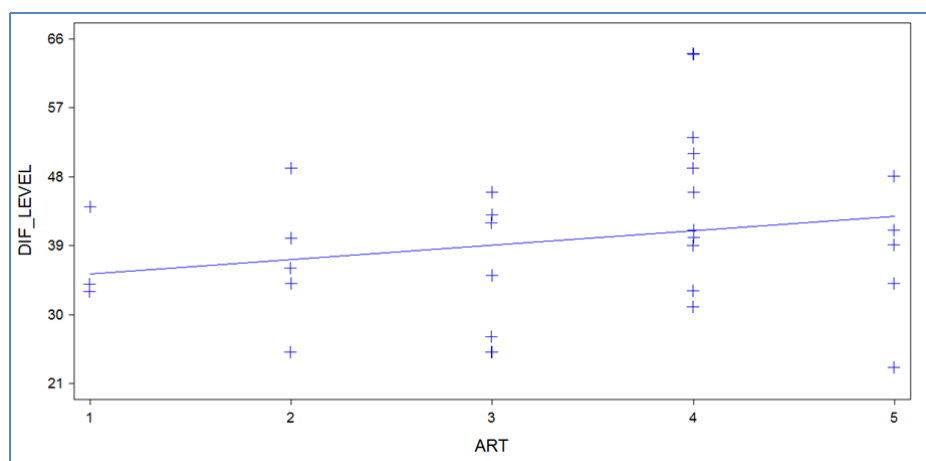


Figure 4.1. Scatter diagram of total difficulty level vs. art skills of students.

In addition, the correlation between students' total difficulty level and the students with both artistic and logical skills was found to be 0.07. This figure indicates zero or insignificant correlation between the both skills and the total difficulty levels of the student population.

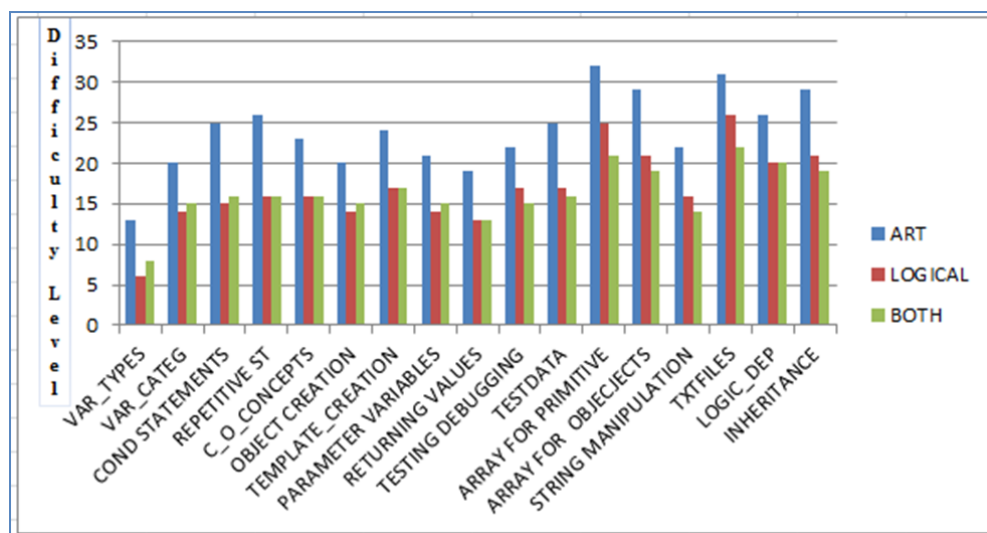


Figure 4.2. Difficulty levels of learners of different categories.

The bar chart in Figure 4.2 was plotted with total difficulty levels of students with art, logical and both skills on the y axis and the list of concepts on the x axis. Despite the weak positive correlation found between artistic skills and difficulty level, Figure 4.2 clearly shows that the students with artistic skills found it more difficult to understand almost all the Java concepts than did the students with logical and both artistic and logical skills.

### 4.2.3 Types of Learners

An item in the questionnaire enabled grouping of the participants into three categories of learners: auditory, kinaesthetic, and visual. The participants were given the option to choose more than one of these types of learners. In the sample, the majority of the student population (76%) was found to be kinaesthetic. The percentage of visual learners was found to be 55%, and the percentage of auditory type of learners was found to be 24% (see Figure 4.3).

The kinaesthetic combined with visual was also found to be significantly higher (39%). There were few learners (12%) with the combination of visual and auditory learning types (see Figure 4.3).

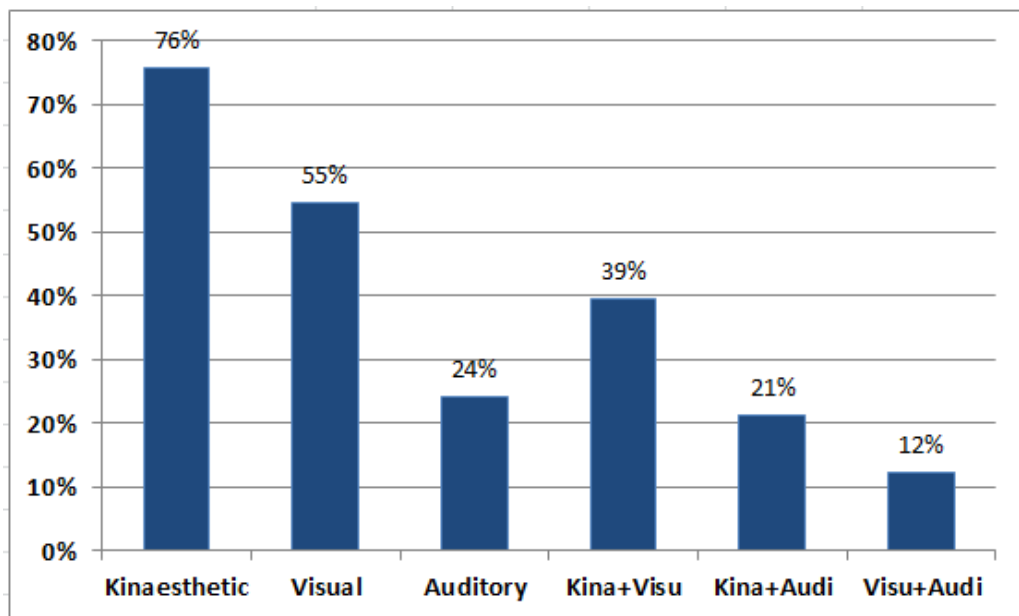


Figure 4.3. Summary of the types of learners.

The results in Figure 4.3 indicate that the teaching tools to be used to enhance teaching Java programming concepts could be more effective if kinaesthetic, visual or both were incorporated in teaching. These findings were taken into consideration in designing teaching activities which were to be used experimentally in phase two of the research. These activities integrated visual aspects by using visual tools and kinaesthetic aspects by using partially completed exercises and teacher guidance. The visual tools were carefully chosen to build up mental models. In addition, worked examples and teacher guidance were used taking into consideration the management of working memory aspects of Cognitive Load Theory (CLT).

#### 4.2.4 Teaching Styles for Different Concepts

The students were asked to suggest the most suitable teaching style for the teaching of each concept from the three given options: auditory, kinaesthetic, and visual. Table 4.2 contains the results of this process as the percentages of styles suggested by the students.

**Table 4.2**  
*Teaching Styles for Different Concepts*

Concept	Auditory	Kinaesthetic	Visual
Variable types	21.2%	48.5%	27.3%
Variable categories (local, instance, parameter)	18.2%	45.4%	33.3%
Conditional statements (if then else)	06.1%	66.7%	27.3%
Repetitive statements	09.1%	63.6%	24.2%
Class & Object concept	09.1%	45.4%	36.4%
Create an object using a class	12.1%	54.5%	30.3%
Class template	12.1%	48.5%	36.4%
Use of parameter variables	15.1%	51.5%	30.3%
Returning a value from a method	21.2%	51.5%	24.2%
Testing and Debugging	06.1%	72.7%	18.2%
Preparation of test data	18.2%	51.5%	27.3%
Using arrays for primitive variables	09.1%	63.6%	24.2%
Using arrays for objects	09.1%	63.6%	24.2%
String manipulation	12.1%	60.6%	24.2%
Text files	09.1%	60.6%	27.3%
Logic depiction methods	12.1%	51.5%	33.3%
Inheritance	06.6%	57.6%	36.4%

According to the findings, the kinaesthetic way of teaching was suggested by the majority of students for teaching all the concepts. These findings were considered in designing teaching activities to teach the difficult concepts identified. In each activity, more than 50% of the time was allocated to hands-on sessions using worked examples.

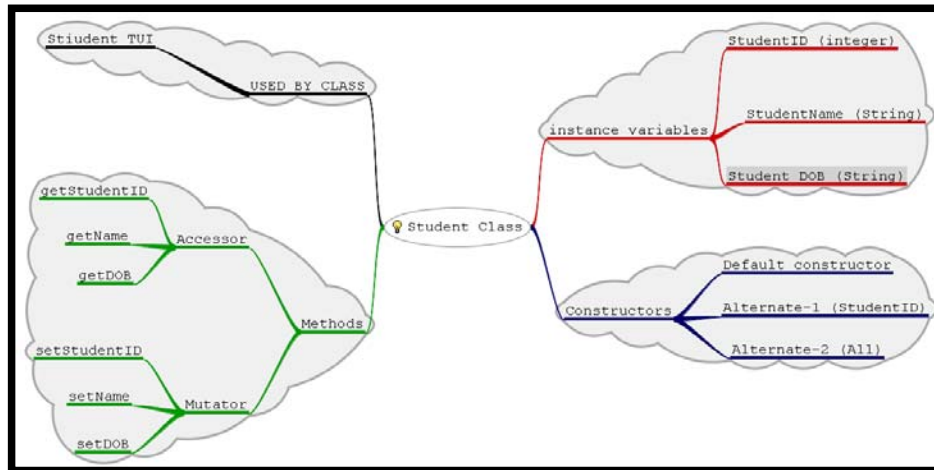
### **4.3 Findings in Phase Two**

The Activity-1 (see Figure 4.4) was used to teach Java programming class structure. This activity had a Mind map to show the class structure, a worked example of a Java class (see Appendix A) and a hands-on exercise. Thirty participants who attended this teaching activity took part in the survey by completing Mini questionnaire-1 (see Figure 4.4).



## Activity – 1

Use the Student.java files to complete this hands-on exercise. You may ask for assistance of the teacher as required.



1. The above Mind Map describes the code given in the Student.java class (worked example).
2. Locate the Java code of instance shown on the Mind Map variables in Student.java file.
3. Locate the Java code of each constructor shown on the Mind Map in Student.java file.
4. Locate each Java code of each method shown on the Mind Map in Student.java file.
5. Find out the difference between accessor and mutator methods by observing the code given Student.java file.
6. Draw a new instance variable leaf (branch) with the name studentAddress to the mind map.
7. Write the Java code for the new instance variable and modify the default and the second alternative constructor in the file Student.java. Compile the class.
8. Draw two methods leaves (branches) with the names setAddress and get address on the mind map.
9. Write the Java code for the two new methods in the file Student.java.
10. Draw a new alternative constructor leaf (branch) with the parameter name aStudentdob to the mind map.
11. Write the Java code for the new alternative constructor and compile the code.
12. Now with the experience you had keep adding more instance variables, constructors and methods.

Figure 4.4. Activity - 1.

### 4.3.1 Mini Questionnaire-1 Findings

#### 4.3.1.1 Findings from the quantitative data

**Table 4.3**

*Summary of Artistic and Logical Skills of the Participants in Mini Questionnaire-1*

Skill Level	Artistic %	Logical %
Poor	0.0	0.0
Average	66.7	3.3
Good	23.3	40.0
Very good	3.3	33.3
Excellent	6.7	23.4

According to Table 4.3, the participants' logical, analytical thinking, and mathematics skills are significantly higher than their artistic skills, such as singing, painting, and writing poetry. About 97% of the students thought that they had good, very good or excellent logical skills whereas the percentage of students with good, very good or excellent artistic skills was 33.3%. The percentage of the students with average artistic skills was about 67%.

The participants indicated the most useful aspect of teaching Activity-1 by choosing one of the options: Mind map, worked example, teacher guidance, or hands-on exercise. A summary of the feedback from the students is given in Table 4.4.

**Table 4.4**

*Summary of the Most Useful Teaching Tools/Methods Used in Activity-1*

Teaching tool/method	Percentage of responses
Mind Map	20.7%
Worked example	43.3%
Teacher guidance	10.0%
Hands-on exercise	26.7%

As shown in Table 4.4, the most useful teaching component of this activity was the use of a worked example (43.3%). The worked example was used with the intention

of easing the intrinsic cognitive load of learners. The visual aspect of the teaching activity was applied using a Mind map with the intention of creating a mental model to help learning. Teacher guidance was used to support students who needed extra support so that their germane cognitive load could be reduced. This result indicates that the cognitive aspect of learning is more significant than that of teacher guidance and mental modeling.

The majority of the students who suggested the worked example to teach this concept had good logical skills (69%) and average artistic skills (69%). Most of the students with very good logical skills (66%) and average artistic skills (67%) preferred the Mind map. All the students who suggested teacher guidance as the most useful had very good logical and average artistic skills. The hands-on exercise was highly valued by 50% of the students with excellent logical skills and 50% of the students with average artistic skills (see Table 4.5).

**Table 4.5**

*Preference of Tools/Methods of Students with Logical and Artistic Abilities in Activity-1*

<i>Tool/Method</i>	Logical					Artistic				
	Excellent	V Good	Good	Average	poor	Excellent	V Good	Good	Average	poor
Mind map	0%	66%	17%	17%	0%	0%	0%	67%	33%	0%
Worked example	23%	8%	69%	0%	0%	8%	0%	23%	69%	0%
Teacher guidance	0%	100%	0%	0%	0%	0%	0%	0%	100%	0%
Hands-on exercise	50%	25%	25%	0%	0%	13%	12%	25%	50%	0%

One of the questions asked in the Mini questionnaire-1 (see Appendix C) was for the students to suggest the best combination of the tools/methods to be used. The percentage of the participants who suggested all the tool/methods: Mind map, worked example, teacher guidance, and hands-on example to be used were 60%.

While 97% of the participants suggested Mind map, worked example, and hands-on example, teacher guidance was suggested to be used by 73% of the participants of the survey (see Table 4.6).

**Table 4.6**

*Summary of Tools/Methods Preferred by Students in Activity-1*

	Mind map	Worked example	Teacher guidance	Hands-on exercise	All tools	Mind Map & Worked example & Hands-on work
Percentage of students	97%	90%	73%	100%	60%	87%

#### 4.3.1.2 Findings from the qualitative data

One of the questions in the Mini questionnaire-1(see Appendix C) was to find out the usefulness of the Mind map in learning the concept. Almost all (97%) of the participants found the Mind map useful to learn the Java class concept. The following were some noteworthy comments:

*Mind map puts everything into a clear view which made it easier to write my Java programs and understand the syntax.*

*Mind map gives a clear idea of what is really going on behind the code an idea of types of objects and variables to be used.*

*Mind map shows the components of the Java class template clearly.*

*Mind map helped me to differentiate instance variables, constructors and methods.*

*The visual image of the Mind map helps to remember the class template.*

*It gives colourful braches highlighting the structure of the Java class.*

*The Mind map is an excellent tool because of its visual aspect.*

*I think Mind maps help in understanding Java better because it turns the spoken and/or written into a visual one.*

The usefulness of the whole exercise was one of the questions in the Mini questionnaire-1, for which almost all the participants commented positively.

There were 12 comments about the usefulness of worked example. Many had commented about the usefulness of worked example to understand the class template and its logic. Some students had mentioned the usefulness of using Mind map with worked examples. Some of these comments were:

*Worked examples give us a better understanding of Java and how Java code work.*

*Worked examples are helpful when used with the Mind map picture.*

*It helps because there were examples of how the template worked. I was able to figure out what was wrong when it went wrong.*

*Worked examples are more practical better than learning theory using PowerPoint slides.*

*Java coding is easier when used worked examples.*

*It was good as there were examples to explain of how the logic worked.*

There were seven comments on the hands-on exercise given to modify the worked example provided.

*We were able to work on the code by ourselves without much effort.*

*This type of learning is better than teaching on the board.*

*This exercise is more exciting and easy to understand.*

*Hand-on exercises are very useful to learn Java.*

*We need to work on the code by ourselves to learn Java.*

*The PowerPoint slides are boring. In this I can experience and view what is going on instead of listening to a lot of theory*

Five students commented on the visual aspects of Mind map. They thought that Mind map not only helped to understand but also enabled them to write Java code faster. The noteworthy comments are:

*The picture in the Mind map helped me practically write code in Java programming.*

*It makes the classes and methods easier to see. It enables for me to write Java code quicker. I believe that Mind map help me to understand Java code better.*

*Mind map lets you visualise what you practically do in Java programming.*

Three students found teacher guidance useful as it enabled them to to contact the tutor for difficult questions whenever required.

#### **4.3.2 Mini Questionnaire-2 Findings**

##### *4.3.2.1 Findings from the quantitative data*

Teaching Activity-2 (see Figure 4.5) was used to teach the creation and manipulation of Java objects. It contained pictures, analogy, worked example, hands-on exercise and teacher guidance.

### Activity -2

Use the Employee.java file to complete this hands-on exercise. You may ask for assistance of the teacher as required.

### Information of Employees

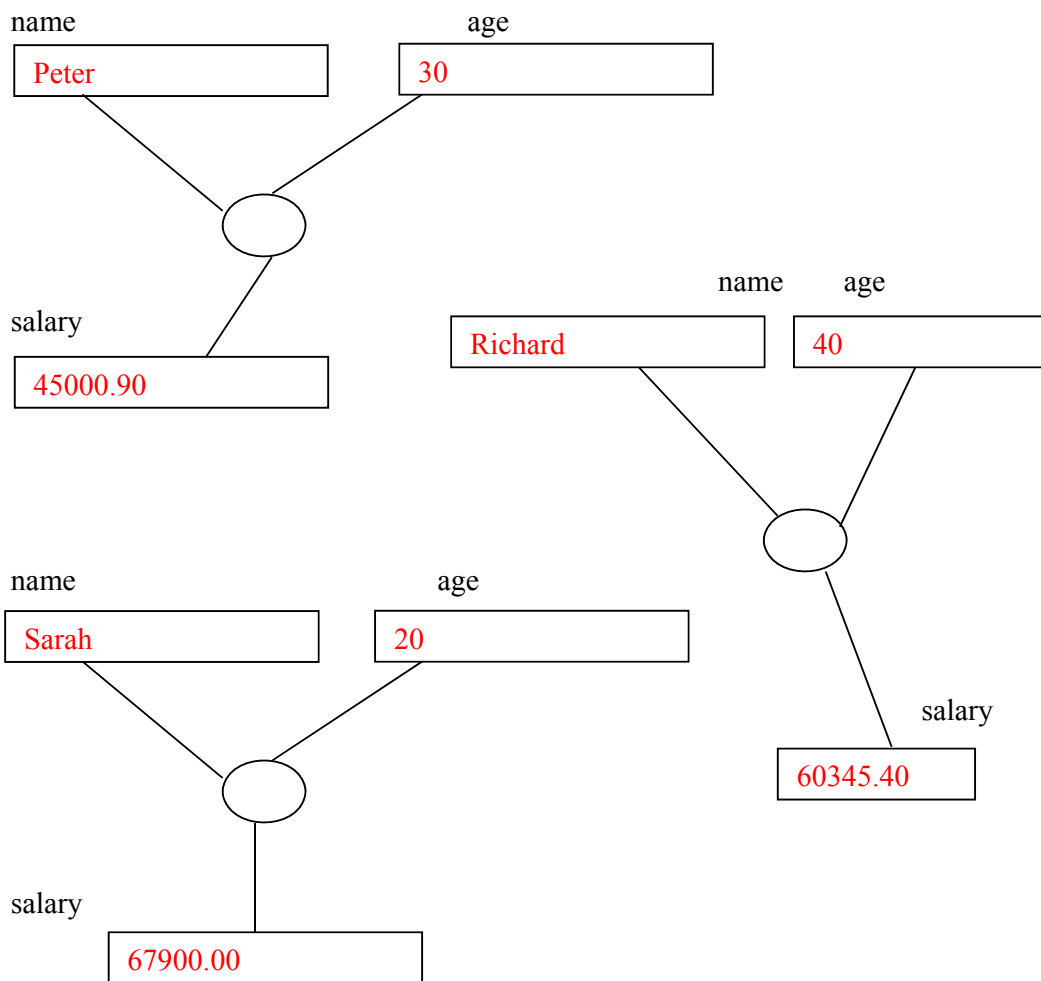
Employee Name	Employee Age	Salary
Peter	30	45000.90
Richard	40	60345.40
Sarah	20	67900.00

If we are to store these data (in red) in a Java program, we have to create a class or template first. (The class Employee.java is provided to you.)

Then we create three objects using that class in the main method.

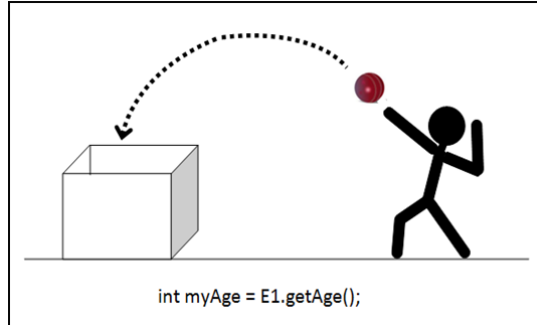
Finally we assign values to the three objects.

### Objects



### Analogy

Getting a value from an instance variable of an object.



### Exercise

Modify the Employee class with an additional property called gender. Add a method called setGender and modify the printEmployeeDetails method.

Write Java code to add the values as shown in the table given below to the three objects created in EmployeeTester class.

### Information of Employees

Employee Name	Employee Age	Salary	Gender
Peter	30	45000.90	M
Richard	40	60345.40	M
Sarah	20	67900.00	F

Figure 4.5. Activity - 2.

At the end of the teaching activity, Mini questionnaire-2 (see Appendix D) was used to collect feedback from students. Twenty eight students participated in the survey in four consecutive years (2009, 2010, 2011 and 2012). The summary of the findings of the participants' artistic and logical skills are listed in Table 4.7.

**Table 4.7**

*Summary of Artistic and Logical Skills of the Participants in Mini Questionnaire-2*

Skill Level	Artistic %	Logical %
Poor	21.4	0.0
Average	17.9	7.1
Good	17.9	39.3
Very good	28.6	39.3
Excellent	14.3	14.3



According to Table 4.7, the participants' logical, analytical thinking, and mathematical skills are higher than their artistic skills such as singing, painting, and writing poetry. Some participants (21%) thought that their artistic skill was poor.

**Table 4.8**

*Summary of the Most Useful Teaching Tools/Methods Used in Activity-2*

Teaching tool/method	Percentage of responses
Worked example	25.0%
Teacher guidance	14.3%
Hands-on exercise	25.0%
Analogy /Picture	35.7%

One of the items in the questionnaire was to choose the best/most useful tool/method for teaching Activity-2. According to the results presented in Table 4.8, the most useful tool/method was the picture and analogy used in the activity. The students rated the use of worked example and hands-on exercise as the second most useful component of this teaching session. These results indicates that pictures/analogs, worked example and hands-on exercise are most effective in teaching this concept.

The majority of the students who were very good in both logical (57%) and artistic (43%) skills preferred the worked example. The majority of students who suggested hands-on exercise had good logical (42%) and artistic skills (42%). The majority of students who suggested teacher guidance as the most useful had good logical skills (75%) and average (75%) artistic skills. The use of Star Structure (see Figure 3.7) and analogy (see Figure 3.8) was highly valued by most students with very good logical (50%) but poor artistic (40%) skills (see Table 4.9)

**Table 4.9**

*Preference of Tools/Methods of Students with Logical and Artistic Abilities in Activity-2*

<i>Tool/Method</i>	Logical					Artistic				
	Excellent	V Good	Good	Average	Poor	Excellent	V Good	Good	Average	Poor
Analogy/Picture	20%	50%	30%	0%	0%	20%	20%	20%	0%	40%
Worked example	0%	57%	29%	14%	0%	0%	43%	0%	27%	20%
Teacher guidance	0%	0%	75%	25%	0%	0%	25%	0%	75%	0%
Hands-on exercise	29%	29%	42%	0%	0%	29%	29%	42%	0%	0%

A question was asked of the participants to indicate the best combination of the tools/methods to be used in this activity. Many participants (64%) suggested all four teaching techniques: analogy /picture, worked example, teacher guidance, and hands-on example to be used. Most participants (93%) suggested a combination of analogy/picture, worked example, and hands-on example to be used. Teacher guidance was suggested by 68% of the participants (see Table 4.10).

**Table 4.10**

*Summary of Tools/Methods Preferred by Students in Activity-2*

	Analogy & Picture	Worked example	Teacher guidance	Hands-on exercise	All tools & methods	Analogy Picture, Worked example & Hand-on work
Percentage of students	89%	93%	68%	96%	64%	93%

#### 4.3.2.2 Findings from the qualitative data

There were 26 positive comments from participants of the survey giving different reasons for the usefulness of Activity-2 in understanding the concept of creation and manipulation of objects.

The following are some noteworthy comments.

*Seeing pictures helps to understand the concept being explained.*

*The pictures help our minds to make the connection between how things interact with each other.*

*Pictures help to visualise concepts.*

*Pictures better help to form a picture in one's mind about the concepts and methods of Java.*

*Your brain remembers pictures more easily.*

*Understanding concepts can be difficult for new students. Using sample diagrams, it becomes easier.*

*Picture made the lesson easier to relate in the head. Pictures really help to describe the concepts that many struggled to get.*

*I think star structure helps in understanding Java because it shows written code in a visual manner.*

*The visual image of the star structure helps to remember the class template & how an object looks like.*

*It provides a visual stimulant for your brain rather than just following steps.*

*The analogy helped me to remember and understand the concept better.*

The participants were asked to suggest any other ways that could be used to improve teaching this concept. The suggestions include:

*Nothing is required to change in teaching style.*

*Less samples of Java files so we can learn to write it all ourselves.*

*Facilitate more hands on learning.*

*Provide less worked examples so that they learn to write them by themselves, and go through or provide answers so that students can check their work.*

One question in the Mini questionnaire-2 allowed the students to make comments on the usefulness of the hands-on session using teacher guidance, worked example, analogy and the pictorial representation to understand creation and manipulation of Java objects. 23 participants had given positive comments. There were seven positive comments about the worked example used in Activity-2. Typical useful comments were:

*This lesson was very good. I like this style of lesson using worked examples over the standard PowerPoint.*

*I could experiment more with worked examples and get experience doing it ourselves.*

*Practical sessions help me to understand and remember Java code.*

*Worked examples are good and we can experience writing the code.*

*Pictures and worked examples are very useful.*

Some found worked example along with pictures useful in understanding creation and manipulation of objects. Some other students had enjoyed using worked example and hands-on exercise which had made the concept easier to grasp for them.

Ten students commented on the practicality of using hands-on exercises. Some commented that their learning was more effective when they used hands-on exercise. Many expressed their aversion to teaching using PowerPoint slides.

Examples of such comments were:

*Yes. Strongly agree with this as PowerPoint can get tedious and with a hand on approach the students understand concepts through trial and error.*

*I find it hard to just sit and listen. My brain tends to tune out when people talk too much.*

*Because doing it as you learn lets you see what is happening and if you get errors it can be explained to the class.*

*Hand on work is more interactive.*

*Hands-on sessions help because you get actually to do something instead of reading about it.*

*Practical or hands-on is much easier to understand.*

Four students had written comments about the usefulness of teacher guidance in this activity. Two of the comments were:

*Teacher guided practical sessions are useful for me.*

*Yes, if you get errors you can ask the teacher.*

#### **4.3.3 Mini Questionnaire-3 Findings**

Activity-3 (see Figure 4.6) with an analogy symbolizing two methods as two stick men, one assigning work to the other, was used to teach parameter passing.

Activity - 3

Use the Employee.java file to complete this hands-on exercise. You may ask for assistance of the teacher as required.

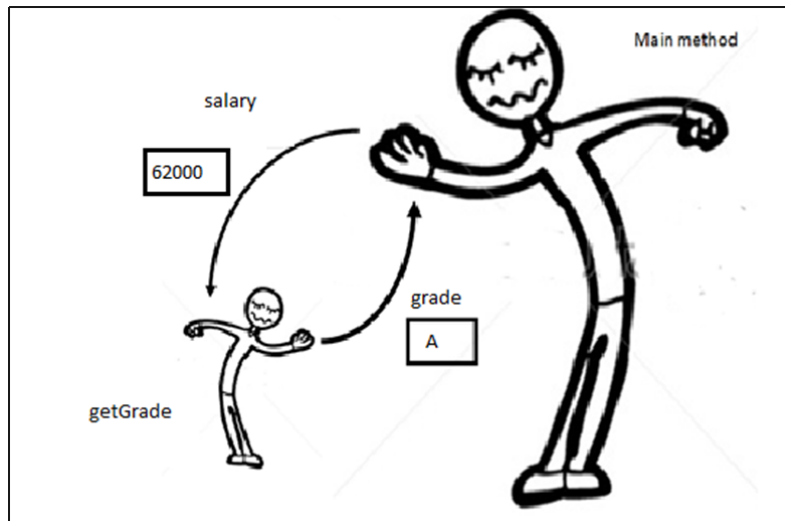
##### **Java Code**

```
public char getGrade(double salary)
{
    char ch=' ';
    if (salary >= 60000)
```

```

ch= 'A';
else if ((salary < 60000) && salary > 40000)
ch = 'B';
else
ch = 'C';
return ch; }

```



1. Open the Employee.java class.
2. Create EmployeeTester.java class with the main method in it.
3. Copy getGrade method to the Employee.java class.
4. Create an Employee object E1 using the command:  
Employee E1 = new Employee()
5. Add the following command to the main method  
E1.setSalary(40000);  
char grade = E1.getGrade(E1.getSalary());  
E1.printSalary();  
System.out.println(grade);  
  
E1.setSalary(70000);  
grade = E1.getGrade(E1.getSalary());  
E1.printSalary();  
System.out.println(grade)
6. Run the program.

You should get the answer  
40000.0  
C  
70000.0  
A  
If you do not get the answer get help from the teacher.

7. Now by using the picture analogy, explain why you get the output C and A to another student in your class.

8. Use the getGrade method with different values in your program.

Exercise: Write a method called ageGroup to which a value of age is passed using an integer type parameter and then the method returns a string value according to the following criteria. Test the method in the main method using different values.

Age group	Return value
>=65	Senior citizen
>=16 and <65	Adult
<16	Child

Figure 4.6. Activity - 3.

In this activity, the possibility of applying schema theory (Ashcraft & Radvansky, 2010) according to which a new schema (knowledge) is created using existing schema (Driscoll, 2000; Mead et al., 2006) was investigated. At the end of the teaching activity Mini questionnaire-3 (see Appendix E) was used to evaluate the suitability of Activity-3 for the purpose. The 21 participants who attended the teaching session took part in this mini survey.

#### 4.3.3.1 Findings from the quantitative data

All the participants of this survey had good, very good or excellent logical skills. Their artistic skills were not as good as logical skills. About 57% of the participants had good artistic skills and the rest had average or poor artistic skills (see Table 4.11).

**Table 4.11***Summary of Artistic and Logical Skills of the Participants in Mini Questionnaire-3*

Skill Level	Artistic %	Logical %
Poor	9.5	0.0
Average	33.3	0.0
Good	57.1	28.6
Very good	0.0	57.1
Excellent	0.0	14.3

The participants' response to the most useful component of the Activity-3 were summarised and categorised as shown in the Table 4.12.

**Table 4.12***Summary of the Most Useful Tools/Methods Used in Activity-3*

Teaching tool/method	Percentage of responses
Visual/Analogy	50%
Teacher guidance	5%
Hands-on exercise	45%

According to the numbers in Table 4.12, the most useful teaching tool for the participants was visual analogy. Nearly half of the participants (45%) thought that the hands-on exercise used was the most useful aspect in this activity. Although, this percentage (45%) was comparatively lower than the Visual/Analogy aspect (50%) of this activity, it had the highest preference from students in the other four activities. Teacher guidance was not considered as useful by many students. This result indicates that the cognitive aspect of learning, especially on schema building and visual mental modeling is the most important in teaching this Java concept. In addition, the participants similarly valued the kinaesthetic way of learning using hands-on work.

The majority of the students with very good or good ratings in both logical (100%) and artistic skills (70%) preferred the analogy used in this activity. The hands-on activity was preferred by the majority of the participants with good or very good



logical skills (90%). These results show the preference of students with high logical and artistic skills for cognitive and kinaesthetic aspects of learning (see Table 4.13).

**Table 4.13**

*Preference of Tools/Methods of Students with Logical and Artistic Abilities in Activity-3*

<i>Tool/Method</i>	Logical					Artistic				
	Excellent	V Good	Good	Average	Poor	Excellent	V Good	Good	Average	Poor
Visual/Analogy	0%	50%	50%	0%	0%	0%	0%	70%	30%	0%
Teacher guidance	100%	0%	0%	0%	0%	0%	0%	0%	0%	100%
Hands-on exercise	20%	70%	10%	0%	0%	0%	0%	50%	0%	0%

The participants of the Mini questionnaire-3 were asked to indicate the best combination of the tools/methods to be used in teaching. Many participants (67%) suggested a combination of all the three teaching techniques: analogy /picture, teacher guidance, and hands-on example to be used. Almost all (95%) preferred the combination of analogy /picture and hands-on exercise. These findings indicate that students prefer analogy /picture, and hands-on work to teacher guidance (see Table 4.14).

**Table 4.14**

*Summary of Tools/Methods Preferred by Students in Activity-3*

	Visual analogy	Teacher guidance	Hands-on exercise	All tools & methods	Visual analogy Picture, & Hand-on work
Percentage of students	95%	71%	100%	67%	95%

#### 4.3.3.2 Findings from the qualitative data

Almost all the students commented positively on the usefulness of Activity-3 to help understand the concept of parameter passing. More than 50% of the comments were about the visual aspects and the analogy used in the activity. Some notable comments are given below.

*It is easy to remember something with visual like pictures.*

*Pictures make concept less abstract.*

*Pictures give us better understanding in a different perspective.*

*Pictures stimulate student's brain.*

*It is a clear representation of the concept.*

*It helped me to understand and remember Java code.*

*I found this analogy useful to understand the concept of passing parameters*

*Pictures help us to visualise what is really happening.*

*This picture helped me to understand how parameters work.*

*The pictorial explanations are very effective in learning.*

Seven students found the practical aspect of the activity useful to learn this concept. They found the varied work involved, the engagement on the activity and interactivity with hands-on work useful. Some found Activity-3 as an easier way of learning. Such comments include:

*Practical or hand-on work is much easier for me to understand.*

*It helps because you get actually to do something instead of reading about it.*

*Varied work is involved in hand-on sessions.*

*Hand-on sessions are less monotonous.*

*It is more interactive.*

*Practical learning is an easier way of learning*

One student suggested that the teacher should provide more exercises, so that students can apply the acquired knowledge by themselves.

The above comments clearly show that the cognitive aspects of learning using pictorial analogies to help new schema building and worked examples to ease intrinsic cognitive load of learners are the most important in learning this Java concept. In addition, the use of hands-on work to incorporate kinaesthetic aspects of learning has been valued by students in learning this concept.

#### **4.3.4 Mini Questionnaire-4 Findings**

Two teaching activities were used to teach the use of arrays. Activity 4.1 (see Figures 4.7 and 4.8) was used to teach arrays with primitive variables and the Activity 4.2 (see Figures 4.9 and 4.10) was used to teach arrays with objects.

#### Activity-4.1

##### WORKED EXAMPLE – ARRAY OF PRIMITIVE TYPE

```
import java.util.Scanner;
public class testArray
{
    public static void main(String [] args)
    {
        int size=5;
        // create array to store five integer values

        int [] numberArray = new int [size];
        // See the pictorial representation Code set-1

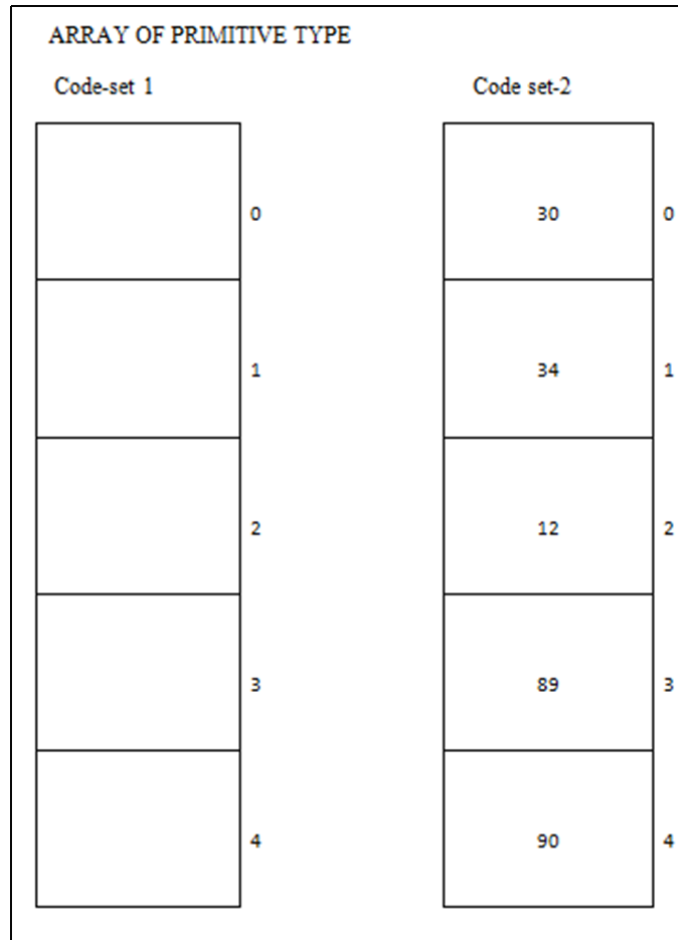
        // Creation of a scanner object
        Scanner in = new Scanner (System.in);

        int index = 0;
        while (index < numberArray.length)
        {
            System.out.print("Enter number " + (index +1));
            // Add name of an employee using a scanner object
            numberArray [index]= in.nextInt();

            index = index +1;
        }
        // See the pictorial representation Code set-2

        //Write Java code to print the five integer values stored on the array values using a
        loop
    }
}
```

*Figure 4.7. Activity – 4.1.*



*Figure 4.8.* Pictorial representation of an array of primitive data – 4.1.

## Activity 4.2

### WORKED EXAMPLE – ARRAY OF OBJECTs

```
public static void main(String [] args)
{
    int size=5;
    // create array to store Employees
    Employee [] emps = new Employee [size];
    // See the pictorial representation Code set-1

    //Create 4 employees objects using default constructor
    for(int index = 0; index < emps.length; index++)
    {
        emps[index] = new Employee();
    }
    // See the pictorial representation Code set-2

    // Creation of a scanner object
    Scanner in = new Scanner (System.in);

    int index = 0;
    String empName;
    int empAge;
    double empsalary;
    while (index < emps.length)
    {
        System.out.print("Enter name" + (index +1));
        // Add name of an employee using a scanner object
        empName = in.next();
        emps[index].setName(empName);

        // Add age of an employee using a scanner object
        empAge = in.nextInt();
        emps[index].setAge(empAge);

        // Add salary of an employee using a scanner object
        empSalary = in.nextDouble();
        emps[index].setSalary(empSalary);

        index = index +1;
    }
    // See the pictorial representation Code set-3

    //Write Java code to print the data of the five employee objects using a loop
}
```

*Figure 4.9. Activity – 4.2.*

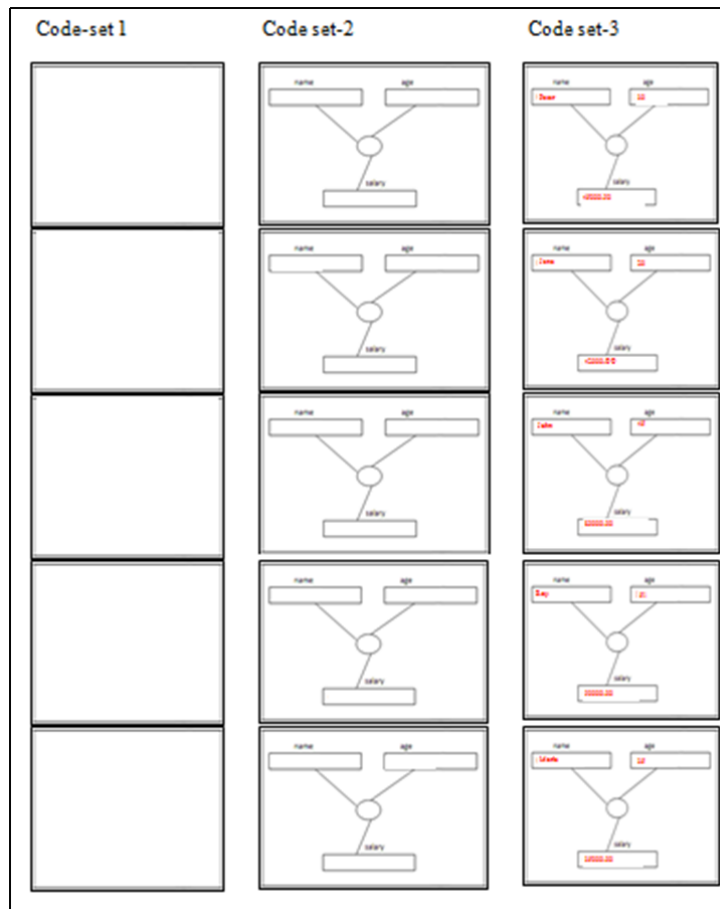


Figure 4.10. Pictorial representation of an array of objects – 4.2.

These teaching activities were designed by combining pictorial array structures, worked examples, hands-on exercises, and teacher guidance. The success of these teaching activities was evaluated by the students who attended the session, by using Mini questionnaire-4 (see Appendix F). The 29 students who completed Activity-4 participated in this survey.

#### 4.3.4.1 Findings from the quantitative data

The participants indicated their logical and artistic skills by choosing one of the five options given in each question. The summarised artistic and logical skills of the participants are shown in Table 4.15.

**Table 4.15***Summary of Artistic and Logical Skills of the Participants in Mini Questionnaire-4*

Skill Level	Artistic %	Logical %
Poor	20.7	0.0
Average	34.5	10.3
Good	17.2	24.1
Very good	17.2	51.7
Excellent	10.3	13.8

The majority of the participants (90%) were found to have good, very good or excellent logical skills. The percentage of the participants with at least good artistic skills was found to be 45%. These figures show that participants' overall logical skills were higher than their artistic skills.

One of the items in questionnaire-4 asked participants to indicate the most useful component of the teaching activity by choosing one of the given options: array structures, worked examples, teacher guidance, and hands-on exercises. The summary of the findings is given in the Table 4.16.

**Table 4.16***Summary of the Most Useful Tools/Methods Used in Activity-4*

Teaching tool/method	Percentage of responses
Array structures	20.7%
Worked examples	31.0%
Teacher guidance	20.7%
Hands-on exercises	27.6%

According to Table 4.16, the most useful teaching tool/method for the students was the worked examples (see Figures 4.7 and 4.9). But a considerable number of participants had chosen other tools/methods as well. This indicates that all the tools/methods used in this activity are important in teaching this concept.



Most students (83%) with very good logical skills were found to be among the participants who had chosen the array structures as the most important teaching tool in this activity. A similar trend was found with 67% of the participants with very good logical skills choosing worked examples as the most important method of teaching (see Table 4.17).

**Table 4.17**  
*Preference of Tools/Methods of Students with Logical and Artistic Abilities- Activity-4*

<i>Tool/Method</i>	Logical					Artistic				
	Excellent	V Good	Good	Average	poor	Excellent	V Good	Good	Average	poor
Array structures	0%	83%	17%	0%	0%	17%	32%	17%	17%	17%
Worked examples	11%	67%	22%	0%	0%	11%	22%	11%	34%	22%
Teacher guidance	17%	33%	17%	33%	0%	0%	0%	34%	33%	33%
Hands-on exercises	25%	25%	37%	13%	0%	12%	13%	13%	50%	12%

A question in the Mini questionnaire-4 requested the students to indicate the best combination of the tool/methods to be used in this activity. Many participants (83%) suggested all the four teaching tool/methods: array structures, worked examples, teacher guidance, and hands-on examples to be used to teach this concept. Almost all participants (97%) suggested a combination of array structures, worked examples, and hands-on examples. Teacher guidance was suggested to be used by 89% of the participants. These results shows that students' preference for a combination of visual, kinaesthetic, and cognitive aspects is the most appropriate to teach this concept (see Table 4.18).

**Table 4.18**  
*Summary of Tools/Methods Preferred by Students in Activity-4*

	Array structures	Worked examples	Teacher guidance	Hands-on exercises	All The Tools & methods	Array structures, Worked Example & hand-on work
Percentage of students	97%	97%	86%	100%	97%	83%

#### 4.3.4.2 Findings from the qualitative data

Almost all the participants (97%) thought that Activity-4 helped them understand the array concept. The participants of the survey were asked to write comments on how the activity helped them learning the array concept. Some noteworthy comments are listed below.

*Brain remembers pictures easily.*

*It is always easier when pictures are used.*

*Visual tools make the lesson easier to understand the concepts that many struggled to get.*

*Pictures helped to visualise and understand the concept.*

*Pictures help our minds to make the connection between theory and practical.*

*Pictures give a different perspective for learning.*

*Pictures visually explain the concept.*

All the participants commented positively on the use of teacher guided hands-on sessions using worked examples in learning array concept. Six out of 29 students thought that the teacher guidance provided during the interactive practical sessions using worked examples was useful to learn this concept. The rest (79%) of the

students had written comments on the usefulness of using hands-on practical sessions using worked examples. Some of such comments include:

*The interactive exercises are very useful in learning.*

*Hand-on sessions are not boring.*

*This exercise helped learning with minimal effort.*

*This exercise explains the difference between arrays of numbers and objects.*

*We could experiment and get experience by ourselves.*

*The worked examples made it easy for us to understand arrays.*

*Hands-on sessions are easier to understand.*

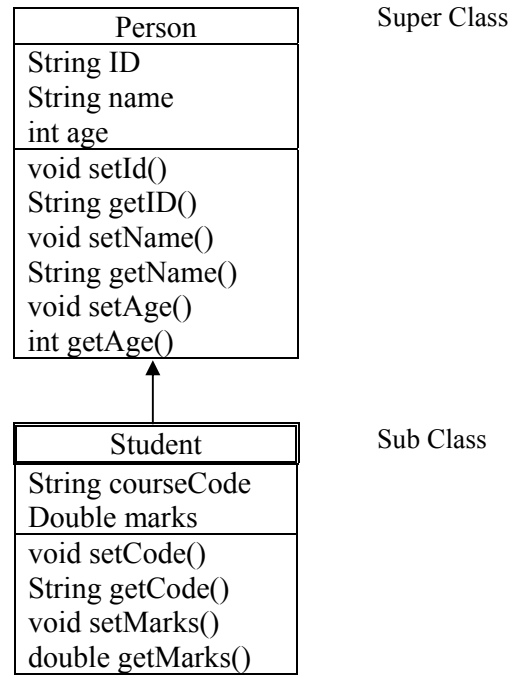
*The activity helped us to understand what was happening with code.*

These comments indicate that the majority of the students with high logical skills prefer a kinaesthetic way of learning. According to the four quadrant brain dominance model (see Figure 2.12), logical activities are in quadrant-A of the left hemisphere of the brain whereas the kinaesthetic activities are in quadrant-C of the right hemisphere of the brain.

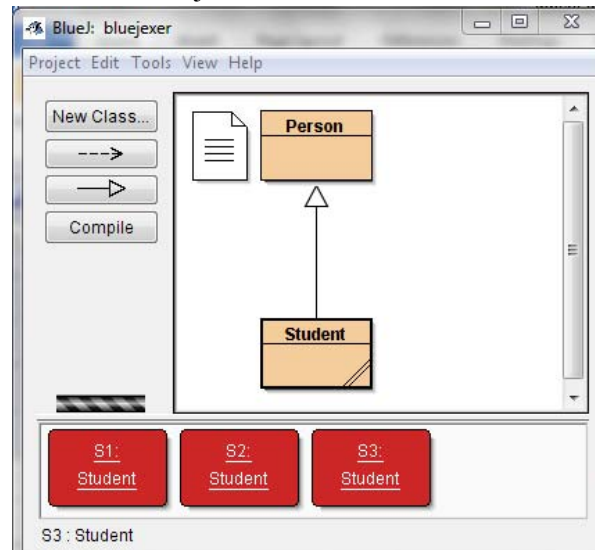
#### **4.3.5 Mini Questionnaire-5 Findings**

BlueJ is a Java program development tool that provides a graphical environment to the users. Teaching Activity-5 (see Figure 4.11) was designed and used to teach the inheritance concept using BlueJ, teacher guided exercise and teacher guidance. Mini questionnaire-5 (see Appendix G) was used to get feedback from students about the usefulness of this teaching activity. Thirty two students participated in this survey.

## Activity – 5



1. Use BlueJ to create the above two classes using inheritance.
2. Create three objects as shown below:



3. Test the three objects of the Student class using the following values.

ID	name	age	courseCode	marks
0001	Peter	23	COMP.5111	70
0002	Jane	21	COMP.5100	80
0003	John	18	COMP.5101	100

Figure 4.11. Activity – 5.

#### 4.3.5.1 Findings from the quantitative data

The participants' judgements on their artistic and logical skills were categorised and summarised as shown in the Table 4.19.

**Table 4.19**  
*Summary of Artistic and Logical Skills of Participants in Mini Questionnaire-5*

Skill Level	Artistic %	Logical %
Poor	0.0	0.0
Average	53.1	12.5
Good	34.4	28.1
Very good	6.3	25.0
Excellent	6.3	34.4

According to this summary, participants' average logical skill was found to be much higher than their artistic skill. Most participants (53%) thought that they had average artistic skills (see Table 4.19). Most (87.5%) students thought that their logical skill was good, very good or excellent.

In the Mini questionnaire-5, the participants were requested to indicate the most useful aspect of the teaching activity by choosing one of the options: BlueJ visual interface, hands-on work, and teacher guidance. A summary of the findings is shown in Table 4.20.

**Table 4.20**  
*Summary of the Most Useful Teaching Tools/Methods Used in Activity-5*

Teaching tool/method	Percentage of responses
BlueJ visual interface	46.9%
Hands-on work example	37.5%
Teacher guidance	15.6%

According to the results shown in Table 4.20, most useful aspect of this teaching activity was the use of the BlueJ visual interface (46.9%). Some students (37.5%) thought that hands-on exercise was the most useful aspect in this activity.

The majority of the students, who preferred the BlueJ visual programming tool had excellent logical skills and average artistic skills. The majority of the students who suggested teacher guidance as the most useful aspect of teaching had very good logical skills and average artistic skills. The majority of the student who nominated the hands-on exercise as the most useful had good logical skills and average artistic skills (see Table 4.21).

**Table 4.21**  
*Preference of Tools/Methods of Students with Logical and Artistic Activity-5*

<i>Tool/Method</i>	Logical					Artistic				
	Excellent	V Good	Good	Average	poor	Excellent	V Good	Good	Average	poor
BlueJ	40%	27%	13%	20%	0%	13%	13%	13%	61%	0%
Teacher guidance	20%	40%	40%	0%	0%	0%	0%	0%	100%	0%
Hands-on exercises	33%	17%	42%	8%	0%	0%	0%	25%	75%	0%

A question was asked of the students to indicate the best combination of the tools/methods to be used in this activity. The majority (61%) of the participants suggested all the techniques: BlueJ, teacher guidance, and hands-on examples to be used. Many participants (91%) suggested two techniques: BlueJ visual interface, and hands-on example. Teacher guidance was suggested to be used by 66% of the participants (see Table 4.22).

**Table 4.22**  
*Summary of Tools/Methods Preferred by Students in Activity-5*

	BlueJ	Teacher guidance	Hands-on exercise	All The Tools & methods	BlueJ & Hand-on Work
Percentage of students	94%	66%	100%	61%	91%

#### 4.3.5.2 Findings from the qualitative data

Most of the students (93%) agreed that BlueJ helped them to understand the inheritance programming concept. Although the majority of the participants of this survey have better logical skills, it is interesting to find the students' preference for the graphical environment of this tool. These results indicate that students' artistic and logical skills are not related to their preference for visual teaching tools such as BlueJ. Of the 32 participants, 28 commented on the usefulness of BlueJ to understand the inheritance concept. 16 comments were about the usefulness of the graphical aspect of BlueJ, especially to illustrate the inheritance between classes. The following are some of the important comments.

*It clearly shows the connection between classes.*

*Graphics in BlueJ is useful to understand the inheritance.*

*It helps to improve understand because it turns the text explanation into graphical.*

*It creates and displays objects visually and enables testing methods visually.*

*Visual objects give us feeling of existence of objects.*

*I think BlueJ's graphical representation of the Java code of inheritance helped me to understand this concept.*

There were some comments from students describing BlueJ as a useful tool not only for teaching but also for self-learning Java programming for beginners. Some students had enjoyed the facility in BlueJ for testing constructors and methods without writing code. Such comments include:

*By using inherited method from a super class in BlueJ, a user can understand the concept clearly.*

*It helps to understand relationships between classes better.*

*Bluej helps to test constructors and methods without writing code.*

*BlueJ is a good tool to understand inheritance for beginners.*

*BlueJ is a useful tool to help understand relationships between classes.*

*BlueJ is an ideal teaching package for self-learning.*

One of the reasons for using BlueJ was to create a mental model of the inheritance concept graphically in the learner's mind and find out the learning outcomes from their point of view. These comments indicate that mental modeling helped to understand difficult Java concepts such as inheritance.

The participants were asked if the teacher guided hands-on exercise using BlueJ was helpful to understand the inheritance concept. Most participants (97%) answered 'yes' to this question. The participants were also asked to write comments on how it could help. 30 students had written positive comments. Some students found the interactivity involved in this teaching activity useful. Such comments include:

*Varied work is involved in hand-on sessions,*

*Hand-on sessions are less monotonous*

*It is more interactive.*

The majority of students thought that hand-on exercise helped develop their understanding due to the kinaesthetic nature of the exercise. Such comments include:

*Practical is an easier way of learning.*

*Practical or hands-on is much easier for me to understand.*

*It helps because you get actually to do something instead of reading about it.*

One participant commented on the usefulness of teacher guidance in learning.

The positive comments on both visual aspects of BlueJ and hands-on exercises indicate that mental modeling using graphics and the kinaesthetic way of learning is the most suitable for teaching the inheritance concept.



#### 4.4 Performance Improvement of Students

A new way of teaching was introduced to students using Mind mapping tools, mental modeling diagrams, and worked examples in years 2009 to 2012. Table 4.23 shows statistical data of students' performance on the Introductory Java Programming course in these four years and compared with the results for 2008.

Table 4.23  
*Students Performance from 2008 to 2012*

Year	No of students	A Grade PASS	B Grade PASS	C Grade PASS	Total PASS	Total FAIL	% PASS	% PASS with A Grade
2008	43	0	3	6	9	34	21%	0%
2009	39	3	4	10	17	22	44%	8%
2010	13	2	2	4	8	5	62%	15%
2011	11	6	1	3	10	1	91%	55%
2012	9	4	1	3	8	1	89%	44%

The participants of the survey in phase one of this research had learnt introductory Java programming course in the conventional way at Waiariki Institute of Technology in year 2008. The pass rate of this group was the lowest (21%) in 2008 (see Table 4.23). The newly devised teaching tools were introduced to teach Java concepts in 2009. Despite the number of students enrolled declining from year 2008, the percentage of students who passed increased dramatically. In particular, the number of students passing with an A grade increased significantly. These results clearly indicate that the students have benefitted from the new teaching tools introduced in 2009. The decrement of student teacher ratio from 2008 to 2012 could also have been a contributing factor for this improvement.

#### 4.5 Summary

The data analysed in Phase One of this research revealed the difficulty level of most of the Java concepts from the students' point of view. The difficult concepts were found to be inheritance, use of arrays for objects and primitive variables, use of text files, parameter variables, and class-object concept.

In Phase One, the correlation between difficulty levels and students' logical skills and artistic skills were calculated and a weak correlation between difficulty level and the artistic skill of students was found.

The answers to the question “What type of learner are you?” with multiple options: Kinaesthetic, visual, and auditory, revealed that the highest percentage of students (75%) were found to be kinaesthetic learners. The visual learner population was the second highest with 55% and the lowest learner population was found to be auditory with 24%. A combination of kinaesthetic and visual learner was found to be 39% of the population.

The participants of Phase One suggested the most suitable teaching style for each concept out of the three options of teaching styles; auditory, kinaesthetic, and visual. The majority of the participants suggested the kinaesthetic way of teaching for most of the concepts. Using visual way of teaching was the second highest choice. The use of auditory was the least popular choice for all the concepts (see Table 4.2). These percentages were taken into account in designing teaching activities to combine visual, kinaesthetic, and auditory aspects of teaching in Phase Two.

Five different teaching activities were designed combining visual, kinaesthetic (hands-on), teacher guidance, and worked examples.

In Activity-1, Mind mapping was used as a visual tool. It was also combined with worked examples, hands-on work and teacher guidance to teach O-O Class concept. According to the findings of the Mini questionnaire-1 (see Appendix C), worked example was the most preferred component for the students. It was also found that students with high logical skills enjoyed the visual aspect of the Mind mapping and worked examples in learning. The combination of worked example, and hands-on example were found to be the best suited for many students.

In Activity-2, a pictorial representation of an object called Star Structures was introduced as a mental model. The creation and manipulation of objects was introduced as a hands-on activity combining visual pictures, worked example and teacher guidance. An analogy was also used to explain returning a value from a method. The feedback collected from the Mini questionnaire-2 (see Appendix D)

indicated that the students had found picture/analogy, worked example, and hands-on application equally important to understand this concept. The students with good or very good logical skills found hands-on work the most useful in this activity. As in Activity-1, the preferred combination of learning was analogy/picture, worked example, and hands-on example.

Activity-3 was the simplest activity in which visual analogy combined with hands-on work and teacher guidance was used to explain the concept of parameter passing and returning a value in a method. According to the summarised data collected using Mini questionnaire-3 (see Appendix E), the visual analogy was found to be the most useful part of the activity for students. This analogy was suggested as useful by the majority of students with high logical and artistic skills. Almost all preferred a combination of analogy and hands-on exercise to learn this concept.

Activity-4 comprised visual array structure, worked examples, hands-on work, and teacher guidance. This activity was designed and used to explore the possibility of improving the quality of teaching the array concept using primitive variables and also array of objects. The Mini questionnaire-4 (see Appendix F) was used to collect feedback from students. According to the summarised data collected, 97% of the students expressed usefulness of the activity to understand the concept with various comments added. The worked example used was found to be the most useful for many and the combination of hands-on session, worked example, and the pictorial array structure was the choice of 97% of the participants. Some students did not find teacher guidance as useful as the other three aspects of this activity. This result indicates that teaching activities combined with kinaesthetic and visual work well together in teaching the array concept.

BlueJ was used in the fifth activity as a visual tool with the intention of creating a mental model and also to reduce the intrinsic cognitive load of the learner's working memory. As in the other activities, hands-on activity and teacher guidance were incorporated in the activity. Data collected using the Mini questionnaire-5 (see Appendix G) clearly exposed the effectiveness of this tool with 93% of the students supporting it. The hands-on work and the BlueJ visual interface were enjoyed by students more than the teacher guidance provided to them in the teaching activity.

This result once again revealed the success of teaching Java concepts by using activities combined with kinaesthetic and visual aspects.

Since the introduction of new teaching activities, students' performance has improved over the last three years resulting in the achievement of higher grades. This result clearly support the success of teaching activities designed in accordance with the principles of Cognitive Load theory (CLT) with properly managed intrinsic, germane, and extraneous cognitive loads. The use of analogies and pictures was also found to be effective in applying mental modeling concepts to teach Java programming.

## **CHAPTER 5**

### **CONCLUSION**

#### **5.1 Insights**

One of the aims of this investigation was to identify the difficult Java concepts which must necessarily be addressed in teaching Java programming at an introductory level. Such core Java concepts were found to be inheritance, the use of arrays, the class and object concepts, and the use of text files.

A second basic aim of the investigation was to determine the most desirable way of learning each concept from the students' point of view. According to the findings, a kinaesthetic approach to learning was found to be the most suitable learning method for all Java concepts experimented with in this research. The second best method for learning Java concepts was found to be a visual approach. The least desired method of learning was the auditory approach. A final and related research question in Phase One was to discover the preferred combination of learning styles for teaching Java concepts, and this was found to be kinaesthetic and visual.

In order to design teaching activities for the second phase, the findings of the percentages of kinaesthetic, visual, and auditory types of learners within the student sample provided vital evidence. Findings in the first phase revealed that more than 50% of the student population is either kinaesthetic or visual. Therefore, 50% to 75% of visual and kinaesthetic components of teaching could desirably be included in activities. The auditory component of teaching could be limited to 25% to 50% in activities. According to the findings of Fowler, et al. (2000), 70%-83% of learners are visual, however this percentage is slightly higher than the findings of this survey, and the variance could be attributed to the different Java programming environments used in the two research projects or to the types of learners who enroll in these courses. This research employed Dr. Java, a non-visual programming environment, whereas Fowler, et al. (2000) utilised students from a RAPTOR visual Java programming environment.

There is also a general belief that people with logical abilities are better at learning computer programming, whereas those with a more artistic sensibility are not as good, and this was investigated as a research question in the Java programming environment. The correlation coefficient between students' artistic ability and their total level of difficulty with Java concepts was found to be 0.24 ( $p = 0.19$ ), showing a weak positive correlation between artistic abilities and total difficulty level. There was no correlation found between total difficulty level with Java concepts and logical ability of students. These findings give some support to the general belief that people with logical abilities are better at learning computer programming. Not surprisingly, the students who opted to do this course considered that they have much better logical skills than artistic skills. This can be clearly seen in the summary of skills of the students who participated in the five mini questionnaires. Although Hadjerrouit (1998) argued that mathematical and logical abilities were not essential for computer programming students, these findings corroborate the fact that those who are lacking mathematical and logical skills find most Java concepts difficult. However, as noted by Hadjerrouit (1998), it cannot be denied that some authors of programming books employ mathematical examples and exercises, and hence logical and mathematical abilities have become a requirement for learners who use those texts.

One of the research questions targeted in Phase Two of this investigation was to study the relationship between students' learning preference and their logical and artistic hemispheric dominance. The summary of five activities in the second phase clearly shows that those learners who claimed to have good or very good logical skills preferred visual aspects of learning such as analogies and pictorial mental models. In addition, a combination of cognitive aspects and kinaesthetic aspects of learning was preferred by many students in all five activities utilised in Phase Two. According to the four quadrant brain dominance model (Figure 2.12), logical activities are in quadrant-A, visual activities are in quadrant-D, and kinaesthetic activities are in quadrant-C. In all five activities, it was apparent that students with logical skills prefer visual and kinaesthetic aspects. These results indicate that the current Java programming students' preference has changed and shifted towards quadrant-D and quadrant-C. This result agrees with the findings of Lumsdaine and Lumsdaine (1995), who predicted a paradigm shift from quadrant-A to quadrant-D in

the 1990s. This trend was described by Reigeluth (1996) as a shifting paradigm, moving from behaviourism to constructivism. In addition, Sung (2010) suggested a 21st century paradigm shift towards active learning with group collaboration, using integrated teaching models. The findings of the research suggest that learning programming is a cognitive issue that is best approached by supportive pedagogical methods.

The final question which was investigated within Phase Two was to evaluate the effectiveness of teaching tools based on a combination of Cognitive Load Theory (CLT), and the concepts of mental modeling and scaffolding in learning Java concepts. A combination of Mind map, worked example and hands-on exercise was found to be the most beneficial for many students to understand the class concept in Java programming. Of the three components employed in this activity, students found the worked example the most useful. The BlueJ software tool and the hands-on worked example were found to be beneficial for the majority of students with higher logical skills in learning the inheritance concept. As suggested by Sweller (2010), worked or partially completed examples reduce intrinsic cognitive load as they alter knowledge levels. These results indicate that the cognitive aspects of learning are the most important in teaching Java concepts.

In addition, mental modeling was also experimented with using low cost teaching tools. The consistent use of a star structure in many activities within this course was found to be beneficial for students, assisting them to visualise the concept of an object. In addition, the use of analogy in teaching was found to be valuable for many students. Moreover, employing pictures and analogies for mental modeling and schema building was experimented with, as suggested by Paivio (2006) and Van Gog et al. (2010). The results indicate that schema theory could be used effectively in knowledge building in teaching Java programming language.

Gallimore and Tharp (1990) identify teacher guidance, mental modeling, and cognitive structuring as different categories of scaffolding. In most activities used in this research, teacher guidance was not found to be highly important for the students. This could be due to sufficient scaffolding being provided using worked examples and mental modeling, such that many students could carry out the hands-on activities on their own without requiring teacher support. Some comments by students

indicated that they wished to do some work on their own right from the beginning. When compared to previous experience, this suggests that learning was becoming comparatively easier due to the superior tools and methods being employed in teaching.

## **5.2 Significance**

The findings as to the most difficult Java programming concepts should be beneficial for most educators who teach Java programming. This research especially explored the use of worked examples as a way of minimising the cognitive loads of working memory. These examples were found to be very effective for many students in learning Java concepts. The two software tools utilised, BlueJ, and Mind mapping, were freeware and were found to be valuable for students to understand the Java class, object and inheritance concepts. The other tools used were pictures, such as Star Structure (see Figure 3.7), and analogies to impart knowledge of Java concepts. The pictures and analogies employed as scaffolding along with worked examples were preferred by students. All of the tools utilised were low cost and affordable by all teachers of Java programming.

The findings of this research agree with the recently developed Dual Code Theory (DCT), according to which visual and verbal information are handled by different channels of the brain (Paivio, 2006). In this research, teaching activities were profitably combined with visual, verbal, and kinaesthetic aspects of teaching, taking students' learning type into consideration. In addition, this research bore out perceived benefits of mental modeling aspects of teaching. These findings could be useful for researchers who are involved in studies of cognition and working memories.

## **5.3 Limitations**

There were several limitations which compromised the accuracy of the findings of this research. In phase two of the research, the teaching activities were experimented with in the classroom environment and the students who attended classes were the participants in the surveys. Unfortunately, the number of students enrolled in the Java programming course has drastically declined since 2010. For this reason, it was difficult to find sufficient students in one class to validate the survey as random



sampling was not possible. As a result, the research had to be carried out in three consecutive years using convenience data sampling of different groups. Although this research addressed the most difficult areas identified in phase one, a single area was neglected. The use of text files was found to be difficult for many students, and while the issue was not addressed using the newly devised teaching techniques with graphics or analogies, worked examples and teacher guidance were employed in teaching this concept. However, due to the reluctance of students to participate in too many surveys, the success of this teaching activity was not evaluated using a questionnaire.

Despite lengthy personality tests available to find the logical and artistic skills of people more accurately, this research was dependent on the participant's judgement of their artistic and logical skills. This would undoubtedly have had impact on the accuracy of the percentages given for the skills of the participants. It was practically impossible to incorporate lengthy personality tests in the surveys conducted in the research, with the inevitable result that accuracy had to be compromised.

#### **5.4 Implications for Future Research**

According to the findings of the second phase of this research, in all the activities, teacher guidance was not considered as useful as other aspects of teaching by many students. Indeed, some students requested more activities without worked examples. It could be possible that the activities were made too simplistic for them and as a result, they found them to be not challenging enough. Another possibility is that once a concept is understood, students want to try it in programs without further interruption. There could be a further investigation, targeting average students in a Java class, to find out how easy tasks should be made.

The students who enrolled in the Java class were found to be good in the logical rather than the artistic domain, and these findings were consistent for the three consecutive years. However, as noted above, the evaluation of logical and artistic skills was based on the learners' own judgements. A truer estimate of these skills is another area to be investigated further. It is also worth examining the distribution of logical and artistic skills among students using more accurate personality tests with a random data sample.

## **5.5 Recommendations**

The findings of this research suggest that teaching Java programming is a cognitive issue rather than a pedagogical one. The most important aspect in teaching is to eliminate extraneous cognitive load by avoiding teaching anything unrelated to the concept being focussed on. The teaching activities utilised to communicate difficult concepts must be designed in such a way that visual, kinaesthetic and auditory aspects of teaching are combined. In addition, it is vital to use worked and/or partly completed examples to reduce intrinsic cognitive load to tolerable levels for the learners.

Mental modeling using pictures and analogies is also a very powerful technique to help with learning Java concepts. Analogies used should be simple and familiar to most students and the pictures employed need be consistent throughout the course to create a supportive mental model for the concepts being taught. Low cost teaching tools such as mind mapping, BlueJ, pictures and pictorial analogies, can be utilised effectively to teach difficult Java programming concepts more easily.

## **5.6 Final Comment**

The selection of a set of suitable teaching tools/methods to teach a concept is usually challenging and could vary with the nature of the concept. Although analogies were found to be very effective, it is often hard to find a suitable analogy for each concept. However, a good mixture of low cost tools, pictures, worked examples, and hands on exercises could readily be utilised to make teaching of any difficult Java concept more straightforward.

## REFERENCES

- Allen, E., Cartwright R., & Stoler, B. (2002). Dr Java: A lightweight pedagogic environment for Java. *ACM SIGCSE Bulletin*, 34(1), 137-141.
- Amckinn. (2009). *Teaching multiple learning styles with R2D2*. Retrieved from <https://apps.lis.illinois.edu/wiki/display/wise/Teaching+Multiple+Learning+Styles+with+R2D2>
- Anderson, J. R. (1993). *Rules of the mind*. Hillsdale, NJ: Erlbaum.
- Arbib, M. A. (1992), Schema Theory, In S. Shapiro (Ed.), *The encyclopedia of artificial intelligence* (2nd ed.). (pp. 1427-1443). NY: Wiley.
- Ashcraft, M., & Radvansky, G. A. (2010). *Cognition* (5th ed.). Boston, MA: Pearson Education.
- Atherton, J. (2010). *Bloom's taxonomy*. Retrieved from <http://www.learningandteaching.info/learning/bloomtax.htm>
- Atkinson, R. C., & Shiffrin, R. M. (1968). Human memory: A proposed system and its control processes. In K.W. Spence (Ed.), *The Psychology of learning and motivation: Advances in research and theory*. (vol. 2, pp. 89–195). NY: Academic Press.
- Baddeley, A. D. (1986). *Working memory*. Oxford, United Kingdom: Oxford University Press.
- Baddeley, A. D. (2001). Is working memory still working? *American Psychologist*, 56, 851-864.
- Baddeley, A. D., & Hitch, G. J. (1974). Working memory. In G.A. Bower (Ed.), *Recent advances in learning and motivation* (vol. 8, pp. 47–89). New York, NY: Academic Press.

- Barland, I. (2008). Some methods for teaching functions first using java. *In Proceedings of the 46th Annual Southeast Regional Conference*. Auburn, USA.
- Beckmann, J. (2010). Taming a beast of burden: On some issues with the conceptualisation and operationalisation of cognitive load. *Learning and Instruction*, 20, 250-264.
- Ben-Ari, M. (2001a). Constructivism in Computer Science Education, *Journal of Computers in Mathematics & Science Teaching*, 20(1), 45-73.
- Ben-Ari, M. (2001b). Program visualisation in theory and practice. *Upgrade*, 11(2), 8-11.
- Berk, L. E. (2003). *Child development* (6th ed.). Boston, MA: Allyn and Bacon.
- Bloch, S. (2009). Teach scheme, reach Java: Introducing object-oriented programming without drowning in syntax, *Journal of Computing Sciences in Colleges*, 24(6), 12-14.
- Bonk, C. J., & Zhang, K. (2008). *Empowering online learning: 100+ activities for reading, reflecting, displaying and doing*. San Francisco, CA: Jossey-Bass.
- Brown, J. S., Collins, A., & Duguid, P. (1989). Situated cognition and the culture of learning. *Educational Researcher*, 18(1), 32-42.
- Brusilovsky, P., Calabrese, E., Hvorecky, Y., Kouchnirenko, A., & Miller, P. (1997). Minilanguages: A way to learn programming principles. *Education and Information Technologies*, 2(1), 65-83.
- Burton, P. J., & Bruhn, R. E. (2003). Teaching programming in the OOP era. *SIGCSE Bulletin*, 35(2), 111-114.

- Butler, M., & Morgan, M. (2007). The learning challenges faced by novice programming students studying high level and low feedback concepts. *In Proceedings ASCILITE Singapore 2007*. Singapore.
- Buzan, T. (1991). *Use both sides of your brain*. London, United Kingdom :Penguin Books.
- Buzan, T. (2011). *What is a Mind Map?* Retrieved from <http://www.tonybuzan.com/about/mind-mapping/>
- Buzan, T., & Buzan, B. (2006). *The mind map book*. London, United Kingdom : BBC
- Cantù, M. (2007). *Microsoft Retires Visual Jsharp*. Retrieved from [http://blog.marcocantu.com/blog/retired\\_jsharp.html](http://blog.marcocantu.com/blog/retired_jsharp.html)
- Carlisle, M. C. ( 2009). RAPTOR: A visual programming environment for teaching object- oriented programming. *Journal of Computing Sciences in Colleges, Consortium for Computing Sciences in Colleges*, 24(4), 275-281.
- Carlson, R. (2007) *What is a mental model?* Retrieved from [http://www.nationalministries.org/missional\\_church/docs/MCT\\_Mental\\_Model.pdf](http://www.nationalministries.org/missional_church/docs/MCT_Mental_Model.pdf)
- Cartner, H., & Hallas, J. (2009). Exploring the R2D2 model for online learning activities to teach academic language skills. *Proceedings of the 26th Annual ascilite International Conference*, Auckland, New Zealand.
- Caspersen, M. E., & Bennedsen, J. (2007). Instructional design of a programming course: A learning theoretic approach. *Proceedings of the Third International Workshop on Computing Education Research*, Atlanta, GA.

- Chandler, P., & Sweller, J. (1991). Cognitive load theory and the format of instruction. *Cognition and Instruction*, 8(4), 293- 332.
- Chen, Z., & Marx, C. (2005). Experiences with Eclipse IDE in programming courses. *JCSC*, 21(2), 104-112
- Chen, C. J., & Toh, S. C. (2005). A feasible constructivist instructional development model for virtual reality (VR)-based learning environments: Its efficacy in the novice car driver instruction of Malaysia. *ETR&D*, 53(1), 111-123.
- Cherry, K. (2011). *Background and key concepts of Piaget's theory*. Retrieved from <http://psychology.about.com/od/piagetstheory/a/keyconcepts.htm>
- Churches, A. (2009). *Bloom's digital taxonomy*. Retrieved from <http://edorigami.wikispaces.com/file/view/bloom%27s+Digital+taxonomy+v3.01.pdf>
- Chwif, L., & Barretto, M. R. P. (2003). Simulation models as an aid for the teaching and learning processing operations management. *In Proceedings of 2003 Winter Simulation Conference*, New Orleans, LA.
- Clark, D., MacNish, C., & Royle, G. F. (1998). Java as a teaching language opportunities, pitfalls and solutions. *Proceedings of the 3rd Australasian conference on Computer science education* (pp. 173-79). University of Queensland, Brisbane, Australia: SIGCSE.
- Clark, R., Nguyen, F., & Sweller, J. (2006). *Efficiency in learning: Evidence-based guidelines to manage cognitive load*. San Francisco, CA: Pfeiffer.
- Cognitive apprenticeship. (n.d). Retrieved from <http://www.edtech.vt.edu/edtech/id/models/cog.html>

- Collins, D. (2002). Java second. The suitability of Java as a first programming language. *Proceedings of the Sixth Java & the Internet in the Computer Curriculum Conference*, North London, United Kingdom.
- Collins, A., Brown, J. S., & Holum, A. (1991). Cognitive apprenticeship: Making thinking visible. *American Educator*, 15(3), 6-11, 38-46.
- Collins, A., Brown, J. S., & Newman, S. E. (1989). Cognitive apprenticeship: Teaching the crafts of reading, writing, and mathematics. In L. B. Resnick (Ed.), *Knowing, learning, and instruction: Essays in honor of Robert Glaser* (pp. 453-494). Hillsdale, NJ: Lawrence Erlbaum.
- Cooper, G. (1998). *Research into Cognitive load Theory and Instructional Design at UNSW*. Retrieved from <http://dwb4.unl.edu/Diss/Cooper/UNSW.htm>
- Culatta , R. (2011). *ADDIE Model*. Retrieved from <http://www.instructionaldesign.org/models/addie.html>
- Culwin, F. (1999). Object Imperatives! *Proceedings of the 30th Technical Symposium on Computer Science Education (SIGCSE)*, New Orleans, LA.
- Daneman, M., & Carpenter, P. A. (1980). Individual differences in working memory and reading. *Journal of Verbal learning and Verbal Behaviour*, 19(4), 450-466.
- Daneman, M., & Tardif, T. (1987). Working memory and reading skill re-examined. In M. Coltheart (Ed.), *Attention and performance*, 7, ( pp. 491-508). London: Erlbaum.
- Dann, W., Cooper, S., & Pausch, R. (2001). Using Visualization to Teach Novices Recursion, *Proceedings of the 6th ACM SIGCSE Conference on Innovation*

*and Technology in Computer Science Education* (pp109–112). Canterbury, United Kingdom.

Davis, L. (2005). MindMapX. *ACM SIGCSE Bulletin*, 37(2), 405.

Dennen, V. P. (2008). Intersecting communities of practice: Merging roles across the academic and blogging worlds. *Proceedings of the e-Society 2008 Conference*, Algarve, Portugal.

Dennen, V. P., & Burner, K. J. (2007). The cognitive apprenticeship model in educational practice. In J. M. Spector, M. D. Merrill, J. Van Merriënboer & M. P. Driscoll (Eds.), *Handbook of research on educational communications and technology* (3rd ed., 425-439). Mahwah, NJ: Erlbaum.

Driscoll, M. P. (2000). *Psychology of Learning for Instruction* (2nd ed.) Boston, MA: Allyn & Bacon.

Dunbar, K. (1997). How scientists think: Online creativity and conceptual change in science. In T. B. Ward, S. M. Smith, & S. Vaid (Eds.), *Conceptual structures and processes: Emergence, discovery and change*. Washington, DC: APA Press.

Embarcadero customer support. (2010). Retrieved from <http://support.embarcadero.com/article/37871>

Felder, R. M. & Silverman, L. K. (1988). Learning and Teaching Styles in Engineering Education, *Engineering Education*, 78(7), 674-681.

Felder, R. M., & Soloman, B. A. (1997). *Index of Learning Styles questionnaire*. Retrieved from <http://www.engr.ncsu.edu/learningstyles/ilsweb.html>



- Ferrari, A., Garbervetsky, D., Braberman, V., Listingart, P., & Yovine, S. (2005). *JScoper: Eclipse support for research on scoping and instrumentation for real time Java applications*. Retrieved from [www-2.dc.uba.ar/exclusivos/garbervetsky/thesis/BFG\\_etx2005.pdf](http://www-2.dc.uba.ar/exclusivos/garbervetsky/thesis/BFG_etx2005.pdf)
- Ferrero, D. J. (2005). Pathways to reform: Start with values. *Educational Leadership*, 62(5), 8-14.
- Flavell, J. H. (1976). Metacognitive aspects of problem solving. In Resnick (Ed.), *The nature of intelligence*. New Jersey, NJ: Lawrence Erlbaum Associates.
- Fowler, L., Allen, M., Armarego, J., & Mackenzie, J. (2000). Learning styles and CASE tools in software engineering. *Proceedings of the 9th Annual Teaching Learning Forum*, Perth, Australia.
- FreeMind – free mind mapping software. (2011). Retrieved from [http://freemind.sourceforge.net/wiki/index.php/Main\\_Page](http://freemind.sourceforge.net/wiki/index.php/Main_Page)
- Gallimore, R., & Tharp, R. (1990). Teaching mind in society: Teaching, schooling, and literate discourse. In L. C. Moll (Ed.), *Vygotsky and education: instructional implications and applications of sociohistorical psychology*, (pp. 175–205). Cambridge, United Kingdom: Cambridge University Press.
- Garner, S. K. (2002). Reducing the cognitive load on novice programmers. In P. Barker & S. Rebelsky (Eds.), *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications*. Chesapeake, VA: AACE.
- Garner, S. K. (2007). An exploration of how a technology-facilitated part-complete solution method supports the learning of computer programming. *Journal of Issues in Informing Science and Information Technology*, 4, 491-501.

- Garner, S. K. (2009). A quantitative study of a software tool that supports a part-complete solution method on learning outcomes. *Journal of Information Technology Education*, 8, 285-310.
- Garner, S., Haden, P., & Robins, A. (2005). My program is correct but it doesn't run: A preliminary investigation of novice programmers' problems. In: ACE 2005. *Proceedings of the 7th Australasian Conference on Computing Education*, Newcastle, Australia.
- Georgantaki, S., & Retalis, S. (2007) Using educational tools for teaching Object Oriented design and programming, *Journal of Information Technology Impact*, 7(2), 111-130.
- Graf, S., Viola, S. R., Kinshuk, & Leo, T. (2007). In-depth A\analysis of the Felder-Silverman learning style dimensions. *Journal of Research on Technology in Education*, 40(1), 79-93.
- Greenaway, R. (2011). *Experiential learning articles and critiques of David Kolb's theory*. Retrieved from <http://reviewing.co.uk/research/experiential.learning.htm#ixzz1dXRYN3v7>.
- Grey, D. & Miles, R. (2002). Teaching Java objectively: Reflections on a web-based java course. *Proceedings of the Sixth Java & the Internet in the Computer Curriculum Conference*, North London, United Kingdom.
- Hadjerrouit, S. (2007). A blend learning model in Java programming: A design-based research approach. *Proceedings of the Computer Science and IT Education Conference 2007*, Mauritius.
- Hadjerrouit, S. (1998). Java as first programming language: A critical evaluation, *ACM SIGCSE Bulletin*. 30(2), 43-47.

- Hadjerrouit, S. (1999). A constructivist approach to object-oriented design and programming. *Proceedings of the 4th Annual Conference on ITICSE*, Cracow, Poland.
- Hagan, D., & Markham, S. (2000). Teaching Java with the BlueJ environment. *Proceedings of Australasian Society for Computers in Learning in Tertiary Education Conference ASCILITE 2000*, Coffs Harbour, Australia.
- HBDI. (2001). Retrieved from <http://cemmgroup.com/hbdi.pdf>
- Herrington, J., & Oliver, R. (1995). Critical characteristics of situated learning: implications for the instructional design of multimedia. In Ellis, A., & Pearce, J. (Eds.), *ASCILITE95 Conference Proceedings* (pp. 253-262). Melbourne, Australia: University of Melbourne.
- Holtel, S. (2005). *Utilizing mind maps for essential use case specification*. Retrieved from <http://roots.dnd.no/modules.php?op=modload&name=Downloads&file=index&req=getit&lid=164>
- Horstmann, C. S. (2005). *Java concepts* (4<sup>th</sup> ed.). NY:Wiley.
- Howard, R. A., Carver, C. A., & Lane, W. D. (1996). Felder's learning styles, Bloom's taxonomy, and the Kolb learning cycle: Tying it all together in the CS2 course. *SIGCSE Bulletin*, 28(1), 227-231.
- Huitt, W. & Hummel, J. (1999). *Educational psychology* [Powerpoint slides]. Retrieved from <http://www.edpsycinteractive.org/edpsyppt/Theory/behthr.ppt>
- Introducing Visual Studio .NET. (2011). Retrieved from [http://msdn.microsoft.com/en-us/library/fx6bk1f4\(v=vs.71\).aspx](http://msdn.microsoft.com/en-us/library/fx6bk1f4(v=vs.71).aspx)

- Introduction to Kawa 3.13. (n.d). Retrieved from  
[http://itee.uq.edu.au/~comp1500/\\_CoreDoc/KawaRead.pdf](http://itee.uq.edu.au/~comp1500/_CoreDoc/KawaRead.pdf)
- Jakovljevic, M. (2003). Concept mapping and appropriate instructional strategies in promoting programming skills of holistic learners. *Proceedings of the 2003 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on Enablement through Technology*, South Africa.
- James, W. (1890). *The principles of psychology*. Retrieved from  
<http://psychclassics.yorku.ca/James/Principles/prin16.htm>
- Jonassen, D. H., Beissner, K., & Yacci, M. A. (1993). *Structural knowledge: Techniques for representing, assessing, and acquiring structural knowledge*. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Jonassen, D., Carr, C., & Yueh, H. (1998). *Computers as mindtools for engaging learners in critical thinking*. Retrieved from  
[http://www.siue.edu/education/techready/5\\_Software\\_Tutorials/5\\_AncillaryPages/Mindtools.pdf](http://www.siue.edu/education/techready/5_Software_Tutorials/5_AncillaryPages/Mindtools.pdf).
- Johnson-Laird, P. N., (1983). *Mental models: Towards a cognitive science of language, inference and consciousness*. Cambridge, United Kingdom: Cambridge University Press.
- Johnson-Laird, P. N., Girotto, V., & Legrenzi, P. (1998). *Mental models: A gentle guide for outsiders*. Retrieved from  
<http://icos.groups.si.umich.edu/gentleintro.html>
- Just, M. A., & Carpenter, P. A. (1992). A capacity theory of comprehension: Individual differences in working memory. *Psychological Review*, 99, 122-149.

- Kalyuga, S. (2006). *Instructing and testing advanced learners: A cognitive load approach*. New York, NY: Nova Science.
- Kannangara, D. (2007). Transition from C++ to JAVA : Use of action research method to improve students' learning. *Proceedings of the Teaching and Learning Conference, Eastern Institute of Technology* (p.131). Napier, New Zealand: AKO.
- Karahasanović, A., & Thomas, R. C. (2007). Difficulties experienced by students in maintaining Object-Oriented systems: An empirical study. In *The 9th Australasian Computing Education Conference (ACE2007)*, Victoria, Australia.
- Kieras, D. E., Meyer, D. E., Mueller, S., & Seymour, T. (1999). Insights into working memory from the perspective of the EPIC architecture for modeling skilled perceptual-motor and cognitive human performance. In A. Miyake & P. Shah (Eds.), *Models of working memory: Mechanisms of active maintenance and executive control* (pp. 183–223). New York, NY: Cambridge University Press.
- Kirschner, P. A., & Erkens, G. (2006). Cognitive tools and mindtools for collaborative learning. *Journal of Educational Computing Research*, 35(2), 199-209.
- Koedinger, K. R., & Aleven, V. (2007). Exploring the assistance dilemma in experiments with cognitive tutors. *Educational Psychology Review*, 19, 239-264.
- Kolb, A. Y., & Kolb, D. A. (2005). *The Kolb learning style inventory-version 3.1 2005 technical specifications*. Retrieved from <http://www.whitewater-rescue.com/support/pagepics/lstechmanual.pdf>

- Kolb, D. A. (1984). *Experiential learning: experience as the source of learning and development*. Englewood Cliffs, NJ: Prentice-Hall.
- Kolb, D. A., Boyatzis, R. E. & Mainemelis, C. (1999). *Experiential learning theory: Previous research and new directions*. Retrieved from <http://www.d.umn.edu/~kgilbert/educ5165-731/Readings/experiential-learning-theory.pdf>
- Kölling, M., Quig, B., Patterson, A., & Rosenberg, J. (2003). The BlueJ system and its pedagogy [Special Issue]. *Journal of Computer Science Education*, 13(4).
- Kölling, M., & Rosenberg, J. (1996). An Object-Oriented program development environment for the first programming course. *SIGSE Bulletin*, 28(1), 83-87.
- Kouznetsova, S. (2007). Using BlueJ and Blackjack to teach object-oriented design concepts in CS1. *Journal of Computing Sciences in Colleges*, 22(4), 49-55.
- Kovacic, Z., Green, J., & Eves, C. (2004). Learning styles of computer concepts students in a distance tertiary institution. *Proceedings of the NZARE National Conference 2004*, Wellington, New Zealand.
- Koznov, D., & Pliskin, M. (2008). Computer supported collaborative learning with mind-maps. In T. Margaria and B. Steffen (Eds.), *Proceedings of 3rd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation*, Porto Sani, Greece.
- Krajcik, J. S., Czerniak, C. M., & Berger, C. F. (2003). *Teaching science in elementary and middle school classrooms: A project-based approach*. New York, NY: McGraw-Hill.
- Kyllonen, P. C. (2002). Knowledge, speed, strategies, or working memory capacity? A systems perspective. In R. J. Sternberg & E. L. Gigorenko (Eds.), *The*

*general factor of intelligence: How general is it?* (pp. 415–445). Mahwah, NJ: Erlbaum.

Lappo, P. (2002). No pain, no XP: Observations on teaching and mentoring extreme programming to university students. *Proceedings of the 3rd International Conference on eXtreme Programming and Agile Processes in Software Engineering*, Alghero, Italy.

Larson, L., Miller, T., & Ribble, M. (2009). Five considerations for digital age leaders: What principals and district administrators need to know about tech integration today. *Learning and Leading with Technology*, 37(4), 12-15.

Lave, J. & Wenger, E. (1991). *Situated learning: Legitimate peripheral participation*. Cambridge, United Kingdom: Cambridge University Press.

Learning theories knowledge base. (2011). Situated learning Theory (Lave) at Learning-Theories.com. Retrieved from <http://www.learning-theories.com/situated-learning-theory-lave.html>

Lieberman, H. (1986). An example based environment for beginning programmers. *Instructional Science*, 14(3), 277-292.

Lipscomb, L., Swanson, J., & West, A. (2008) *Scaffolding*. Retrieved from <http://projects.coe.uga.edu/epltt/index.php?title=Scaffolding>

Lister, R., Berglund, A., Clear, T., Bergin, J., Garvin-Doxas, K., Hanks, B., Whalley, J. (2006). Research perspectives on the objects-early debate. *SIGCSE Bulletin Inroads*, 38(4), 173-192.

Litzinger, T. A., Lee, S. H., Wise, J. C., & Felder, R. M. (2007). A psychometric study of the index of learning styles. *Journal of Engineering Education*, 96(4), 309-319.

- Liu, S. (2011). *Knowledge representation*. Retrieved from <http://wenku.baidu.com/view/db5ab4ff04a1b0717fd5ddbd.html>
- Logie, R. H. (1999). Working memory. *The Psychologist*, 12(4), 174-178.
- Lui, A., Kwan, R., Poon, M., & Cheung, Y. (2004). Saving weak programming students: Applying con-structivism in a first programming course. *Association of Computing Machinery SIGCSE Bulletin*, 36(2), 72-76.
- Lumsdaine, M., & Lumsdaine, E. (1995). Thinking preferences of engineering students: Implications for curriculum restructuring. *Journal of Engineering Education*, 84(2), 193-204.
- Lunney, T. F., McCullagh, P. J., & Lundy, P. J. (2003). Java as the lingua franca for teaching graduate students. *Proceedings of Second International Conference on the Principles and Practice of Programming in Java*, Kilkenny City, Ireland.
- Lynch, P. (2011). *Atkinson-Shiffrin model*. Retrieved from <http://www.experiment-resources.com/atkinson-shiffrin-model.html>
- Ma, L., Ferguson, J., Roper, M., & Wood, M. (2007). Investigating the viability of mental models held by novice programmers. *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education*, Covington, Georgia, 499-503.
- Machanick, P. (2007). Teaching Java Backwards. *Computers & Education*, 48(3), 396-408.
- MacNaughton, G. (2003). *Shaping early childhood. Learners curriculum and contexts*. Berkshire, England: Open University Press.



- Martin, C. J. (2007). *Scribbles: An exploratory study of sketch based support for early collaborative object oriented design. 12th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, Dundee, Scotland.
- Mathematics and Computer Programming. (2003) Retrieved from <https://nrich.maths.org/discus/messages/2069/6386.html?1057677366>
- McCluskey, J. J., & Parish, T. S. (1993). A comparative study of cognitive skills in learning hypercard by right-brain dominant, left-brain dominant, and mixed brain dominant students. *Education*, 113(4), 553-555.
- McKinney, A. (2009). *Teaching multiple learning styles*. Retrieved from <http://introductiononlinepedagogy.pbworks.com/w/page/20123560/Teaching%20Multiple%20Learning%20Styles>
- McLeod, S. A. (2007). Multi store model of memory. Retrieved from <http://www.simplypsychology.org/multi-store.html>
- Mead, J., Gray, S., Hamer, J., James, R., Sorva, J., St.Clair, C., & Thomas, L. (2006). A cognitive approach to identifying measurable milestones for programming skill acquisition. *Proceedings of ITiCSE 2006*, Bologna, Italy, 182-194.
- Meyer, D. E. (2011). *The brain, cognition, and action laboratory: EPIC*. Retrieved from <http://www.umich.edu/~bcalab/epic.html>
- Millen, D., Schriefer, A., Lehder, D., & Dray, S., (1997). *Mind maps and casual models: Using graphical representations of field of research*. Retrieved from <http://www.sigchi.org/chi97/proceedings/poster/mil.htm>

- Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63, 81-97.
- Moyer, D. (2010). *Integrated development environments (IDEs): Present and future functionality*. Retrieved from <http://91-527-f2010.wiki.uml.edu/file/view/IDE+midterm+r2.pdf>
- Muller, O. (2005). Pattern oriented instruction and the enhancement of analogical reasoning. *ICER 2005: Proceedings of the 2005 International Workshop on Computing Education Research* (pp. 57-67). New York, NY: ACM Press.
- Newell, A., & Simon, H. A. (1972). *Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall.
- Norman, D. A. (1983). Some observation of mental models. In D. Gentner & A. L. Stevens (Eds.), *Mental model*. Hillsdale, NJ: Erlbaum
- Norman, D. A. (1990). *The design of everyday things*. New York, NY: Doubleday Currency.
- Norton, A. (1997). Object interfaces. In *Proceedings of the Twenty-Second Annual SAS Users Group International Conference*. Cary, NC.
- Olan, M. (2004). Dr. j vs. the bird: java ide's one-on-one. *Journal of Computing Sciences in Colleges*, 19(5), 44-52.
- Overbaugh, R. C., & Schultz, L. (n.d.). *Bloom's taxonomy*. Retrieved from [http://www.odu.edu/educ/roverbau/Bloom/blooms\\_taxonomy.htm](http://www.odu.edu/educ/roverbau/Bloom/blooms_taxonomy.htm)
- Paas, F., Renkl, A., & Sweller, J. (Eds.). (2003). Introduction: Cognitive load theory and instructional design: Recent developments [Special Section], 38, 1-4.

- Paas, F., Van Gog, T., & Sweller, J. (2010). Cognitive load theory: New conceptualizations, specifications, and integrated research perspectives. *Educational Psychology Review*, 22, 115-121.
- Paivio, A. (2006). *Dual coding theory in education*. Retrieved from <http://www.umich.edu/~rdytolrn/pathwaysconference/presentations/paivio.pdf>
- Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., Paterson, J. (2007). A survey of literature on the teaching of introductory programming, *ACM SIGCSE Bulletin*, 39(4), 204-223.
- Pitts, C., Ginns, P., & Errey, C. (2006). *Cognitive load theory and user interface design: Making software easy to learn and use – Part1*. Retrieved from <http://www.ptg-global.com/papers/psychology/cognitive-load-theory.cfm>
- Prasad, C., & Fielden, K. (2003). Introducing Programming: A balanced approach. *The New Zealand Journal of Applied Computing and Information Technology*, 7(1), 89-94
- Prasad, C. & Li, X. (2004). Teaching introductory programming to information systems and computing majors: Is there a difference? *Proceedings of the Sixth Australasian Computing Education Conference*, Dunedin, New Zealand, 261-267.
- Product announcement. (2011). Retrieved from <http://msdn.microsoft.com/en-us/vjsharp/default.aspx>
- Puntambekar, S., & Hubscher, R. (2005). Tools for scaffolding students in a complex learning environment: What have we gained and what have we missed? *Educational Psychologist*, 40(1), 1-12.

- Reeves, T. C. (1993). Evaluating interactive multimedia. In D. M. Gayeski (Ed.), *Multimedia for learning: Development, application, evaluation* (pp. 97-112). Englewood Cliffs, NJ: Educational Technology Publications.
- Reigeluth, C. M. (1996). A new paradigm of ISD? *Educational Technology*, 36(3), 13-20.
- Reis, C., & Cartwright, R. (2004). Taming a professional IDE for the classroom. *ACM. SIGCSE Bulletin*, 36(1), 156–160.
- Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, 13(2), 137-172.
- Roper, C. (2012). *Programmers*. Retrieved from <http://programmers.stackexchange.com/questions/89158/do-you-have-to-be-good-at-math-to-be-a-good-programmer>
- Ryu, J., Lai, T., Colaric, S., Cawley, J., & Aldag, H. (2000). *Dual coding theory*. Retrieved from <http://iteach.saintleo.edu/InstructionalDesign/Paivio.html>
- Sebastian, C. (2004). *Brain dominance test*. Retrieved from <http://www.ipn.at/ipn.asp?BHX>
- Seel, N. M. (2001). Epistemology, situated cognition, and mental models: Like a bridge over troubled water. *Instructional Science*, 29(4-5), 403-427.
- Smith, E. E. (2000). Neural bases of human working memory. *Current Directions in Psychological Science*, 9, 45-49.
- Smith, E. E., & Jonides, J. (1999). Storage and executive processes in the frontal lobes. *Science*, 283, 1657-1661.

- Stahl, G. (2006). Supporting group cognition in an online math community: A cognitive tool for small-group referencing in text chat. *Journal of Educational Computing Research*, 35, 103-122.
- Sung, H. (2010). *Response to intervention: A catalyst for paradigm shift for 21st century education*. Retrieved from <http://jepsc.org/JEPSCFall2010.pdf>
- Sweller, J. (1988). Cognitive load during problem solving: Effects on learning, *Cognitive Science*, 12(2), 257-285.
- Sweller, J. (1999). *Instructional design in technical areas*. Camberwell, Australia: ACER Press.
- Sweller, J. (2004). Instructional design consequences of an analogy between evolution by natural selection and human cognitive architecture. *Instructional Science*, 32, 9-31.
- Sweller, J. (2010). Element interactivity and intrinsic, extraneous, and germane cognitive load. *Educational Psychology Review*, 22(2), 123-138.
- Sweller, J., Van Merriënboer, J., & Paas, F. (1998). Cognitive architecture and instructional design. *Educational Psychology Review*, 10, 251–296.
- Tabak, I. (2004). Synergy: A compliment to emerging patterns of distributed scaffolding. *Journal of the Learning Sciences*, 13(3), 305-335.
- Tan, N. Y. L., Chen, W., & Looi, C. K. (2009). GroupScribbles as a rapid CSCL tool: Learning experiences of pre-service teachers. *Proceedings of the 17th International Conference on Computers in Education [CDROM]*. Hong Kong.

- Thomas, N. J. T. (2010). *Dual coding and common coding theories of memory*. Retrieved from <http://plato.stanford.edu/entries/mental-imagery/theories-memory.html>
- Thomas, L., Ratcliffe, M., Woodbury, J., & Jarman, E. (2002). Learning styles and performance in the introductory programming sequence. In *Proceedings of the 33rd SIGCSE Technical Symposium* (pp. 33-37). New York, NY: ACM.
- Ursyn, A., & Scott, T. (2007). Web with art and computer science. *Proceedings of ACM SIGGRAPH 2007 educators program*, San Diego, CA.
- Utting, I. (2006). Problems in the initial teaching of programming using Java: The case for replacing J2SE with J2ME. *Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (pp. 193-196). Bologna, Italy.
- Van Gerven, & Pascal W. M. (2003). The efficiency of multimedia learning into old age. *British Journal of Educational Psychology*, 73(4), 489-505.
- Van Gog, T., Paas, F. & Sweller, J. (2010). Cognitive load theory: Advanced in research on worked examples, animations, and cognitive load measurement. *Educational Psychological Review*, 22, 375-378
- Van Haaster, K., & Hagan, D. (2004). Teaching and learning with BlueJ: An evaluation of a pedagogical tool. *Information Science and Information Technology Education Joint Conference*. Rockhampton, Australia.
- Van Merriënboer, J. J. G., & Paas, F. (1990). Automation and schema acquisition in learning elementary computer programming. *Computers in Human Behavior*, 6(3), 273-289.

- Van de Ven, G. & Govers, E. (2007). The difficulty with programming: Improving teaching and learning in introductory programming. *The New Zealand Journal of Applied Computing and Information Technology*, 11(1), 80-91.
- Vogts, D., Calitz, A. P., & Greyling, J. H. (2010). The effects of professional and pedagogical program development environments on novice programmer perceptions. *South African Computer Journal*, 45, pp. 53-58.
- Welcome to the RAPTOR home page (2011). Retrieved from <http://raptor.martincarlisle.com/>
- Wen, D., Graf, S., Lan, C. H., Anderson, T., & Dickson, K. K. (2007). Supporting Web-based Learning through Adaptive Assessment. *FormaMente Journal*, 1-2(2), 45-79.
- Werhane, P. H., Hartman, L.P., Moberg, D., Englehardt, E., Pritchard, M., & Parmar, B. (2011). Social constructivism, mental models, and problems of obedience. *Journal of Business Ethics*, 100(1), 103-118.
- Wertsch, J. V. (1985). *Vygotsky and the social formation of the mind*. Cambridge, MA: Harvard University Press.
- Willis, C.L., & Miertschin, S. L. (2005). Mind tools for enhancing thinking and learning skills. *Proceedings of 6th Conference on Information Technology Education*, New Jersey, NJ:ACM.
- Willis, C.L., & Miertschin, S. L. (2006). Mind maps as active learning tools. *Journal of Computing Sciences in Colleges* 21(4), 266–272.
- Willis, J. (1995). A recursive, reflective instructional design model based on constructivist-interpretivist theory. *Educational Technology*, 35(6), 5-23.

- Willis, J., & Wright, K. E. (2000). A general set of procedures for constructivist instructional design: The new R2D2 model. *Educational Technology*, 40(2), 5-20.
- Winslow, L. E. (1996). Programming pedagogy : A psychological overview. *SIGCSE Bulletin*, 28, 17-22.
- Wittwer, J., & Renkl, A. (2010). How effective are instructional explanations in example-based learning? A meta-analytic review. *Educational Psychology Review*, 22(4), 393-409.
- Wood, D., Bruner, J. S., & Ross, G. (1976). The role of tutoring in problem solving. *Journal of Child Psychology & Psychiatry & Allied Disciplines*, 17(2), 89-100.
- Wu, C., Dale, N. B., & Bethel, L. J. (1998). Conceptual models and cognitive learning styles in teaching recursion. *Proceedings of the Twenty-Ninth SIGCSE Technical Symposium*, Atlanta, Georgia, GA, 292-296.
- Ying, Z., Gang, H., Nuyun Z., & Hong, M. (2009). Smart tutor: Creating IDE-based interactive tutorials via editable replay. *Proceedings of the of the 31st International Conference on Software Engineering, Vancouver, Canada*, 559-562
- Youssoof, M., Sapiyan, M., & Kamaluddin, K. (2006). Reducing cognitive load in learning computer programming. *Transactions on Engineering, Computing and Technology*, 12, 259-262.
- Yuan, K., Steedle, J., Shavelson, R., Alonzo A., & Oppezzo, M. (2006). Working memory and fluid intelligence and science learning. *Educational Research Review*, 1(2), 83-98.



Zander, C., Thomas, L., Simon, B., Murphy, L., McCauley, R., Hanks, B., &  
Fitzgerald, S. (2009). Learning styles: Novices decide. *Proceedings of  
ITiCSE*, 223-227.

*Every reasonable effort has been made to acknowledge the owners of copyright material. I would be pleased to hear from any copyright owner who has been omitted or incorrectly acknowledged.*

## APPENDICES

### APPENDIX A : CODE OF STUDENT CLASS

```
/* File: Student.java
 * Purpose: This Student class contains data relevant to students (id, name,date of
 *         birth).
 * The query methods provide access to the private instance variables.
 * Author:
 * Modified
 * Date
 */
public class Student
{
    private int studentID;          // to store student id number
    private String studentName;     // student's name
    private String studentDOB;      // to store DOB

    /**
     * Default constructor
     */
    public Student()
    {
        studentID = 0;
        studentName = "";
        studentDOB = "";
    }

    /**
     * Alternative constructor 1
     * @param student_id
     */
    public Student(int student_id)
    {
        studentID = student_id;
    }

    /**
     * Alternative constructor 2
     * @param id
     * @param name
     * @param dob
     */
    public Student(int id, String name, String dob)
    {
        studentID = id;
        studentName = name;
        studentDOB = dob;
    }
}
```

```

/**
 * Query method to access student's id number
 * @return studentID
 */
public int getStudentId()
{
    return studentID;
}
/**
 * Command method to assign a value for a student's id number
 * @param Id
 */
public void setId(int Id)
{
    studentID = Id;
}
/**
 * Query method to access student's name.
 * @return StudentName
 */
public String getName()
{
    return studentName;
}
/**
 * Command method to assign a value for a student's name
 * @param aName
 */
public void setName(String aName)
{
    studentName = aName;
}
/**
 * Query method to access student's date of Birth.
 * @return course
 */
public String getDOB()
{
    return studentDOB;
}
/**
 * Command method to assign a value for a student's date of Birth.
 * @param aDOB
 */
public void setCourse(String aDOB)
{
    studentDOB = aDOB;
}
}

```

## APPENDIX B : CODE OF THE EMPLOYEE CLASS

```
/*
 * Class name: Employee
 * Purpose: To create object of the type Employee and manipulate
 * Author:
 * Date created:
 */

public class Employee
{
    // Instance variables
    private String name;
    private int age;
    private double salary;

    // Default Constructor
    public Employee()
    {
        name="";
        age =0;
        salary = 0.0;
    }
    // Alternative Constructor
    public Employee(String aName, int aAge, double aSal )
    {
        name=aName;
        age =aAge;
        salary = aSal;
    }

    // This method is used to add a name to an object
    public void setName(String aaName)
    {
        name = aaName;
    }

    // This method is used to get the value of the name of an object
    public String getName()
    {
        return name;
    }

    // This method is used to add age to an object
    public void setAge(int aaAge)
    {
        age = aaAge;
    }
}
```

```
// This method is used to get the value of the age of an object
public int getAge()
{
    return age;
}
```

```
// This method is used to set salary value to an object
public void setSalary(double aaSal)
{
    salary = aaSal;
}
```

```
// This method is used to get the value of the age of an object
public double getSalary()
{
    return salary;
}
```

```
// This method is used to print the name of an object
public void printName()
{
    System.out.println(name);
}
```

```
// This method is used to print the Age of an object
public void printAge()
{
    System.out.println(age);
}
```

```
public void printSalary()
{
    System.out.println(salary);
}
```

```
}
```

```
// This is to test the Employee class
public class EmployeeTester
{
    public static void main (String[] args)
    {
```

```
//Create an object called E1 using Employee class
    Employee E1 = new Employee();
```

```
//Create an object called E2 using Employee class
    Employee E2 = new Employee();
```

```
//Create an object called E3 using Employee class
```

```
Employee E3 = new Employee();

//Add name Don to name property of E3 object
E3.setName("Don");

//Add name Don to name property of E1 object
E1.setName("Peter");

// Print name of E3 using printName method
E3.printName();

}
}
```

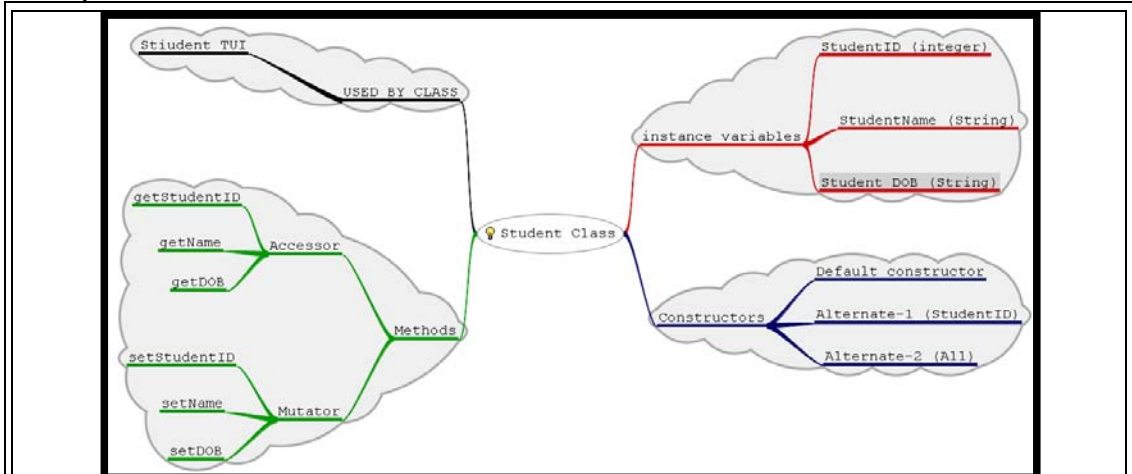
## APPENDIX C: MINI QUESTIONNAIRE - 1

### NOTE

Please read the information sheet and sign the consent form before answering the questions in this survey.

The tutor used mind maps to explain the concept of Java class template to students in the last two weeks.

Examples:



Q1. Do you think that the mind maps help to understand Java class templates better?

YES

NO

Q2. If your answer is yes to the question above question, Why do you think mind maps help to understand Java class templates better?

Please write your comments?

Please write your comments on what else the tutor could do to help you to understand Java class template better?

In our last class, we used worked examples along with mind maps and teacher guided practical sessions for the students to learn by experiencing how instance variables, constructors, and methods work. This method was used instead of using powerpoint slides.

Q3. Do you think it was a better way of teaching?

YES	
NO	

Q4. If the answer is YES to the above question, why do you think the worked examples and teacher guided hands-on sessions help to learn Java class template better? Please write your comments.

Q4. Which one of the following aspects of teaching was the most useful in this teaching session?

Visual features	
Worked Examples	
Teacher Guidance	
Hands-on Practical	



Q5. What combination of the following tool/method(s) do you suggest for teaching this concept? You may tick more than one option.

Mind map	
Worked Example	
Teacher Guidance	
Hands-on Practical	

Q6. How do you rate your artistic ability like singing, painting, and writing poetry etc.? Please tick one option

Poor	
Average	
Good	
Very good	
Excellent	

Q7. How do you rate your logical, analytical thinking, and mathematics ability? Please tick one option

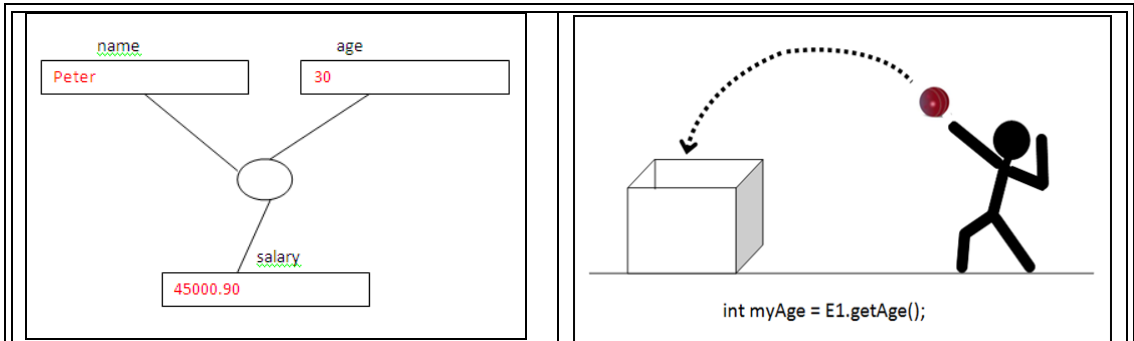
Poor	
Average	
Good	
Very good	
Excellent	

## APPENDIX D: MINI QUESTIONNAIRE-2

Please read the information sheet and sign the consent form before answering the questions in this survey

The tutor used graphical pictures to teach creation of objects using a class template, assigning values, and retrieving contents.

### Star Structure



Q1. Do you think that the star structure and analogy used helps students to understand creation and manipulation of the contents of objects better?

YES

NO

Q2. If your answer is yes to the above question, why do you think they help to learn the concept better?

Please write your comments.

Please write your comments on what else the tutor could do to help you understand this Java concept better.

In our last class, we used worked examples along with the star structure as a pictorial representation of an object to understand the creation and manipulation of objects. It was a teacher guided session and the students learnt by modifying the Java code given to them. This method was used instead of using PowerPoint slides.

Q3. Do you think it was a better way of teaching?

YES

NO

If the answer is YES to the above question, why do you think the teacher guided hands-on sessions using worked examples help learning Java concepts better? Please write your comments.

Q4. Which one of the following aspects of teaching was the most useful in this teaching session?

Worked Examples

Teacher Guidance

Hands-on Practical

Visual / Analogy used

Q5. What combination of the following tool/method(s) do you suggest for teaching this concept? You may tick more than one option.

Visual / Analogy used	
Worked Example	
Teacher Guidance	
Hands-on Practical	

Q6. How do you rate your artistic ability like singing, painting, and writing poetry etc.? Please tick one option

Poor	
Average	
Good	
Very good	
Excellent	

Q7. How do you rate your logical, analytical thinking, and mathematics ability? Please tick one option

Poor	
Average	
Good	
Very good	
Excellent	

## APPENDIX E: MINI QUESTIONNAIRE – 3

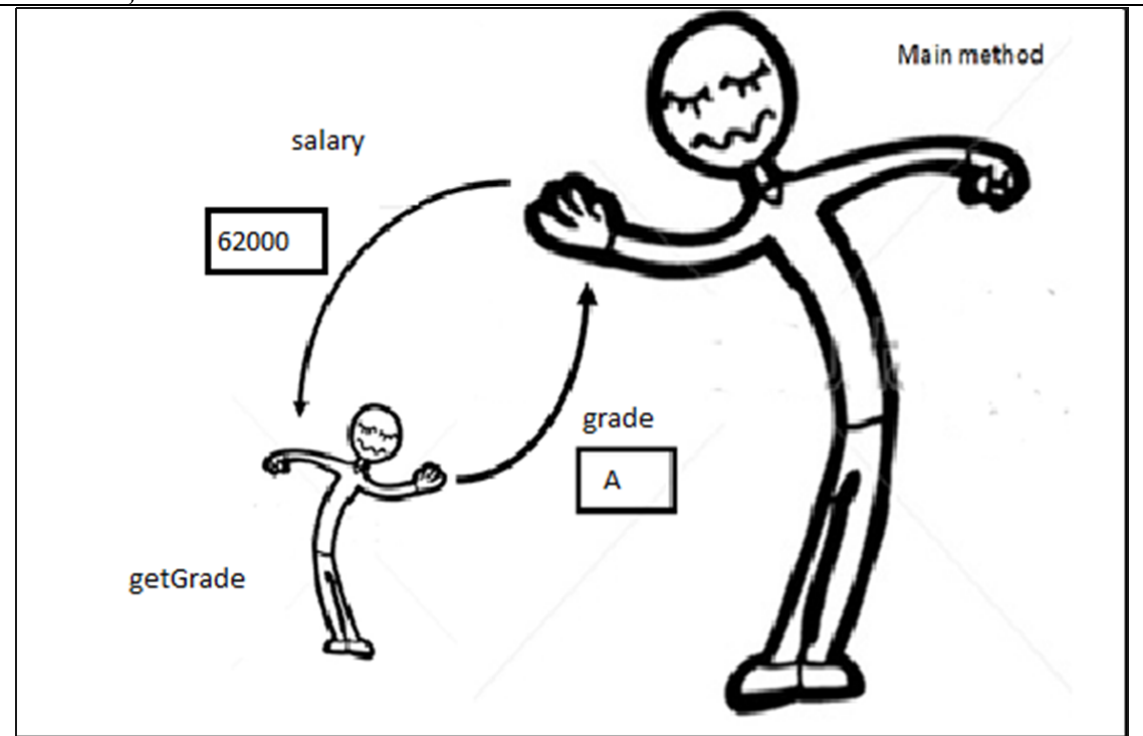
Please read the information sheet and sign the consent form before answering to the questions in this survey

The tutor used a picture to visualise the use of parameter variables of a method.

Example:

### Java Code

```
public char getGrade(double salary)
{
    char ch=' ';
    if (salary >= 60000)
        ch= 'A';
    else if ((salary < 60000) && salary > 40000)
        ch= 'B';
    else
        ch= 'C';
    return ch;
}
```



Q1. Do you think that the picture analogy, hands-on exercise, and teacher support helped students to understand the use of parameter variables in a method?	
YES	
NO	

Q2. If your answer is yes to the above question, why do you think this activity helped to learn this concept?

Please write your comments.

--

Please write your comments on what else the tutor could do to help you understand this Java concept better?

Q4. Which one of the following aspects of teaching was the most useful in this teaching session?

Visual / Analogy used	
Teacher Guidance	
Hands-on Practical	

Q5. What combination of the following teaching tool/method(s) do you suggest for teaching this concept? You may tick more than one option.

Visual / Analogy used	
Teacher Guidance	
Hands-on Practical	

Q6. How do you rate your artistic ability like singing, painting, and writing poetry etc.?  
Please tick one option.

Poor	
Average	
Good	
Very good	
Excellent	

Q7. How do you rate your logical, analytical thinking, and mathematics ability?  
Please tick one option

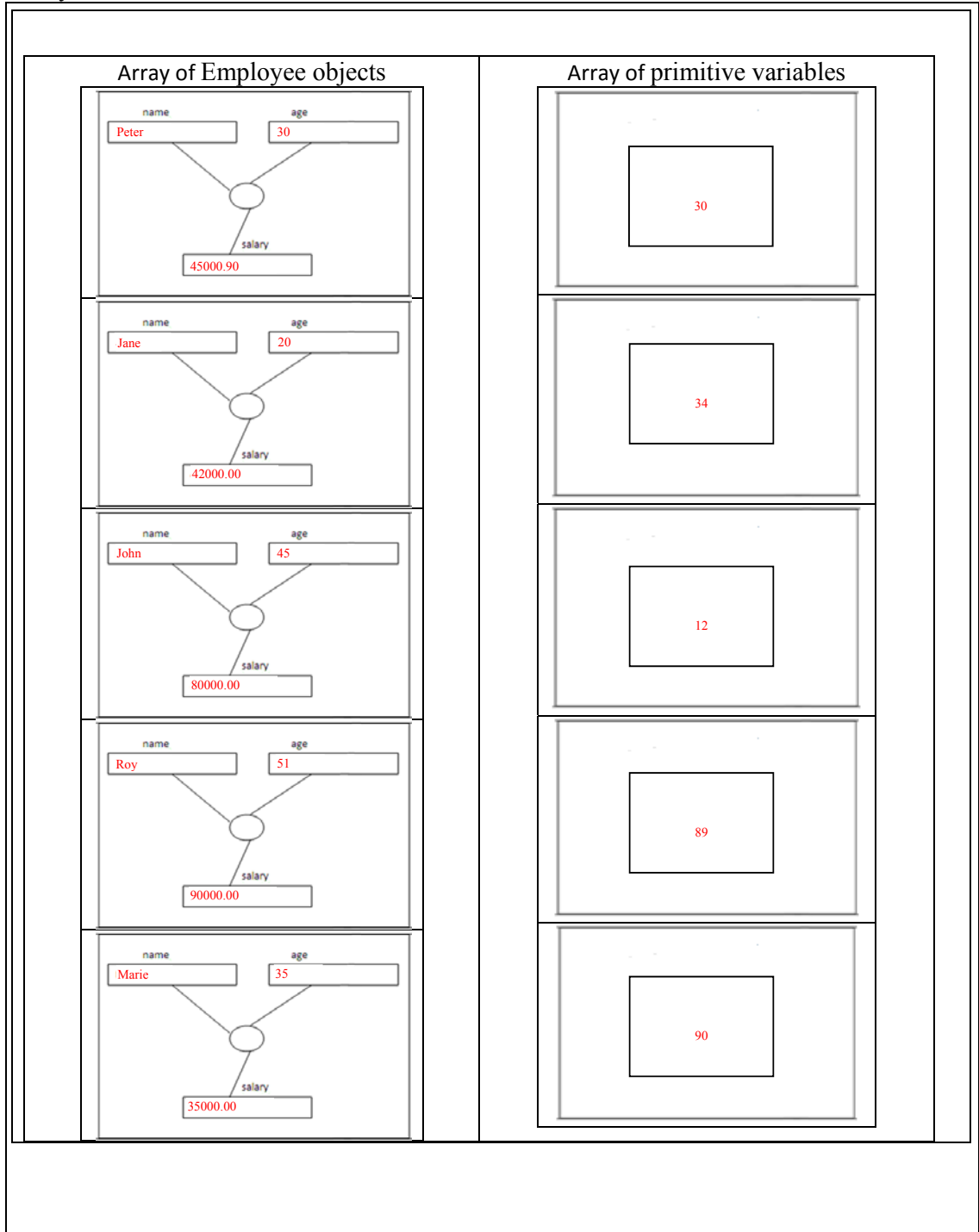
Poor	
Average	
Good	
Very good	
Excellent	

## APPENDIX F: MINI QUESTIONNAIRE - 4

Please read the information sheet and sign the consent form before answering to the questions in this survey

The tutor used array structures to visualise the objects to teach the use of variables and objects in arrays.

### Array Structures





Q1. Do you think that this activity helped students to understand the use of arrays of variables and objects well?	
YES	
NO	

Q2. If your answer is yes to the above question, Why do you think it helped to learn the concept better?

Please write your comments.

Please write your comments on what else the tutor could do to help you understand this Java concept better.

In our last class, we used worked examples along with the array structures as pictorial representations to support learning the use of arrays. It was a teacher guided session and the students learnt by understanding and modifying the Java code given to students. This method was used instead of using powerpoint slides.

Q3. Do you think it was a better way of teaching?

YES	
NO	

Q4. If the answer is YES to the above question, why do you think the teacher guided hands-on sessions using worked examples help learning Java better? Please write your comments.

--

Q4. Which one of the following aspects of teaching was the most useful in this teaching session?

Array Structure	
Worked Examples	
Teacher Guidance	
Hands-on Practical	

Q5. What combination of the following tool/method(s) do you suggest for teaching this concept? You may tick more than one option.

Array Structure	
Worked Example	
Teacher Guidance	
Hands-on Practical	

Q6. How do you rate your artistic ability like singing, painting, and writing poetry etc.? Please tick one option.

Poor	
Average	
Good	
Very good	
Excellent	

Q7. How do you rate your logical, analytical thinking, and mathematics ability?  
Please tick one option

Poor	
Average	
Good	
Very good	
Excellent	

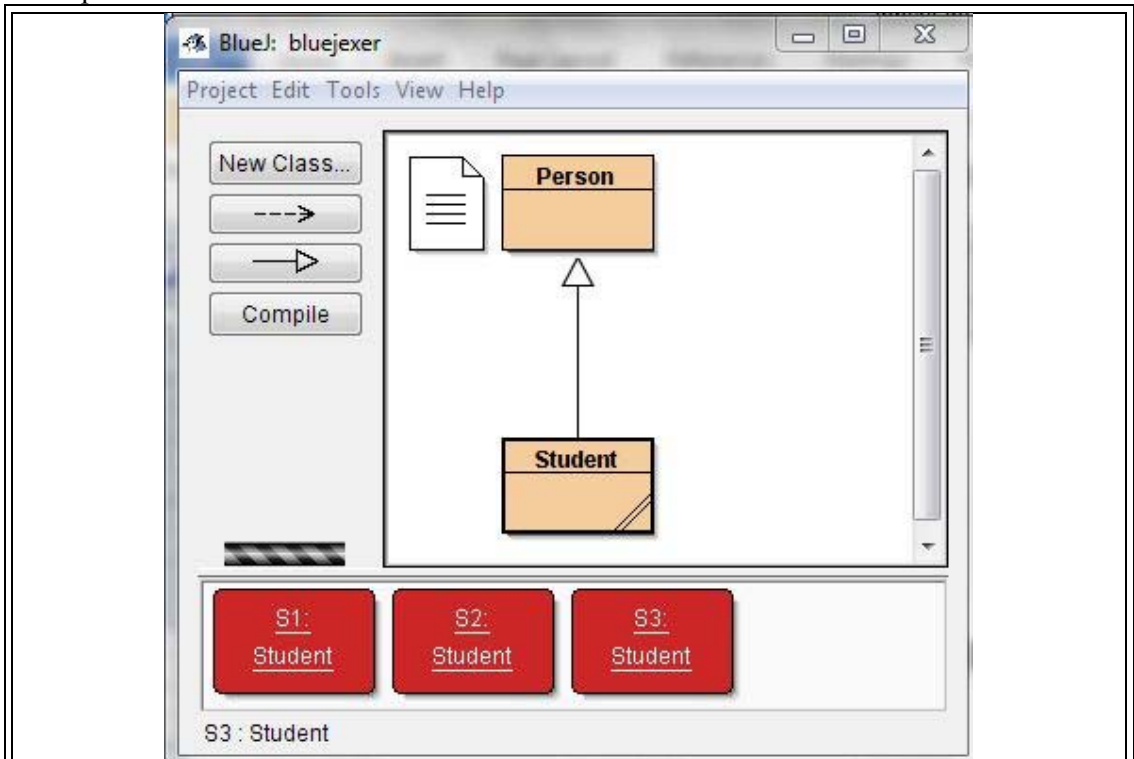
## APPENDIX G: MINI QUESTIONNAIRE - 5

## NOTE

Please read the information sheet and sign the consent form before answering the questions in this survey.

The tutor used BlueJ visual Java programming environment as a tool to explain Inheritance and polymorphism by creating super class (Person) and sub class (Student). Bluej was also used to test methods and constructors.

Example:



Q1. Do you think that the BlueJ helps to understand inheritance programming concepts better?

YES

NO

Q2. If your answer is yes to the above question, why do you think BlueJ helps to understand inheritance programming concepts better?

Please write your comments.

--

We did an exercise using a teacher guided practical hands-on session using BlueJ to learn inheritance concept of the Java Language.

Q3. Do you think it was a better way of teaching?

YES	
NO	

Q4. If the answer if YES to the above question, why do you think teacher guided hands-on sessions using Bluej help to learn inheritance and Java constructors and methods better?  
Please write your comments

Q5. Which one of the following aspects of teaching was the most useful in this teaching session?

BlueJ Visual Interface	
Teacher Guidance	
Hands-on Practical Exercise	

Q6. What combination of the following tool/method(s) do you suggest for teaching this concept? You may tick more than one option.

BlueJ Visual Interface	
Teacher Guidance	
Hands-on Practical Exercise	

Q7. How do you rate your artistic ability like singing, painting, and writing poetry etc.?  
Please tick one option

Poor	
Average	
Good	
Very good	
Excellent	

Q8. How do you rate your logical, analytical thinking, and mathematics ability?

Please tick one option

Poor	
Average	
Good	
Very good	
Excellent	

## APPENDIX H: PARTICIPANT INFORMATION SHEET 1



Curtin University of Technology

School of School of Science & Computing  
The Science and Mathematics Education Centre (SMEC)

### Participant Information Sheet 1

My name is Don Nimal Padmasiri Kannangara. I am currently completing a piece of research for my Doctor of Philosophy (PhD) at Curtin University of Technology.

#### **Purpose of Research**

I am investigating the possibility of improving teaching Java programming at the introductory level using a number of teaching tools in accordance with the principles of Cognitive Load Theory.

In the first phase of the research, I hope find out the areas and the concepts which were hard to understand for most students when they were learning the introductory level programming using the Java language. The questionnaire contains a list of areas and concepts which are assumed to be important to understand for students.

#### **Your Role**

I am interested in finding out the difficulty level of each of the areas and concepts listed in the questionnaire. The questionnaire is available on the internet. I will ask you to choose one of the difficulty level options from the combo box in each of the areas and concepts listed in the questionnaire. If you had any other difficult areas or concepts when you were learning the Java language, I ask you to type it in the space provided and choose the difficulty level for each of them from the combo boxes. The web address will be given to you by person who gives out this information sheet.

**Consent to Participate**

Your involvement in the research is entirely voluntary. You have the right to withdraw at any stage without it affecting your rights or my responsibilities. When you have signed the consent form or ticked the check box on the webpage, I will assume that you have agreed to participate and allow me to use your data in this research.

**Confidentiality**

The information of personal details is not required in this questionnaire. The questionnaire will not have space for you to enter your name or any other identifying information. In adherence to university policy, the collected data will be kept in a locked cabinet for five years before it is destroyed.

**Further Information**

This research has been reviewed and given approval by Curtin University of Technology Human Research Ethics Committee (Approval number SMEC-01-09). If you would like further information about the study, please feel free to contact me on +64 7 3468688 or by email: [don.kannangara@waiairiki.ac.nz](mailto:don.kannangara@waiairiki.ac.nz). Alternatively you can contact my supervisor Prof. Darrell Fisher on +61 8 9266 3110 or email: [d.fisher@curtin.edu.au](mailto:d.fisher@curtin.edu.au)

**Thank you very much for your involvement in this research. Your participation is greatly appreciated.**



## APPENDIX I: CONSENT FORM



### CONSENT FORM

---

- I understand the purpose and procedures of the study.
  - I have been provided with the participant information sheet.
  - I understand that the procedure itself may not benefit me.
  - I understand that my involvement is voluntary and I can withdraw at any time without problem.
  - I have been given opportunity to ask questions.
  - I agree to participate in the study outlined to me.
- 

**Signature** \_\_\_\_\_ **Date** \_\_\_\_\_

**Witness Signature** \_\_\_\_\_ **Date** \_\_\_\_\_

## APPENDIX J: PARTICIPANT INFORMATION SHEET 2



Curtin University of Technology

School of School of Science & Computing  
The Science and Mathematics Education Centre (SMEC)

### Participant Information Sheet 2

My name is Don Nimal Padmasiri Kannangara. I am currently completing a piece of research for my Doctor of Philosophy (PhD) at Curtin University of Technology.

#### **Purpose of Research**

I am investigating the possibility of improving teaching Java programming at the introductory level using a number of teaching tools in accordance with the principles of Cognitive Load Theory. In the second phase of the research, I hope to find out about the usefulness of the teaching tool that I have used in my session. I also would like to know about the adequacy of the subject knowledge given, relevance of information given, and also about the teaching style used in my teaching session. The mini-questionnaire contains about 5 questions.

#### **Your Role**

I am interested in finding out how effective the teaching tool that was used in my Java programming session today. I will ask you to answer the questions given in the mini-questionnaire. Please do not include your personal information such as name or ID number on the questionnaire.

#### **Consent to Participate**

Your involvement in the research is entirely voluntary. You have the right to withdraw at any stage without it affecting your rights or my responsibilities. When you have signed the consent form I will assume that you have agreed to participate and allow me to use your data in this research.

**Confidentiality**

The information of personal details, is not required in this questionnaire. In adherence to university policy, the collected data will be kept in a locked cabinet for five years before it is destroyed.

**Further Information**

This research has been reviewed and given approval by Curtin University of Technology Human Research Ethics Committee (Approval number SMEC-01-09). If you would like further information about the study, please feel free to contact me on +64 7 3468688 or by email: [don.kannangara@waiariki.ac.nz](mailto:don.kannangara@waiariki.ac.nz). Alternatively you can contact my supervisor Prof. Darrell Fisher on +61 8 9266 3110 or email: [d.fisher@curtin.edu.au](mailto:d.fisher@curtin.edu.au)

**Thank you very much for your involvement in this research. Your participation is greatly appreciated.**

## APPENDIX K: ETHICAL APPROVAL

memorandum



<b>To</b>	Don Nimal Padmasiri Kannangara, SMEC
<b>From</b>	Pauline Howat, Coordinator for Human Research Ethics, Science and Maths Education Centre
<b>Subject</b>	Protocol Approval <b>SMEC-01-09</b>
<b>Date</b>	9 February 2009
<b>Copy</b>	Darrell Fisher, SMEC Divisional Graduate Studies Officer, Division of Science and Engineering

Office of Research and Development

### Human Research Ethics Committee

TELEPHONE 9266 2784  
FACSIMILE 9266 3793  
EMAIL hrec@curtin.edu.au

Thank you for your "Form C Application for Approval of Research with Minimal Risk (Ethical Requirements)" for the project titled "*TEACHING TOOLS FOR EFFECTIVE TEACHING AND LEARNING OF COMPUTER PROGRAMMING USING JAVA FOR BEGINNERS*". On behalf of the Human Research Ethics Committee I am authorised to inform you that the project is approved.

Approval of this project is for a period of twelve months **27th January 2009 to 26th January 2010**.

If at any time during the twelve months changes/amendments occur, or if a serious or unexpected adverse event occurs, please advise me immediately. The approval number for your project is **SMEC-01-09**. Please quote this number in any future correspondence.

A handwritten signature in cursive script, appearing to read "Pauline".

PAULINE HOWAT  
Coordinator for Human Research Ethics  
Science and Maths Education Centre

---

Please Note: The following standard statement must be included in the information sheet to participants: *This study has been approved by the Curtin University Human Research Ethics Committee (Approval Number SMEC-01-09). If needed, verification of approval can be obtained either by writing to the Curtin University Human Research Ethics Committee, c/- Office of Research and Development, Curtin University of Technology, GPO Box U1987, Perth, 6845 or by telephoning 9266 2784.*

## APPENDIX L: QUESTIONNAIRE – PHASE ONE

\*1) I agree to participate in this survey.

☐ Yes ☐ No

\*2) Your gender?

☐ Male ☐ Female

\*3) Your age group?

☐ below 20 years ☐ 20 to 29 years ☐ 30 to 39 years ☐ 40 to 49 years ☐ 50 and above

\*4) Your work experience?

☐ None ☐ Less than 5 years ☐ More than 5 years

\*5) Your highest academic qualification prior to this course?

☐ Form 5 or below (NZ) / 10 Std (India) ☐ Form 6 or 7 (NZ) / 10+2 Std (India) ☐ Tertiary qualification

\*6) How do you rate your artistic ability like singing, painting, and writing poetry etc.?

☐ Poor ☐ Average ☐ Good ☐ Very good ☐ Excellent

\*7) How do you rate your logical, analytical thinking, and mathematics ability?

<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Poor	Average	Good	Very good	Excellent

8) Please select one of the difficulty levels for each of the concepts or areas listed below with the experience you had while learning the Java language at the introductory level.

	1	2	3	4	5
	too difficult	very difficult	difficult	not difficult	very easy
Variable types (int, char, & double etc.)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Variable categories (local, parameter & instance)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Using conditional statements (if ..then ..else)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Using repetitive statements (loops)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Understanding of the concept of classes and objects	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Creation of an object using a Class	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Creation of a template for a class	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Use of parameter variables (arguments) in a method	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Returning a value from a method	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Testing and debugging	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Preparation of test data for a program	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Using arrays for primitive data types	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Using arrays for objects of a class	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
String manipulation in Java (using methods in String class)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Using text files for input and output	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Inheritance	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Logic depiction methods (structure diagrams etc.)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

9) Please enter the difficult concepts or areas which are not listed in the above question and the difficult level for each of them.

1 too difficult	2 very difficult	3 difficul t	4 not difficul t	5 very easy	Plese enter the difficult concept/are as here
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>

10) Which one of the following logic depiction methods do you prefer for program design?

24519864			
<input type="checkbox"/> Structure Diagram	<input type="checkbox"/> Structured English	<input type="checkbox"/> Flowchart	Other (Please Specify): <div><input type="text"/></div>

There are three different types of learning styles

1. The **auditory learner** learns by hearing and expects someone to explain.



2. The **kinaesthetic learner** requires hand on experience.



3. A **visual learner** learns by drawing diagrams, using mental pictures, and re writing lecturer's notes etc.
















































11) In your judgement, What type of learner are you? You may tick more than one learner type. Please write your comments.

<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Other (Please Specify):
Auditory learner	Kinaesthetic learner	Visual learner	<div style="border: 1px solid black; height: 60px; width: 250px;"></div>

12) Which one of the following teaching styles would be the most suitable to teach each of the following areas? Please indicate your second choice (if any) on the text box.

	Auditory	Kinaesthetic	Visual	Your Second Choice
Variable types (int, char, & double etc.)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<div style="border: 1px solid black; width: 80px; height: 20px;"></div>
Variable categories (local, parameter & instance)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<div style="border: 1px solid black; width: 80px; height: 20px;"></div>



Using conditional statements (if ..then ..else)				<input type="text"/>
Using repetitive statements (loops)				<input type="text"/>
Understanding of the concept of classes and objects				<input type="text"/>
Creation of an object using a Class				<input type="text"/>
Creation of a template for a class				<input type="text"/>
Use of parameter variables (arguments)in a method				<input type="text"/>
Returning a value from a method				<input type="text"/>
Testing and debugging				<input type="text"/>
Preparation of test data for a program				<input type="text"/>
Using arrays for primitive data types				<input type="text"/>
Using arrays for objects of a class				<input type="text"/>
String manipulation in Java (using methods in String class)				<input type="text"/>
Using text files for input and output				<input type="text"/>
Inheritance				<input type="text"/>
Logic depiction methods (structure diagrams etc.)				<input type="text"/>