



OpenAIR@RGU

The Open Access Institutional Repository at Robert Gordon University

<http://openair.rgu.ac.uk>

Citation Details

Citation for the version of the work held in 'OpenAIR@RGU':

<p>REY, J. S., 2009. From Alice to BlueJ: a transition to Java. Available from <i>OpenAIR@RGU</i>. [online]. Available from: http://openair.rgu.ac.uk</p>
--

Copyright

Items in 'OpenAIR@RGU', Robert Gordon University Open Access Institutional Repository, are protected by copyright and intellectual property law. If you believe that any material held in 'OpenAIR@RGU' infringes copyright, please contact openair-help@rgu.ac.uk with details. The item will be removed from the repository while the claim is investigated.

FROM ALICE TO BLUEJ : A TRANSITION TO JAVA

JEANETTE S. REY

A thesis submitted in partial fulfilment of the
requirements of
The Robert Gordon University
for the degree of Master of Science (by Research)

June 2009

ABSTRACT

Jeanette S. Rey

Master of Science (by Research)

Title: "From Alice to BlueJ: A Transition to Java"

Keywords: Object-Oriented Programming, Alice, BlueJ, Java, Pedagogy, Assessment

The teaching of introductory courses in computing has seen several changes over the last decade. These changes not only affected the curricula when the emphasis was shifted from Imperative (also Procedural) to Object-Oriented Programming (OOP) but also reignited debates regarding which is the better programming language. Furthermore, the shift in emphasis also has encountered challenges with the object-oriented pedagogy. More recently, the assessment procedure for how students are learning object-oriented concepts has been given attention.

When the programming language Java was adopted to teach object-oriented programming, it was not without difficulties. Various studies cited the development environment for Java, which was designed for professional programmers and its complex syntax structures as the main source of Java's difficulties (Kolling & Rosenberg 2001).

The studies were not only limited to identifying the problems of teaching and learning Java but they also identified solutions. One of these included the creation of programming tools and environment to help novice programmers learn object-oriented concepts effectively. Among the integrated development environments created for teaching object-oriented programming using Java is BlueJ. Another programming tool for teaching object-oriented programming is Alice. The technology of animated program visualization keeps the focus on objects while teaching about behaviour and state (Dann et al 2003).

The study concerns how the different programming tools help students in learning object-oriented concepts. One is classified as a text-based tool, BlueJ, and the other is graphical-based tool, Alice. There are three main questions for this study:

1. Does the process of learning object-oriented concepts using graphical-based tools differ from using text-based tools?
2. Do graphical-based tools support text-based tools in learning object-oriented concepts?
3. Do graphical-based tools offer more help in understanding object-oriented concepts than text-based tools?

To answer the questions, the researcher conducted a survey whereby two sets of questionnaires were distributed to the students of Robert Gordon module entitled Object-Oriented Programming Techniques (CM1011). The student respondents found significant difference in the use of the graphical-based and text-based programming tools in understanding the following object-oriented concepts: Message Passing, Encapsulation and Polymorphism. The data gathered were also indicative that a graphical-based programming tool like Alice is helpful in learning object-oriented concepts with the use of a text-based programming tool like BlueJ. Whether graphical tools like Alice help more in understanding object-oriented concepts than text-based tools like BlueJ was inconclusive. The initial study suggests that there was no significant difference with students' confidence in learning the various object-oriented concepts using the programming tools. The student respondents appeared to recognise that both programming tools are useful in learning various object-oriented concepts. However, it seems that they expected Alice to be a more sophisticated animation tool and that the animations produced would be of a cinematic calibre.

The study aspires to contribute to the improvement of the object-oriented pedagogy. Specifically, it aims to contribute in the development of teaching methodologies for object-oriented programming and then create learning strategies for object-oriented programming and, not to forget, make the assessment of object-oriented programming more effective and suitable. Alongside the improvement of object-oriented programming pedagogy, the study also tries to make the computing course curricula more appropriate and flexible with the use of the various programming tools.

Suggestions on how the study can be made more rigorous have been listed including use of additional data gathering instruments and methodology. Also, recommendations on how else the questions can be written were incorporated.

A C K N O W L E D G E M E N T

My sincerest gratitude to everybody who helped me complete the thesis. First of all, to my team of supervisors who have been so patient with me. Dr. Stuart Watt who had helped me concretised what I wanted to do and write about. To Dr. Garry Brindley and Mr. Gordon Eccleston for guiding me with the preparation of my questionnaires. And most of all, to Dr. Roger McDermott who did not only serve as my supervisor but also my editor & proof-reader rolled into one. He even, sometimes go inside my mind and interpret what's inside it. To Mr. Alex Wilson, whose invaluable suggestions regarding statistics have given the researcher perspectives on how to interpret the data quantitatively and qualitatively. And to my panellists, Dr. Janet Hughes and Professor Patrick Holt for making the contents of the thesis more robust and complete.

To my brother-in-law, Jorgen who inspired me to pursue the study here in Aberdeen, Scotland and helped me with all the finances. To my sister, Ate Culeth, who has opened their doors to me and has cared for me ever since I arrived here. To my mother & father (deceased) who have brought us to love learning continuously. To my brothers and sister, Kuya Edwin, Jovy and Jenny who have poured their continuous support. To my nieces and nephews who have inspired me non-stop. And to Millicent Monet, my daughter, who gave me the push to complete the revisions.

To my relatives and friends who have been so kind with their words especially when I feel like going home and not be bothered about the thesis. And to people, who have inspired me especially during those days that I don't feel motivated and ready just to forget about the thesis; Ronald, Bayo and Edmund. And, also to my colleagues in the Computing Research Department for listening and talking to me whether it is about our work or miscellaneous topics.

The completion of this thesis would not be possible without the people above and the Lord who has never let go of me. Again, my gratitude to all of you and to others who have helped in one way or the other.

Thank you so much!

TABLE OF CONTENTS

TITLE PAGE	1
ABSTRACT	II
ACKNOWLEDGEMENT	V
TABLE OF CONTENTS	VI
LIST OF TABLES	VIII
LIST OF FIGURES	IX
CHAPTER 1	1
INTRODUCTION	1
THE FIELD OF STUDY.....	1
THE COURSE CURRICULUM	2
THE RESEARCH QUESTIONS.....	3
OVERVIEW OF THE RESULTS	4
CHAPTER 2	5
BACKGROUND OF THE STUDY	5
BACKGROUND	5
PROGRAMMING PARADIGMS.....	6
PROGRAMMING LANGUAGE.....	7
PROGRAMMING TOOLS	8
CHAPTER 3	18
REVIEW OF RELATED LITERATURES	18
OBJECT-ORIENTED PARADIGM	18
CHANGE IN PROGRAMMING EMPHASIS	20
PROGRAMMING TOOLS	22
SUMMARY	24
CHAPTER 4	25
RESEARCH QUESTIONS	25
THE QUESTIONS	25
WHY ANSWER THE QUESTIONS?	26
CHAPTER 5	28
METHODOLOGIES	28
RESEARCH DESIGN	28
RESPONDENTS OF THE STUDY.....	29
QUESTIONNAIRE	29
CONSTRUCTION OF THE QUESTIONNAIRES.....	31
ALICE AND BLUEJ IN OBJECT-ORIENTED PROGRAMMING	31
DISTRIBUTION AND RETRIEVAL OF THE QUESTIONNAIRES	32
TREATMENT OF DATA	33
• <i>Frequency Distribution and Percentage</i>	33

• <i>Mean</i>	33
• <i>Standard Deviation</i>	34
MANN-WHITNEY TEST	34
STATISTICAL PACKAGE FOR SOCIAL SCIENCES (SPSS)	35
CHAPTER 6	36
PRESENTATION, RESULTS AND ANALYSIS	36
CHAPTER 7	57
CONCLUSIONS AND SUGGESTIONS FOR FURTHER WORK	57
SUMMARY	57
FINDINGS AND CONTRIBUTIONS	58
SUGGESTIONS FOR FURTHER WORK	62
BIBLIOGRAPHY	66
APPENDIX A	X
APPENDIX B	XV

LIST OF TABLES

TABLE 1 RESPONDENTS' AGE AND GENDER.....	36
TABLE 2 RESPONDENTS' COURSE.....	37
TABLE 3 RESPONDENTS' MATHEMATICS ATTAINMENT.....	38
TABLE 4 RESPONDENTS' PROGRAMMING EXPERIENCE.....	39
TABLE 5 RESPONDENTS' COMPUTER AND INTERNET EXPOSURE (ALICE).....	40
TABLE 6 RESPONDENTS' INTERNET EXPOSURE (BLUEJ).....	41
TABLE 7 RESPONDENTS' PREVIOUS KNOWLEDGE OF THE TOOL/PROGRAMMING LANGUAGE	42
TABLE 8 LENGTH OF RESPONDENTS' USE OF THE PROGRAMMING TOOLS.....	43
TABLE 9 RESPONDENTS' FAMILIARITY WITH THE PROGRAMMING TOOLS' INTERFACES	44
TABLE 10 RESPONDENTS' FAMILIARITY WITH PROGRAMMING TOOLS' TERMS (ALICE)	45
TABLE 11 RESPONDENTS' FAMILIARITY WITH PROGRAMMING LANGUAGE' TERMS (JAVA)	46
TABLE 12 MEAN & STANDARD DEVIATION: ALICE & BLUEJ IN COURSEWORK	47
TABLE 13 MANN AND WHITNEY TEST: ALICE & BLUEJ IN COURSEWORK	47
TABLE 14 UNDERSTANDING OBJECT-ORIENTED CONCEPTS USING ALICE & BLUEJ	49
TABLE 15 MANN AND WHITNEY TEST FOR UNDERSTANDING OBJECT-ORIENTED CONCEPTS	50
TABLE 16 MEAN FOR COPING WITH ERRORS	51
TABLE 17 MANN AND WHITNEY TEST FOR COPING WITH ERRORS.....	52
TABLE 18 MEAN & STANDARD DEVIATION OF RESPONDENTS CONFIDENCE IN LEARNING OBJECT-ORIENTED CONCEPTS	53
TABLE 19 MANN AND WHITNEY TEST FOR CONFIDENCE IN LEARNING OBJECT- ORIENTED CONCEPTS	54

LIST OF FIGURES

FIGURE 1 BLUEJ'S INTERFACE	9
FIGURE 2 BLUEJ'S OBJECTS	10
FIGURE 3 BLUEJ'S INSPECTOR	11
FIGURE 4 ALICE'S WORLD	13
FIGURE 5 ALICE'S OBJECTS	13
FIGURE 6 ALICE'S METHODS	14
FIGURE 7 ALICE'S FUNCTIONS	15
FIGURE 8 ALICE'S INTERFACE.....	16

CHAPTER 1

INTRODUCTION

The chapter gives a short description of the study with a general description of the field concerned, a summary of the research questions and an overview of the results obtained.

The Field of Study

The Object-Oriented Paradigm as a software development programme started to gain attention in the mid-80s and this has carried on until today. With so much attention devoted to it, object-oriented programming (OOP) has been integrated not only in the computer science curriculum but has also seen attempts to integrate the subject into secondary education (Henriksen & Kolling 2004). A lot of material has been written describing this type of programming (Wegner 1990) and how object-oriented programming can be better taught as an introduction to computer science or to software development in general (Pugh et al 1987). A substantial amount of time is spent in finding how the object-oriented pedagogy can be improved. Problems in teaching and learning object-oriented programming were identified and addressed by the various studies (e.g. Kolling & Rosenberg 1996). Teachers have started to look at how the assessment component of the object-oriented pedagogy can adapt to these changes (Box 2004; Cable 2001).

Recently, the addition of other type of courses (e.g. Graphics and Animation, Information and Technology, etc.) to more traditional courses on computer science and computer engineering has made the background of learners more diverse. The diversity now demands a closer look at how object-oriented concepts can be better understood by the learners coming from different backgrounds and disciplines. As learner backgrounds

and disciplines become more diverse, the more the pedagogy of object-oriented programming needs to adapt (Burgess & Hanshaw 2006).

It is not only the teaching and learning of object-oriented programming that needs to cope but also the assessment component of the pedagogy (Box 2004). Lecturers are continuously searching for activities that will allow learners to grasp object-oriented concepts and help them apply these to their programming tasks. The combinations of all these considerations have been the topic of publications, readings, and activities in different arenas such as the Association for Computing Machinery (ACM) Special Interest Group on Computer Science Education (Lewis 2000; Rankin et al 2007). Furthermore, these activities have substantial impact on the curriculum, which, has had to be continually developed and revised (Stiller & LeBlanc 2003). This present study was conducted to contribute to the improvement of OOP pedagogy and in the enhancement of the programming curricula.

The Course Curriculum

The study, on which this work was based, was conducted at Robert Gordon University School of Computing in the module entitled Object-oriented Programming Techniques with module reference CM1011. The prerequisite for the course is the module Introduction to Object-oriented Programming with module reference CM1010. The aim of the module (CM1011) is to provide the student with the problem solving skills needed to design, refine and evaluate object-oriented solutions to programming problems of moderate complexity and to develop the student's proficiency in implementing and testing such programs in an object-oriented programming environment. The indicative student workload includes lectures (20 hours), tutorials (10 hours), laboratories (24 hours), assessment (30 hours) and private study (66 hours). There was one (1) coursework

assessment, which involved program design, and development exercises, which tested the learning outcomes (Robert Gordon University 2006).

The Research Questions

The objective of the study was to investigate the improvements in the receptions of object-oriented concepts using two programming tools, which were aimed at helping novice programmers cope with the difficulties of object-oriented programming. These window-based interfaces were aimed at providing a smooth transition to Java in learning object-oriented concepts.

In general terms, the study asks the question: “How do the different types of programming tools, one graphical the other text-based help students to learn object-oriented concepts?” It attempts to differentiate between how learners use graphical-based and text-based tools in their programming tasks, whether the two kinds of programming tools are complementary and which offers the most advantage in understanding specific object-oriented concepts.

By answering these questions it is hoped that the teaching, learning and assessment of object-oriented concept can be improved through:

- Developing, updating and revising existing course curriculum
- Introduction of the programming environments at the right time in the course curriculum
- Proper combination of the different programming tools in the course curriculum
- Identification of how teaching, learning and assessment of object-oriented concepts can be delivered more effectively and efficiently, and
- Identification of other sources of assessing learners’ understanding of object-oriented concepts.

Overview of the results

Results were analysed from the questionnaires distributed to students using the two programming environments, graphical-based and text-based. It was found that the use of a visual programming tool such as Alice to introduce object-oriented programming complemented the use of a text-based programming tool like BlueJ, which was used to build upon this introduction. It was anticipated that students would find Alice an easier environment to complete coursework. The initial study revealed that there was some difference in the use of Alice and BlueJ in understanding the object-oriented concepts of message passing, encapsulation and polymorphism. Perhaps surprisingly, there appeared no significant difference in the use of Alice and BlueJ in understanding Class, Object, Method and Inheritance. Moreover, this initial study suggests that the individual use of the programming tools by the students did not have significant difference in increasing their confidence in understanding the various object-oriented concepts. Mann-Whitney test on the data accepted the null hypothesis that there was no significant difference in the use of Alice or BlueJ in increasing students' confidence in all object-oriented concepts.

CHAPTER 2

BACKGROUND of the STUDY

This chapter describes different concepts related and relevant to the research. Short and concise descriptions are included.

Background

One of the reasons why this study was conducted was to contribute to the improvement of the object-oriented pedagogy. The need to take a closer look how students learn object-oriented concepts was prompted by the difficulties the researcher has encountered in object-oriented programming.

A review of the published materials about object-oriented pedagogy suggested to the researcher that these difficulties were not isolated (e.g. Thomasson et al 2006). These insights and perspectives gained were not only about the difficulties of the object-oriented pedagogy but also about strategies and methods on how to overcome such difficulties (Wei et al 2005). An initial hypothesis was that the cause of the difficulties in learning object-oriented programming stemmed from the particular programming language (i.e. Java). However, further inquiry recognised that the programming language is only one of the challenges of object-oriented pedagogy. Those making the transition to Java and object-oriented techniques must resolve many issues. It is not only the selection of a programming language but also the development environment. The ideal environment would be inexpensive, would be easy to learn, and would highlight, rather than obscure, the design of an object-oriented system (Sanders & Heeler 2001). The identification of the programming tool best suited for the task starts by knowing your list of options.

Programming Paradigms

Historically there are four major programming paradigms namely: Imperative (Procedural), Functional, Declarative and Object-Oriented paradigms.

Imperative Programming (also Procedural) is a programming paradigm that identifies the steps that the program must execute. The basic concept in an imperative paradigm is the procedure call, also known as the routines or methods (Sebesta 1996:20).

In contrast to this, Object-oriented programming (OOP) stresses the use "objects" and their interactions to design application. Its fundamental concepts include ideas of inheritance, modularity, polymorphism, and encapsulation (Pratt & Zelkowitz 1996:35). The object-oriented paradigm identifies the class, object, method, message passing, inheritance, encapsulation, abstraction and polymorphism as the fundamental concept of OOP (Armstrong 2006). The language Simula in the 1960s paved the way for the object-oriented paradigm to be considered as a new way of thinking about how programs can be prepared. Instead of concentrating on procedures, object-oriented programming helped to shift the focus to software objects that can be reused.

Two other programming paradigms have been developed. Functional programming treats computation as the evaluation of mathematical functions and avoids state and mutable data. It emphasizes the application of functions, in contrast with the imperative programming style that stresses changes in state (Hudak 1989). The final paradigm is Declarative programming (which might better be called logical programming by analogy with mathematical programming and linear programming). This in its broadest sense uses mathematical logic for computer programming. In this view of declarative programming, logic is used as a purely declarative representation language, and a theorem-prover or model-generator is used as the problem-solver (Sebesta 1996:22).

The Imperative (Procedural) and the object-oriented have been the most popular paradigms used for industrial or commercial software development. For the last decade,

the use of imperative programming has been replaced by the object-oriented paradigm. This transition has become evident not only in the computing curricula but as well as with the lecturers' struggles that has grown up with the imperative paradigm (Mitchell 2000). The shift from procedural to object-oriented has ignited debates from computer science educators regarding which to teach first (Lister et al 2006). The challenges of shifting from procedural to object-oriented were not only restricted to educators but also to students (Vilner et al 2007).

Programming Language

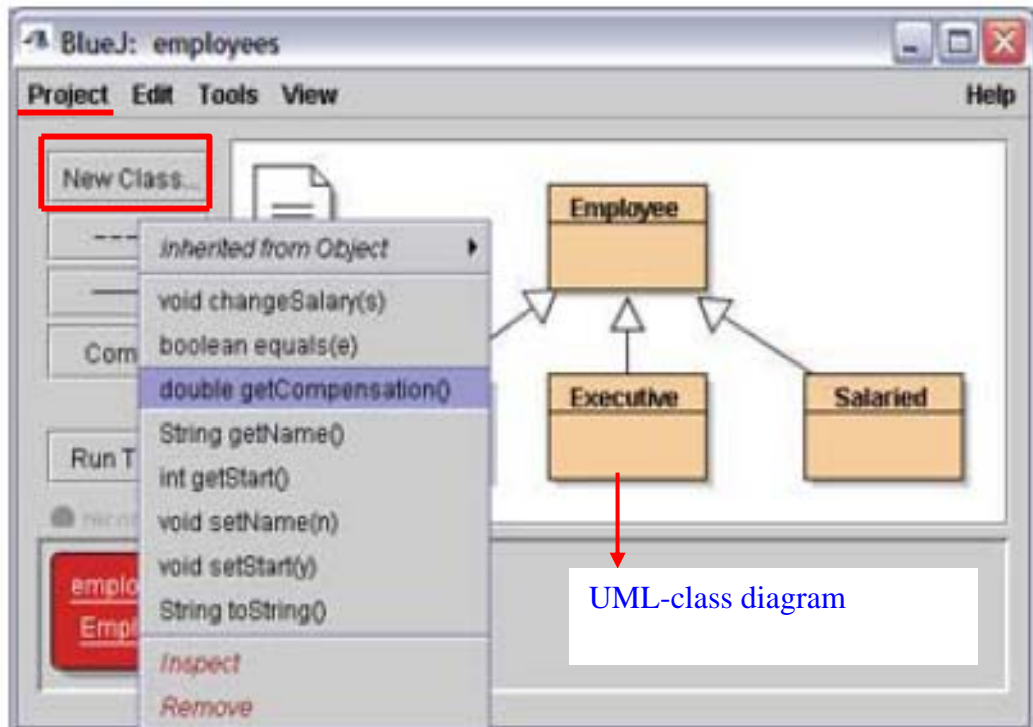
The debate continues on not only about which programming paradigm to use for teaching but to the related question of which programming language to use (Irimia 2001). There are a myriad of programming languages that programmers can use in preparing their programs but over the last decade Java has gained significant foothold among the computing software development community. Numerous studies have been conducted to evaluate Java's suitability as an object-oriented and as an introductory programming language (Hadjerrouit 1998). However, Kolling and Rosenberg (2000) identified that the Java's development environment as a major difficulty in Java courses.

Programming Tools

While Java was as an excellent language for teaching the object-oriented paradigm, the software development environments available were regularly identified as a significant source of problems (Kolling & Rosenberg 2000) and various programming tools were created to help make Java easier to learn (e.g. Dr. J, Eclipse, etc.).

One of the tools suggested to aid in this task is BlueJ. BlueJ is an integrated development environment for Java, which has been specifically designed to support object-oriented design and programming (Kolling & Rosenberg 1996). BlueJ supports a unique introduction of OO concepts by providing an interactive interface (Kolling & Rosenberg 2001). Unified Modeling Language (UML)-like class diagrams are used to present on screen a graphical overview of a project structure. And then it allows objects from any given class in a software project to be created interactively. These objects are visible to the user and any of its public methods can be interactively invoked by selecting it from a pop-up menu. Dialogue windows are used so that parameters and method results can be entered and presented (Kolling & Rosenberg 2001). Olan (2004) illustrated the interactivity that happens when students use BlueJ below:

Figure 1 BlueJ's Interface



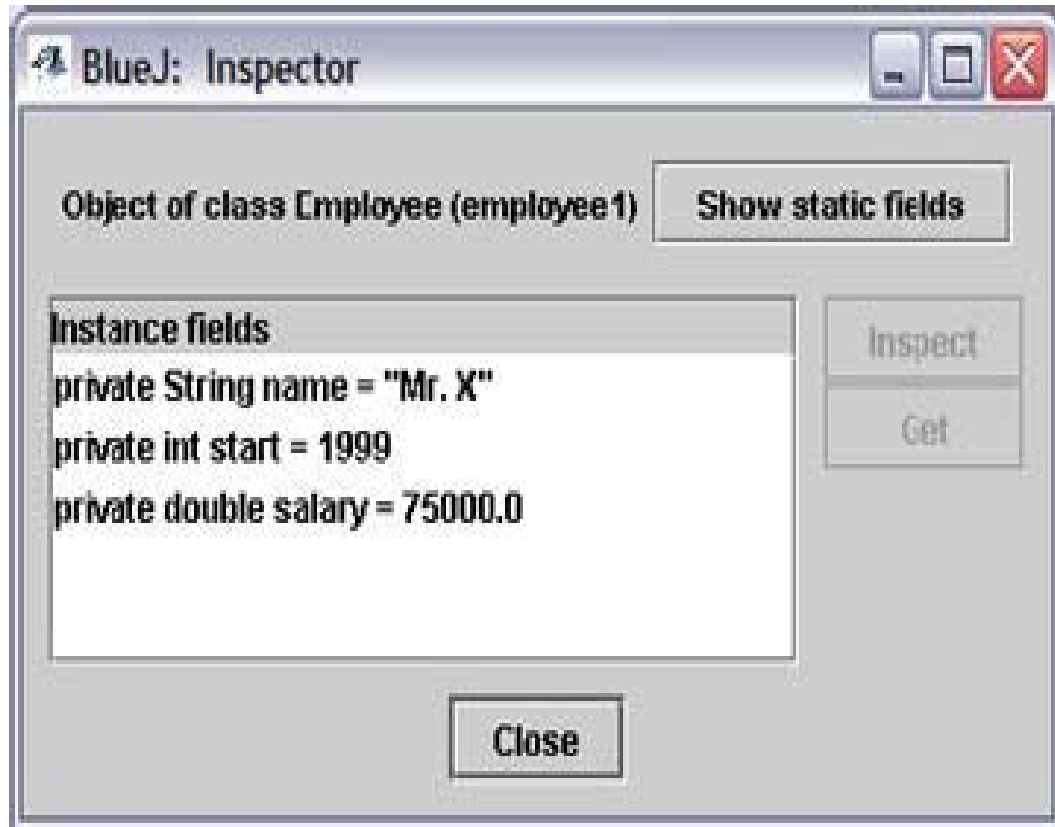
The project as the organisational unit of BlueJ is organised as a directory in a file system. All Java source code for the project resides in this directory. Once a project is defined, students add class/es by clicking the New Class button. The main window displays classes using simplified UML-class diagrams.

Figure 2 BlueJ's Objects



Students select a constructor for a class thereby creating an object. Then a dialog box shows the signature and documentation for the selected constructor, and presents a template for entering the values of arguments.

Figure 3 BlueJ's Inspector



Students create an object to display a UML object diagram in BlueJ's "object bench". By selecting this diagram, the inspector is given an access for viewing the state of the object. These interactive features of BlueJ allow the user to experiment and test the functionality of a class without requiring a test driver.

Sanders et al (2001) summarised BlueJ's characteristics:

- It is free for everybody and for any kind of use
- Runs on top of various versions of Sun's Java Development Kit, including JDK 1.3.
- Easy to use with short learning curve
- Automatic construction of class diagrams
- Customizable templates for class skeletons

- Ability to instantiate objects and test methods without a driver program
- Integrated debugger
- Menu items to preview or create HTML documentation (via javadoc)
- Ability to import packages not created with BlueJ
- Automatic make utility

Another recently developed tool is Alice. Alice is a known 3D programming environment that allows students to implement algorithms by manipulating a wide variety of 3D objects. Alice provides a drag-and-drop interface of 3D objects to facilitate a more engaging, less frustrating programming experience for novice programmers (Gross & Powers 2005). “Alice gives students the opportunity to learn about object-oriented programming concepts without the syntax frustrations imposed by text-based programming languages” (Gordon 2006). Using the lecture materials from CM1010 module of RGU lecturers Roger McDermott, Garry Brindley and Gordon Eccleston, the Alice interface is illustrated below:

Figure 4 Alice's World

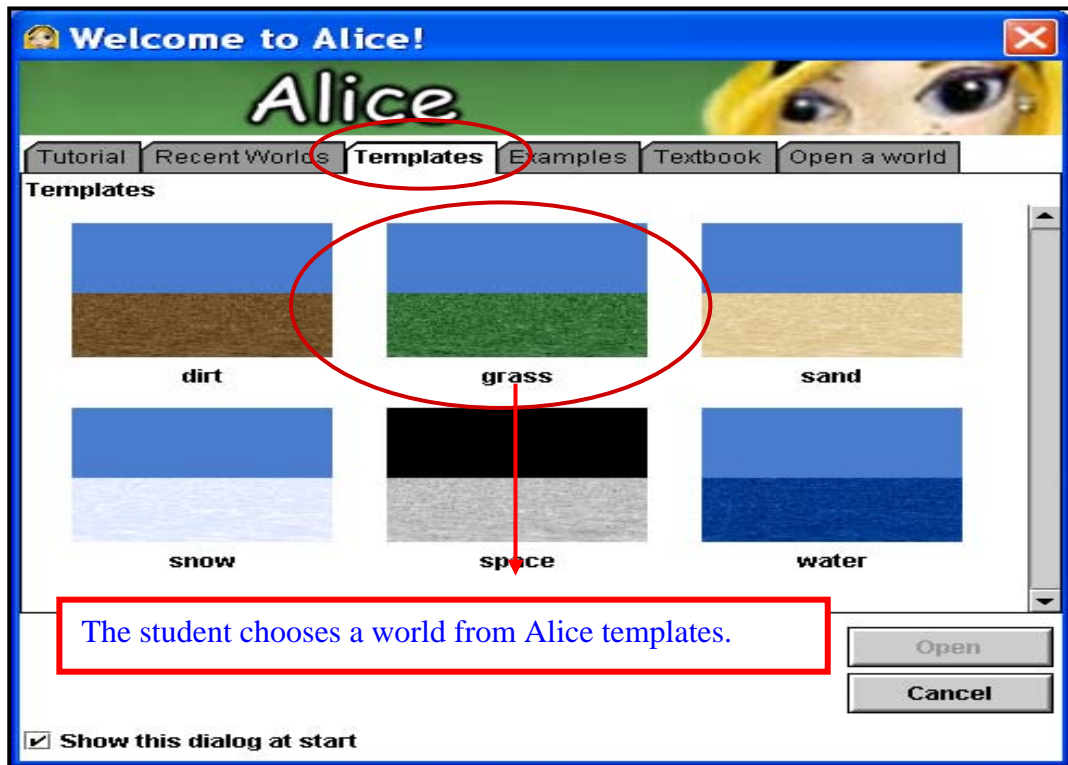


Figure 5 Alice's Objects

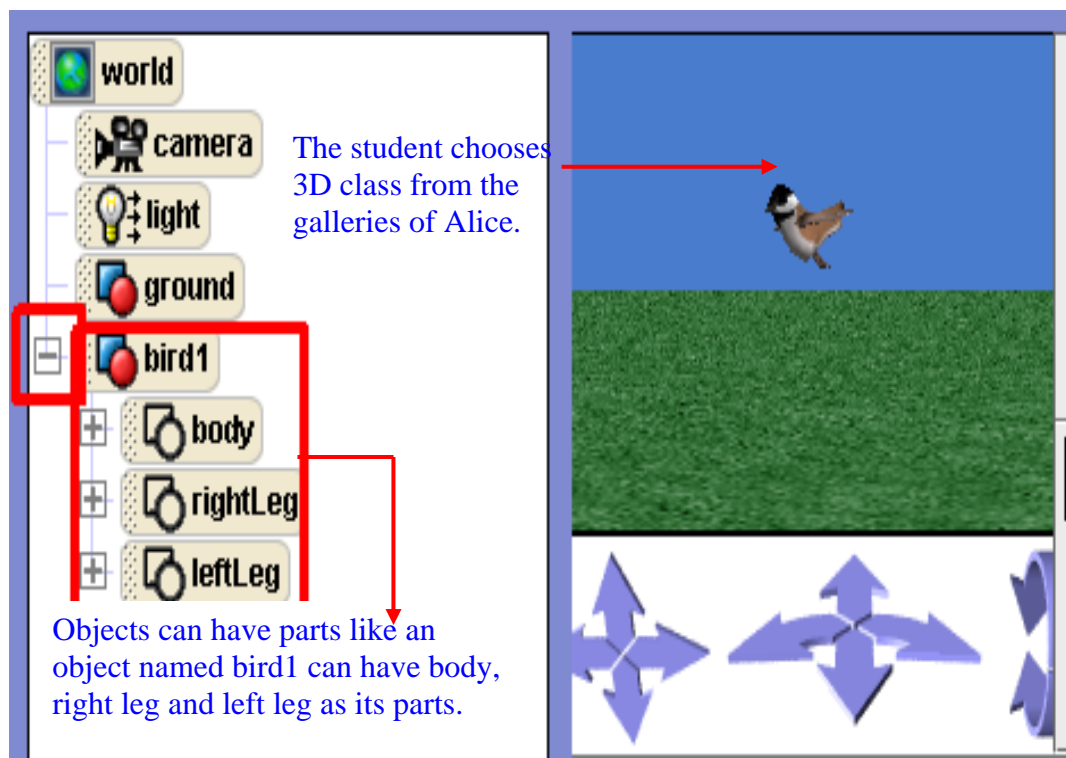


Figure 6 Alice's Methods

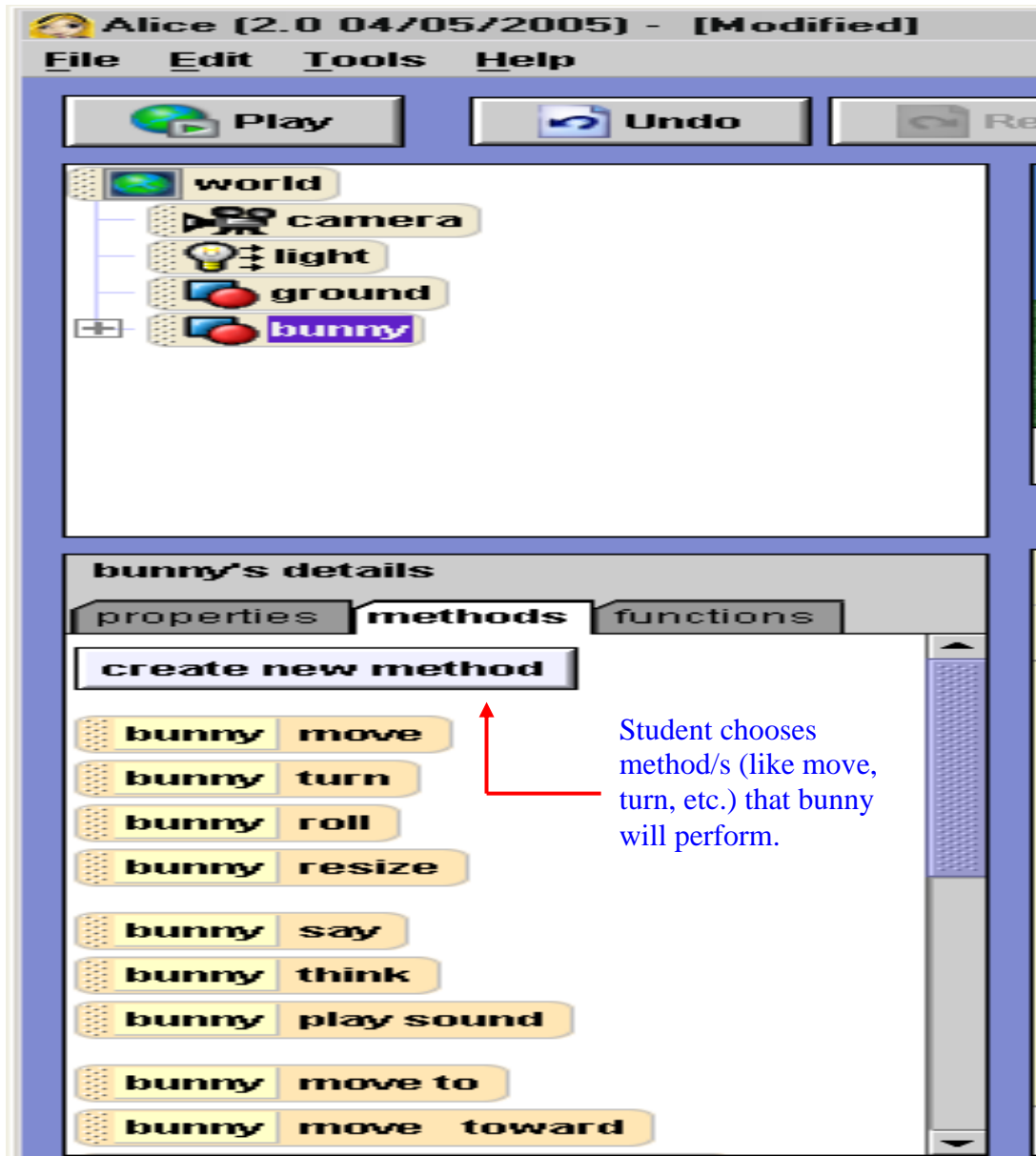


Figure 7 Alice's Functions

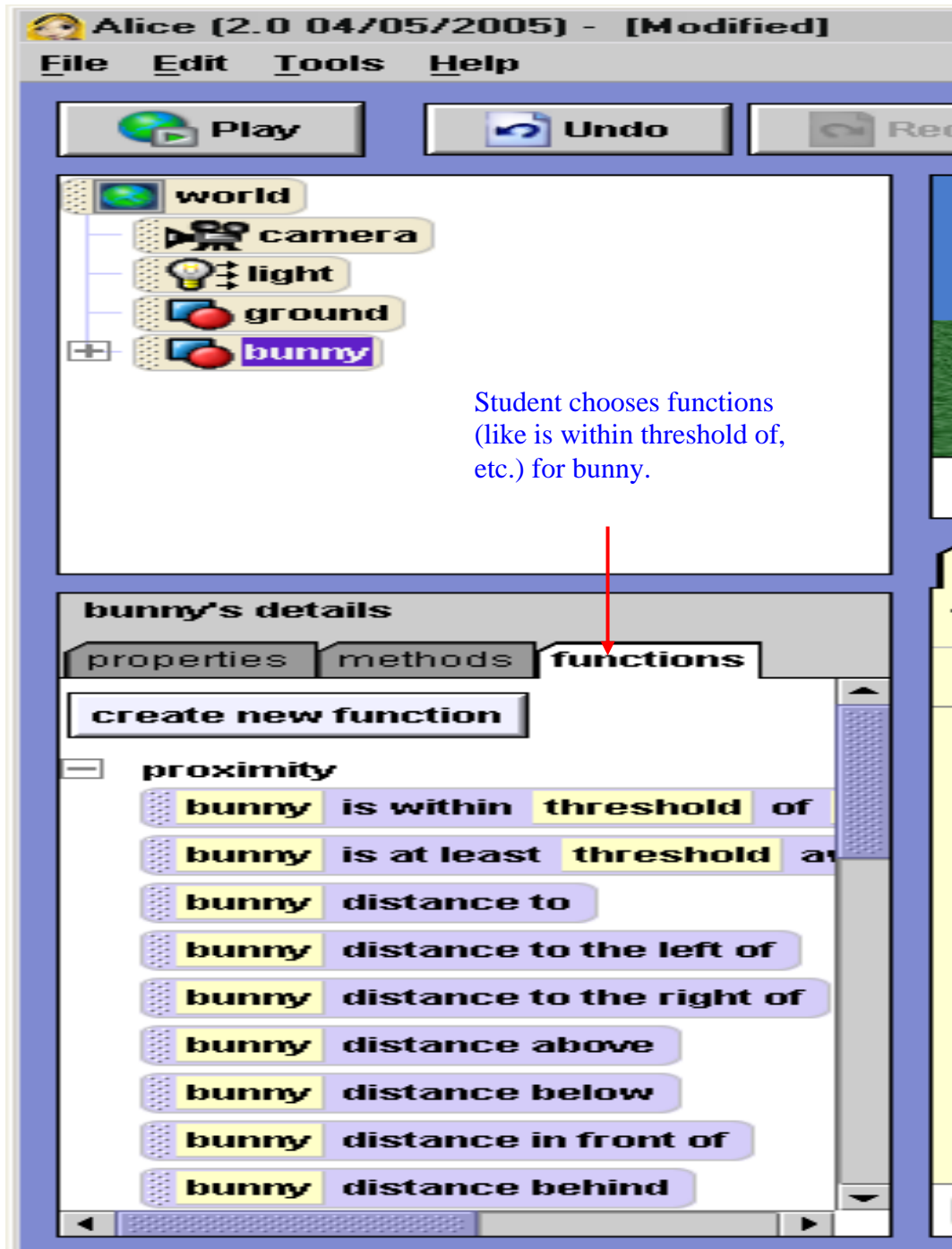
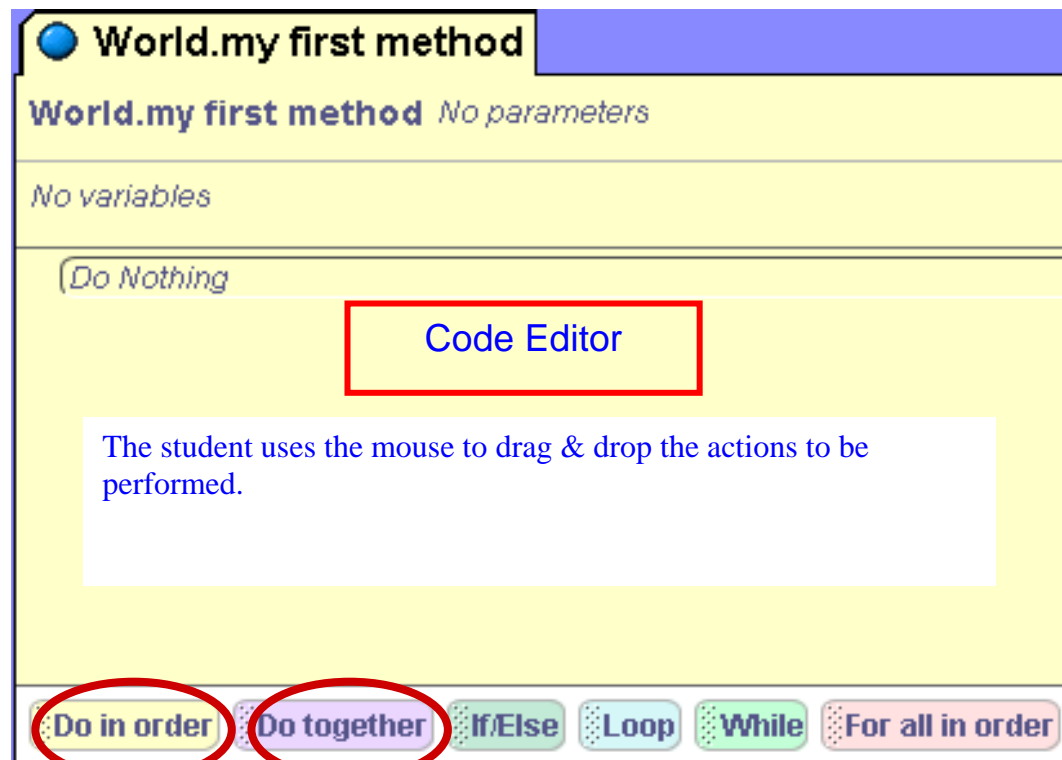


Figure 8 Alice's Interface



The student uses the mouse to drag & drop the actions to be performed.

Sequential Action Block:
actions occur one after another

Simultaneous Action Block:
actions occur at the same time

Through drag and drop arrangement of objects, their associated methods, and standard control constructs, programs are developed in Alice. A student chooses an object in the world and calls one of its methods. A method call is incorporated in their program through a graphical interface in which they drag the name of the method from the object and drop it into the calling method. A textual representation of the method call appears wherever the method is dropped, assuming that it is a valid location. There are no syntax errors since each line of code cannot be edited (Gross & Powers 2005).

CHAPTER 3

REVIEW OF RELATED LITERATURES

This chapter is a review of academic literature concerning the pedagogy of object-oriented programming. This is presented by topic to help in understanding how the problems of the thesis evolved. It started with the question “How can object-oriented programming pedagogy be improved?” Then it asks, “What programming tools are available to help write object-oriented programs using Java?” A further question arises, namely “Which of the two programming tools, text-based or graphical-based, offers the easier learning environment for novice programmers?” All these questions are in some way addressed by the main question the thesis sets out to answer.

Object-Oriented Paradigm

There have been many articles written about the object-oriented programming since its inception. A check of the Association for Computing Machinery’s website and their digital library, lists many articles describing object-oriented programming (Pokkunuri 1989 and reference therein).

The descriptions of object-oriented programming found in these papers range from definitions of the paradigm itself to applications such as array programming (Mougin & Ducasse 2003), databases (Patterson & Haddow 2003), etc. As Rentsch (1982) stated, “My guess is that object-oriented programming will be in the 1980’s what structured programming was in the 1970’s. Everyone will be in favour of it. Every manufacturer will promote his products as supporting it. Every manager will pay lip service to it. Every programmer will practice it (differently). And no one will know just what it is.” The benefits of object-oriented programming were emphasised by Snyder (1986), “Object-oriented programming is a practical and useful programming methodology that encourages

modular design and software reuse”. Object-oriented programming support of the concept of “inheritance” was seen as a key to modular design and code reuse (see Danforth & Tomlinson 1988).

The popularity of this type of methodology prompted universities to review and revise their curricula and integrate object-oriented programming at the start of computing curricula (Or-Bach & Lavy 2003). Its integration into the curricula paved the way for lecturers and teachers to write about their classroom experiences of teaching the subject (Pugh et al 1987; Osborne 1992; Schahczenski 2000). However, this integration was never without problems and difficulties such as reports on problems encountered when programmers move to object-oriented programming (Luker 1994) and of how object-oriented programming is taught (Roumani 2006).

The educational lessons learnt have enriched the pedagogy of object-oriented programming by involving classroom experience in the process and it has given different perspectives on object-oriented pedagogy. One lesson is that it needed a different mindset than did traditional (i.e. procedural) programming (Neubauer & Strong 2002). Lewis (2000) said, “It is a change away from accepted structured analysis, design and programming methodologies toward their counterparts in the object-oriented paradigm”. Another viewpoint was given by Zhu and Zhou (2003), “If people really master the object-oriented programming, they may even program an object-oriented program with a traditional language such as C”.

The inclusion of OOP in the computing curricula has provoked people in the higher education to study how its pedagogy can be improved. In addition, the challenges encountered not only by teachers but also students in learning such concepts have encouraged numerous studies on how the pedagogy can be better enhanced (Hughes & Peiris 2006).

Change in Programming Emphasis

When object-oriented programming was included in the curricula, a change in programming emphasis became evident (Mitchell 2000). This change in emphasis has been a prominent topic of several case studies (Lister et al 2006; Vilner et al 2007). With the use of top down analysis, a set of modules, fixed in sequence and tailored to specifications is identified in the traditional approach. In object-oriented analysis, bottom-up analysis is employed to develop a set of components. These components are combined in different ways to form not only a solution to the problem but a more general solution. These components are then intended to be reusable (Mazaitis 1993).

Multiple studies have also documented the struggles involved in this change in emphasis. One of the problems encountered is the choice of programming language used in teaching (Brilliant & Wiseman 1996; Mitchell 2000). The programming language Java has become the popular choice in writing object-oriented codes during the last decade because of its machine independent platform and its promising program reusable (Special Interest Group on Computer Science Education 2005). Studies have been conducted and suggestions made so the transition from procedural to object-oriented programming can be manageable (Alphonse & Ventura 2002). At the University of Kiel, Berghammer and Huch (2005) wrote, “However, experience has shown that this puts a burden on the students if one starts imperative programming with Java’s overhead of object-oriented notations”.

Moreover, the teaching of programming concepts in general has become more complicated as the acceptance of non-procedural programming increases (Brilliant & Wiseman 1996). Classroom teachers started to find help on how they can cope with the difficulties of object-oriented programming (Kolling & Rosenberg 2001; Zhu & Zhou 2003; Bierre et al 2006). Development of constructivist teaching method arose upon the

realisation that the predominant model of instruction was inadequate for most students since it does not engage the mind appropriately (Hadjerrouit 1999). Not only were the teaching styles affected by the change in programming emphasis but also the learning styles. It was even claimed by Howard et al (1996) that, “It is possible to sustain a course over an entire semester with the lofty goals of reaching the many preferred learning styles of your students and at the same time guiding them to the deeper levels of your subject’s cognitive domain.”

As more research has been done about how the pedagogy of object-oriented programming can be enhanced, the assessment component has also been highlighted. Lister and Leaney (2003) stated, “Decades ago, when Bloom’s taxonomy was first published, the effect of the taxonomy was to highlight that schoolteachers placed too much emphasis on testing knowledge and comprehension. Today, the taxonomy highlights that IT academics place premature emphasis on the higher level of the taxonomy.” They emphasised further that Bloom’s taxonomy needs a mix of strategies, to test students at all levels of the taxonomy (Lister & Leaney 2003). One strategy used by the Open University (OU) is to balance the assessment component is their Tutor-Marked Assignment (TMA), working in partnership with the written textbooks. “Tutors not only mark the assignments but comment on them constructively, thus enabling the student’s written words (and ideas), as well as the teacher’s, to determine the nature and style of the teaching they receive” (Rowntree 2006). However, TMAs are still a teacher-dominated medium, since it is only the tutors that give feedbacks. In addition, most students will write what they think their teachers would like to read (Rowntree 2006).

The realisation that learning to program using this paradigm is never easy has made academics vigilant in coping with the difficulties. In the above literature, the change from procedural to object-oriented programming affected the programming language

choice, teaching and learning, and most recently the assessment component of the programming course has seen some revisions and updates.

Programming Tools

The awareness of the difficulties of the pedagogy of object-oriented programming using Java has driven both academics in universities and software developers in the industrial field to design and create programming tools, which help the novice programmers acquire the necessary skills (e.g. Java Power Tools, Jeroo, Karel). The majority of these tools are window oriented. Kempf and Stelzner (1987) said, “Successful learning of an object-oriented programming style is greatly facilitated by a flexible, window-oriented interface and a step-by-step instructional methodology”.

Two programming tools that are well known in helping to learn object-oriented concepts are BlueJ and Alice. BlueJ is an interactive environment that teaches object-oriented programming to beginners by visualisation and experimentation (Bailie et al 2003). Several papers have presented on how BlueJ can be used to reinforce the basic concepts of object-oriented design (Kouznetsova 2007). Some papers, claim significant improvements in introducing object-oriented concepts (Kolling & Rosenberg 2001). Haaster and Hagan (2004) described that “BlueJ gives students a graphical picture of the classes and objects in a system, allows students to interact with them directly, simplifies testing of methods and classes, and removes the necessity for much difficult and confusing Java code such as the main method in a class”. However, although BlueJ does visualise objects in terms of Unified Modeling Language (UML) type class diagrams, the main process by which code is assembled is textual i.e. students would write Java-code in a text editor which could then be compiled and executed.

Alice is a programming environment that introduces students to computer programming by manipulating objects in a 3D virtual world (Carnegie Mellon University 2006). Dann et al (2003) stressed that through the use of Alice, “Animated program visualisation can be used to support innovative instructional methods for teaching beginners about objects, their behaviour, and state”. It was emphasised that Alice is not a toy since it includes the programming constructs found in general-purpose languages such as Java and C++ and a simple form of parallel programming (Kelleher et al 2007). Alice uses a drag and drop interface to construct code blocks which are then executed producing an animation.

In this present study, BlueJ is classified as text-based programming tool while Alice is the graphical-based programming tool. Wong (2006) supports this classification when he wrote that graphical programming environments show the connection and the objects together in the same view and normally, a simple drag and drop methodology connects two of the objects. He also added that in a text-based environment, the definition of the related objects and the statements that connect them frequently are in different parts of the application. However, it should be noted that elementary graphical-based environments typically also incorporate elementary text-based aspects e.g. it is possible to print to the screen in Alice. In graphical-based environments, designers usually combine the two types of programming code by including text-based code in a text window object (Wong 2006). Furthermore, Alice generally allows storytelling to be incorporated into programming by students creating their characters (3D objects) and program their behaviour (Powers et al 2007).

Summary

When the use of object-oriented programming has become common both in the computing field and universities, we have become aware and conscious that this change in programming emphasis would not be easy for everybody. Problems with the choice of programming language and in the object-oriented pedagogy occur. Not only the teaching and learning but also the assessment needs realignment and adjustment to cope with the demands of the pedagogy of object-oriented programming. Empowering learners to be accountable for what, why and how they would like to learn seems to be an exciting option for object-oriented programming. As Felder and Silverman (1998) put, “Learning in a structured educational setting maybe thought of as a two-step process involving the reception and processing of information. Common sense and reflective thinking become available to students, who select the material they will process and ignore the rest. The second step may involve simple memorization or inductive or deductive reasoning, reflection or action, and introspection or interaction with others. The outcome is that the material is either learnt in one sense or another or not learnt”.

The use of Java in writing object-oriented code has never been easy for novice programmers. Programming tools were created to simplify and improve the writing of code using Java. Window-based tools like BlueJ and Alice were use to ease the transition to writing code in a full Integrated Development Environment (IDE). These programming tools try to make the syntax constructs of Java less complicated not only for the novice programmers but also for programmers who are switching from procedural to object-oriented programming.

CHAPTER 4

RESEARCH QUESTIONS

This chapter presents the main research's questions and discusses why answers to these questions are being sought and the importance of answering them.

The Questions

The pedagogy of object-oriented programming (OOP) has been found to be challenging for both teachers and students requiring new strategies and methodologies as well as innovative learning tools. Thus, the teaching-learning process is in continuous search of a better or more appropriate tools, methods or procedures to use and implement.

However, teaching object-oriented concepts has never been an easy task for educators. Multiple studies and publications (Kolling & Rosenberg 2001; Zhu & Zhou 2003; Bierre et al 2006) have been presented to offer help in teaching object-oriented concepts. The vast majority of these studies and publications have shown that object-oriented pedagogy is never a simple process. The complexities of the OOP pedagogy include the choice of programming language (Brilliant & Wiseman 1996), programming environment (Kempf & Stelzner 1987) and also shifting from procedural to OOP (Adams 1996). Furthermore, the pedagogy of object-oriented programming is continually evolving and needs periodic assessment and revision. With this in mind, we need to answer the following questions:

1. Does the process of learning object-oriented concepts using graphical-based tools differ from using text-based tools?
2. Do graphical-based tools support text-based tools in learning object-oriented concepts?
3. Do graphical-based tools offer more help in understanding object-oriented concepts than text-based tools?

Why answer the questions?

As the review of literature has shown, the change from procedural to object-oriented programming has made the pedagogy of OOP more challenging, not only for the lecturers but also for students. This change not only involved a choice in programming language such as Java but also much greater consideration of the strategies and mechanisms concerning the assessment of skill acquisition in the subject. Programming tools have been designed to help programmers cope with the rigors of the various software development activities. Whether text-based or graphical-based, the ultimate aim of the different programming tools is to enhance learning the object-oriented programming. Furthermore, innovative ways of assessing how students learn object-oriented concepts have also been investigated. Seffah et al (1999) described in their paper a Web-based system that defines training needs for object-oriented developers by identifying the strong and the weak areas of their knowledge and skills. Cable (2001) discussed the use of Primary Trait Analysis (PTA) as an assessment tool.

By answering these questions, it is hoped that the teaching, learning and assessment processes of object-oriented programming can be made more effective and efficient. This will help lecturers to develop, revise, and update the course curriculum.

These changes will hopefully make the curriculum more robust and adaptable.

Specifically, it will be helpful in the:

- Identification and description of teaching methodologies for object-oriented concepts
- Preparation of object-oriented curricula
- Preparation of programming curricula for other, i.e. non-computing, disciplines
- Creation of a working model of assessment for object-oriented concepts
- Identification of new sources of assessing the learning of object-oriented concepts
- Creation of more interactive assessments for object-oriented concepts, and
- Investigation of the appropriate uses of programming tools, either text-based or graphical-based or combination of the two.

These questions and the desire to find answers were brought about by the researcher's own experiences in learning and eventually her experiences teaching object-oriented programming. These difficulties were not limited to teaching and learning object-oriented concepts but later on with their assessment as well. These challenges led to the realisation that the OOP pedagogy would have to be supported with a healthy perspective on assessment to be successful.

CHAPTER 5

METHODOLOGIES

This chapter discusses the research method used to gather the relevant data. It explains the research design, instruments used, construction, validation of the instrument, the data gathering procedures, the programming tools used and the statistical treatment of the data.

Research Design

A combination of the following methodologies has been employed to answer the research question “How do graphical-based and text-based programming tools help students in learning object-oriented concepts”.

One of the research methods used was the Quantitative Research. The researcher collected facts and studied the relationship of one set of facts to another through a survey method. Statistical quantities such as the mean and standard deviation constructed and statistical tests such as the Mann-Whitney Test applied.

Action Research is essentially an on-the-spot procedure designed to deal with a concrete problem located in an immediate situation. This means that the step-by-step process is constantly monitored (ideally, that is) over varying periods of time and by a variety of mechanisms (questionnaires, diaries, interviews and case studies, for example) so that the ensuing feedback may be translated into modification, adjustments, directional changes, redefinitions, as necessary, so as to bring about lasting benefit to the ongoing process itself (Cohen and Manion 1989:223). The researcher with her team of supervisors concentrated on the difficulty of the object-oriented programming pedagogy. The study

hopes to provide feedbacks that will be helpful to the modification and adjustment of the programming curricula.

Respondents of the Study

The respondents of the study are the sixty (30 for each set of questionnaires) students of the course Object-Oriented Programming Techniques at Robert Gordon University under the supervision of Gordon Eccleston, Roger McDermott and Garry Brindley. The class schedule was every Wednesdays and Fridays (11am – 1 pm at C23 laboratory) of the 2nd Semester, SY 2006-2007. Since the research involved human participants, the researcher was guided by the British Psychological Society's "Ethical Principles for Conducting Research with Human Participants" (British Psychological Society 2006) and Robert Gordon University's "Research Governance and Ethics" (Robert Gordon University 2006). The necessary approvals from the class lecturers and paperwork were obtained and submitted. The researcher also visited the class before the actual distribution of the questionnaires. General information about the study and contact options was indicated in the cover letter of the questionnaires.

Questionnaire

The questionnaire was the main instrument used by the researcher to collect data.

This information sought concerned:

- Respondents' computing backgrounds and how they were using computers,
- How the respondents used the two programming tools; Alice for graphical-based tools and BlueJ which is a Java IDE for text-based tools,

- How respondents learnt object-oriented concepts using the graphical-based and the text-based tools and
- Respondents' profile

The author wanted to answer the research question “Does the process of learning object-oriented concepts using graphical-based tool differ from that using text-based tool?” The key tool for this investigation were the questions, “How easy is it to use Alice/Java in completing your coursework?” and “How easy is it to use Alice/Java in understanding the various object-oriented concepts?” Additionally, the data derived from the question, “How easy is it to recover from different errors in Alice/Java?” were also used to answer research question number one.

The second research question “Do graphical-based tools support text-based tools in learning object-oriented concepts”, was addressed by questions, “How easy is it to use Alice/Java in completing your coursework?” and “How easy is it to use Alice/Java in understanding the various object-oriented concepts?” Here the ratings of the respondents gave information on how easy for them to complete their coursework using Alice/Java and also, understand various object-oriented concepts. Also, the question “How easy is it to recover from errors?”

The ratings given by the respondents on the question, “How much has Alice/Java increased your confidence in learning various object-oriented concepts?” were used to answer research question number three “Do graphical-based offer more help in understanding object-oriented concepts than text-based tools?”

The respondents were also asked to complete three sentences and answer two questions about their use of Alice and Java. These questions gave the respondents the opportunity to express their opinions and thoughts using their own words about Alice and BlueJ in an open-ended text. These open-ended questions were also used to tackle the

three research questions, e.g. “Alice is a nice introduction to programming” which addressed qualitatively the transition from Alice to BlueJ/Java.

Construction of the Questionnaires

To allow an objective construction of the instrument, academic literature on questionnaire design was consulted. The World Wide Web was a source of a lot of information concerning the construction of questionnaire. Advice was sought from knowledgeable people in finalizing the questionnaire specifically, the team of module lecturers. The questionnaire was pilot tested by six (6) people before the actual distribution. There were two (2) sets of questionnaires; the Alice (see Appendix A) and the Java (see Appendix B) Questionnaires, each consisting of the following parts:

- Questions about Information Technology Usage
- Respondents pre-knowledge of the programming tools
- Respondents usage of the programming tools
- Respondents’ Comments about their use of the programming tools, and
- Respondents’ Profile

Alice and BlueJ in Object-Oriented Programming

Learning how to program is known not to be an easy task especially in object-oriented programming. Various research has tried to identify the difficulties with which both the teachers and learners need to cope in teaching and learning object-oriented concepts (de Clue 1996; Leavens 1991). Some of the research has identified different programming or pedagogical tools, which have been created to help in the process (Rasala et al 2001).

Java as an object-oriented programming language, has gained popularity during the 1990s. But because of its complicated syntax structures Java has often alienated learners. Java has given the impression that it is a difficult object-oriented language. To lessen this difficulty for both the learners and the teachers, programming tools have been created specifically window-based interfaces. Two of these programming tools were used in the first year programming course, which consisted of two modules, the first semester CM1010 Introduction to OOP (which used Alice) and its successor module CM1011, Object-Oriented Programming Techniques.

Alice as a graphical-based programming tool gained popularity because of animated outputs that lessens the pressure of object-oriented programming. In the module Introduction to OOP, storyboards are created first then Alice is used to introduce the assorted object-oriented concepts.

As explained in Chapter 3 BlueJ is an interactive window-based environment for Java. In the module, Object-Oriented Programming Techniques, Java was slowly introduced to students by using BlueJ as the code development environment. This allowed students to gain familiarity with the object-oriented concepts by first using Alice and then BlueJ to prepare them for the transition to writing codes in a full Java IDE such as Eclipse.

Distribution and Retrieval of the Questionnaires

Both questionnaires were given out and collected in March 2006. The Alice questionnaire was distributed first, and following this, the Java questionnaire was distributed. It would have been better if the Alice questionnaires were distributed in the 1st semester but at that stage, no reasonable hypothesis could be formulated. Random Sampling was used to identify respondents because the researcher could not ensure that students who registered in the 1st semester would also register for the distribution of the

Java questionnaire in the 2nd semester. Every 2nd student in a row was given a questionnaire and approximately 15 minutes were spent on filling up the questionnaire. The questionnaire was gathered and collected by the researcher after the allotted time. The researcher with the support of her team of supervisors retrieved all questionnaires distributed during the 1st and 2nd distribution.

Treatment of Data

The following statistical tools were used in the analysis and interpretation of the collected data.

- **Frequency Distribution and Percentage**

The Frequency Distribution tables give the readers a summary of the responses. The Percentage of responses for each question was used to determine the quantitative relation to the full set of responses. Generally, the frequency distribution and percentage were used in describing the respondents' Information Technology Usage, their use of the programming tools and their profile.

- **Mean**

The Mean was used to describe the perception of the respondents on each indicator. It was interpreted using the Likert scale concept. As an example of this, consider the question concerning Computer and Internet Usage. Respondents were asked to categorise their responses concerning computer and Internet usage using the following scale:

Computer & Internet Usage:

Interpretation	Weight
Everyday	1
At least once a week	2
At least once per fortnight	3
At least once per month	4
Less often	5
Never	6

Similarly, in the question about respondents' familiarity with the programming tool/language terms:

Familiarity with Terms:

Interpretation	Weight
Very Familiar	1
Familiar	2
Likely Familiar	3
Less Familiar	4
Not at all	5

- **Standard Deviation**

The mean together with the standard deviation were used to find out students' perception of how familiar they were with the terms for both programming tools, how they were using Alice and BlueJ in their coursework, how subsequently they worked in Alice and BlueJ. And finally, how confident each of the programming tool made them feel in learning object-oriented concepts.

Mann-Whitney Test

The Mann-Whitney test, also called the rank sum test, is a nonparametric test that compares two unpaired groups (Graphpad Software 2006). This was not a matched

sample since two independent samples, one from each population, were used (Anderson et al 1993:721). The Mann-Whitney Test was used to compare how students used Alice and BlueJ in their coursework, how easy it was for them to understand the object-oriented concepts, how easy it was for them to recover from errors and how each of the programming tool increased their confidence in learning object-oriented concepts.

Statistical Package for Social Sciences (SPSS)

The main computer package used in this study was SPSS (originally, Statistical Package for the Social Sciences). The researcher manipulated and processed the data gathered from the questionnaires using SPSS. Specifically, it was used to create frequency distribution and percentage tables, the mean and standard deviation and to test whether there was difference between the two populations using Mann-Whitney nonparametric method (Anderson et al 1993:721).

CHAPTER 6

PRESENTATION, RESULTS and ANALYSIS

This chapter presents the quantitative and qualitative data from the questionnaires and its corresponding analysis and interpretation.

Table 1 Respondents' Age and Gender

Age Category	Respondents' Gender									
	Alice				BlueJ					
	Male	%	Fem	%	Male	%	Fem	%		
Less than 18	1	3.33%			1	3.33%				
18 – 21	21	70.00%	4	13.33%	18	60.00%	5	16.67%		
22 – 25	2	6.67%			2	6.67%				
26 – 30	1	3.33%			2	6.67%				
31 – 35							1	3.33%		
35 +			1	3.33%			1	3.33%		
	25	83.33%	5	16.66%	23	76.67%	7	23.33%		
Total	30			100%		30			100%	

Table 1 presented the frequency and percentage distribution of the student respondents according to age and gender. For both questionnaires (Alice and BlueJ) majority of the student respondents came from the age category of 18 to 21 and the males dominated females in number. As expected most of the respondents were in the age bracket 18 to 21 since the course Object-oriented Programming Techniques is offered during the 1st year of their programs. There were a couple of mature

respondents belonging to the age bracket of 26 and above. Male respondents dominated female respondents by more than 50% in this survey.

Table 2 Respondents' Course

Course Title	Respondents' Course			
	Alice		BlueJ	
	Count	%	Count	%
Internet & Multimedia	6	20.00%	7	23.33%
Graphics & Animation	14	46.67%	15	50.00%
Business & e-Commerce			1	3.33%
Computer Science	7	23.33%	4	13.33%
Computing & Information	3	10.00%	3	10.00%
Total	30	100%	30	99.99%

Table 2 depicted the frequency and percentage distribution of the student respondents according to their course. Graphics and Animation had the most number of students for both questionnaires (Alice – 46.67% and BlueJ – 50.00%). Business & e-Commerce had 1 for the BlueJ questionnaire and none for Alice questionnaire. Computing & Information for both questionnaires had 10.00%. There were slightly more Computer Science students responding to the Alice questionnaire (23.33%) than BlueJ (13.33%). Internet and Multimedia had 20.00% for Alice and 23.33% for BlueJ. There

were no Computing for Intelligent System enrolled for both groups. Noticeably, Graphics & Animation students dominated the respondents profile in terms of course.

Table 3 Respondents' Mathematics Attainment

Mathematics Qualifications	Respondents' Mathematics Attainment			
	Alice		BlueJ	
	Count	%	Count	%
Standard Grade	5	16.67%	7	23.33%
Higher Grade	9	30.00%	9	30.00%
Others	8	26.67%	8	26.67%
All	1	3.33%	5	
Standard Grade & Higher Grade	4	13.33%		16.67%
Standard Grade & Others	3	10.00%	1	3.33%
Total	30	100%	30	100%

Table 3 showed that for both the Alice and BlueJ questionnaires, 30.00% (majority of the respondents) of the students had Higher Grade Mathematics qualification. Other Mathematics qualifications (e.g. modules they have taken at their first universities) for both questionnaires garnered 26.67%. For Alice questionnaires, 13.33% have both Standard and Higher Grades and for BlueJ it was slightly higher 16.67%. Respondents had the mathematical qualifications deemed acceptable by the university.

Table 4 Respondents' Programming Experience

Programming Years	Respondents' Programming Experience			
	Alice		BlueJ	
	Count	%	Count	%
None at all	2	6.67%	1	3.33%
Less than 1 year	9	30.00%	10	33.33%
More than 1 year	19	63.33%	19	63.33%
Total	30	100%	30	99.99%

Table 4 showed that 63.33% of the respondents had more than one-year programming experience for both groups. Less than 10% of the respondents had no programming experience at all. Those who had less than one (1) year programming experience for both questionnaires were in the range 30% - 34%. It seemed like the majority of the respondents already had programming experience for over a year. The questionnaire did not include questions that would evaluate and categorise the kind of programming to which the respondents had been exposed.

Table 5 Respondents' Computer and Internet Exposure (Alice)

How often?	Computer Usage	Command Prompt	WWW	E-mail	News groups	IM	FTP
	%	%	%	%	%	%	%
Everyday	96.67%	3.33%	96.67%	93.33%	6.67%	76.67%	26.67%
At least once a week	3.33%	13.33%	3.33%	6.67%	10.00%	10.00%	50.00%
At least once per fortnight		10.00%			20.00%	3.33%	6.67%
At least once per month		6.67%			13.33%		10.00%
Less often		33.33%			23.33%	10.00%	6.67%
Never		33.33%			26.67%		
Total	100%	100%	100%	100%	100%	100%	100%

Table 6 Respondents' Internet Exposure (BlueJ)

How often?	Computer Usage	Command Prompt	WWW	E-mail	News groups	IM	FTP
	%	%	%	%	%	%	%
Everyday	93.33%	10.00%	93.33%	83.33%	10.00%	53.33%	23.33%
At least once a week	6.67%	10.00%	6.67%	16.67%	6.67%	26.67%	33.33%
At least once per fortnight					10.00%		20.00%
At least once per month		6.67%			23.33%	3.33%	10.00%
Less often		50.00%			23.33%	6.67%	10.00%
Never		23.33%			26.67%	10.00%	3.33%
Total	100%	100%	100%	100%	100%	100%	100%

Tables 5 and 6 described the respondents Internet exposure and how often they use the different services of Internet. A good percentage of the respondents (averaging 95.00%) used computers and browse the World Wide Web everyday and only 5% in average used computers once a week as well as browsing the World Wide Web. An average of 88.33% of the respondents used the Electronic Mail everyday and 11.67% once a week use the Electronic Mail. An average of 65% used Instant Messaging, 25% (average) downloaded and uploaded files and 8.34% (average) joined Newsgroups and only 6.66% (average) used the command prompt everyday. The following percentages apply to the Internet Services that student respondents have never used; 26.67% had never

joined Newsgroups, 5% had never used Instant Messaging and 1.66% had never uploaded or downloaded files. The command prompt had never been used by 28.33% (average) of student respondents.

Table 7 Respondents' Previous Knowledge of the Tool/Programming Language

Answer	Have you heard of the programming tool before using it in the laboratory?			
	Alice		Java	
	Count	%	Count	%
Yes			27	90%
No	30	100%	3	10%
Total	30	100%	30	100%

Table 7 showed respondents' knowledge of the Alice programming tools and the Java programming language. None of the respondents had heard of Alice while only 10% of the BlueJ respondents had not heard of Java before using it in the laboratory. Ninety percent (90.00%) heard Java from varied sources (e.g. previous school/courses/employment, internet, mobile and online games, TV programs & from friends) before using it in the laboratory. And a few (2 to 3 respondents) had used it in their previous employment or they have been trained to use Java.

Table 8 Length of Respondents' Use of the Programming Tools

Duration	How long have respondents used the programming tool/language?			
	Alice		Java	
	Count	%	Count	%
Less than 3 months	6	20.00%	14	46.67%
3 – 6 months	23	76.67%	11	36.67%
7 – 9 months	1	3.33%	2	6.67%
10 – 12 months			1	3.33%
More than 1 year			2	6.67%
Total	30	100%	30	100%

Table 8 presented how long respondents had used both Alice and BlueJ. Most respondents used Alice in 3 to 6 months while 46.67% of BlueJ respondents had used Java for less than 3 months. The difference was clearly accounted for by the timing of the questionnaire distribution. One to two respondents had used Java for a year or more than a year. There were, however, no questions to probe how these respondents used Java, whether the Java usage was academic or professionally (i.e. if they have used it to write application programs).

Table 9 Respondents' Familiarity with the Programming Tools' Interfaces

Duration	How long respondents got familiarised with the programming tools' interfaces?			
	Alice		BlueJ	
	Count	%	Count	%
Less than 1 week	11	36.67%	7	23.33%
1 – 2 weeks	17	56.67%	13	43.33%
3 – 4 weeks	2	6.67%	4	13.33%
More than 1 month			6	20.00%
Total	30	100%	30	100%

Table 9 answered the question about how long the respondents take before they felt familiarised with the Alice and BlueJ interfaces. In both questionnaires, the respondents had answered 1 to 2 weeks before they had gotten use to each interface (56.67% for Alice and 43.33% for BlueJ). Nobody from the Alice questionnaire took more than 1 month and there were six respondents in BlueJ who said that it has taken them more than 1 month.

Table 10 Respondents' Familiarity with Programming Tools' Terms (Alice)

Alice Terms	Mean	Std. Deviation
Interactive	1.57	0.679
3D Objects	1.67	0.922
Visualisation	1.87	0.776
Animation	1.47	0.730
Drag & Drop	1.17	0.379
Debugging	1.80	0.997
World	1.40	0.621
Create New Event	1.47	0.681
Create New Variable	1.43	0.679
Camera	1.53	0.730

Table 11 Respondents' Familiarity with Programming Language' Terms (Java)

Java Terms	Mean	Std. Deviation
Command Prompt	1.70	0.915
JVM	3.87	1.306
Compiler	1.50	0.731
Byte Code	3.07	0.944
Run	1.47	0.819
Projects	1.57	0.679
IDE	3.40	1.354
Java Libraries	2.30	1.055
Java Editors	1.90	0.960
Debugger	2.03	1.098

Tables 10 and 11 contained words, which were commonly encountered when using Alice and BlueJ. Using the mean and standard deviation, the tables depicted how familiar student respondents were with the listed terms. In Alice's terms, student respondents said they were very familiar with all the listed terms. In BlueJ, student respondents put the following terms (JVM, Byte Code, and IDE) in the middle of the scale, which indicated less familiarity with them. The student respondents rated the terms Java Libraries and Debugger as just familiar.

Table 12 Mean & Standard Deviation: Alice & BlueJ in Coursework

Coursework	Mean	Std. Deviation
Alice	1.90	0.759
BlueJ	3.33	1.373

Table 13 Mann and Whitney Test: Alice & BlueJ in Coursework

	Progg Envi	N	Mean Rank	Sum of Ranks
Coursework	Alice	30	21.33	640.00
	BlueJ	30	39.67	1190.00
	Total	60		

	Coursework
Mann-Whitney U	175.000
Wilcoxon W	640.000
Z	-4.218
Asymp. Sig. (2-tailed)	0.000

Tables 12 & 13 presented how easy or difficult it was for respondents to use the environment Alice or BlueJ using a 5-point Likert scale with 1 representing “easy” and 5 representing “difficult”. Alice respondents’ mean is 1.90, indicating that the respondents deemed it easy to complete their coursework while BlueJ respondents gave an intermediate value of 3.3 indicating that it was neither easy nor difficult to complete their coursework using BlueJ. The student respondents found using Alice easier than using the Java-based IDE BlueJ in their coursework completion. Also, Alice’s standard deviation

was smaller, which showed that the ratings of the respondents were nearer to its mean than BlueJ hence demonstrating consistency in the ratings of Alice. The mean with the standard deviations were used to find out whether the use of graphical-based differs from text-based programming environment in learning object-oriented concepts, i.e. research problem number two. By comparing the means and standard deviations, it is found out that respondents considered Alice the simpler introduction to learning object-oriented concepts and served as support to using BlueJ in learning object-oriented concepts.

Using the Mann-Whitney Test to compare whether the two groups differ when using Alice and BlueJ to complete their coursework, the following results were obtained: The null hypothesis in the Mann-Whitney analysis was that there was no significant difference between the use of Alice or BlueJ in coursework completion. At 5% significance level, the test was done with the parameter $z = 0.00$, $p < 0.05$. The result of the test was that the null hypothesis was rejected, and thus, there was significant difference in how students used Alice and BlueJ in their coursework completion. The result from the Mann-Whitney test was applied to differentiate learning object-oriented concepts using a graphical or text-based programming tool which is problem number one. The result pointed to difference between using Alice and BlueJ in completing the coursework.

Table 14 Understanding Object-Oriented Concepts Using Alice & BlueJ

Object-oriented Concepts	Alice		BlueJ	
	Mean	SD	Mean	SD
Class	1.80	0.71	1.77	0.97
Object	1.60	0.62	1.90	1.03
Method	1.60	0.62	1.83	1.05
Message Passing	2.13	0.94	2.77	1.10
Inheritance	2.52	0.95	2.50	1.22
Encapsulation	2.93	1.33	3.63	1.18
Polymorphism	2.79	1.26	3.67	1.14

Table 14 described how easy or difficult it was for respondents to use Alice and BlueJ to understand object-oriented concepts. Table 14 gave the impression that as the concepts become complicated the difficulty of using both programming tools increases. The standard deviations for the BlueJ questionnaires were higher than for Alice in all of the object-oriented concepts except for Encapsulation and Polymorphism. Thus, there were greater variations in the respondents' answers using BlueJ than Alice with almost all of the object-oriented concepts except Encapsulation and Polymorphism. The difference in the means of the two groups did not exceed a value of 0.5 for the following object-oriented concepts: Class, Object, Method, and Inheritance while for the following: Message Passing, Encapsulation and Polymorphism, difference between the means of the two groups is greater than 0.5. Since the means of Alice and BlueJ are not far away from each other, it could be argued that the use graphical-based tool does not impede learning with text-based tools i.e. research problem number two.

Table 15 Mann and Whitney Test for Understanding Object-Oriented Concepts

	UnderClass	UnderObject	UnderMethod	UnderMsgPas
Mann-Whitney U	411.50	399.00	428.00	303.50
Wilcoxon W	876.50	864.00	893.00	768.50
Z	-0.61	-0.82	-0.36	-2.25
Asymp. Sig. (2-tailed)	0.54	0.41	0.72	0.024

	UnderInherit	UnderEncap	UnderPoly
Mann-Whitney U	418.50	261.00	228.00
Wilcoxon W	883.50	667.00	634.00
Z	-0.26	-2.02	-2.59
Asymp. Sig. (2-tailed)	0.80	0.043	0.010

Table 15 tested the understanding of students in learning object-oriented concepts using Mann and Whitney Test. The null hypothesis that there was no significant difference in understanding the various object-oriented concepts using Alice or BlueJ. The following results were obtained:

The test results were not statistically significant at the 5% level for the following object-oriented concepts: class with $z = 0.54$ and $p > 0.05$; objects with $z = 0.41$ and $p > 0.05$; methods with $z = 0.72$ and $p > 0.05$; inheritance with $z = 0.80$ and $p > 0.05$. Thus, the null hypotheses were accepted.

For the following object-oriented concepts at 5% significance level: message passing with $z = 0.024$ and $p < 0.05$; encapsulation with $z = 0.043$ and $p < 0.05$; polymorphism with $z = 0.010$ and $p < 0.05$, the null hypothesis were rejected. There were significant differences in understanding these object-oriented concepts.

The numerical results derived from the survey showed that the use of Alice or BlueJ by student respondents in understanding the object-oriented concepts of Class, Object, Method and Inheritance do not appear to differ. However, the use of Alice or BlueJ to the concepts of Message Passing, Encapsulation and Polymorphism does appear to make a difference. This information contributed to the answer of the research question “Does the process of learning object-oriented concepts using graphical-based tools differ from using text-based tools?”

Table 16 Mean for Coping with Errors

Types of Errors	Alice		BlueJ	
	Mean	SD	Mean	SD
System Error Messages	3.21	1.11	2.73	1.02
Output	2.07	1.08	2.72	1.25
Drag & Drop / Syntax Errors	2.53	1.04	2.33	0.96

Table 16 showed how respondents coped with errors using the programming tools. The derived means and standard deviations were used to find out whether graphical-based tools supported text-based tools in learning object-oriented concepts. For System Error Messages, Alice had a mean value of 3.21 and standard deviation of 1.11 while BlueJ’s mean was 2.73 and its standard deviation was 1.02. It seemed that BlueJ respondents found it easier to recover from System Error Messages than Alice respondents. The respondents also, found it easier to deal with Java’s syntax error (mean was 2.33) than Alice’s drag and drop capability (mean was 2.53). This may be because the respondents have been exposed first to Alice, giving them time to get use to Java’s error messages and

may have been due to the fact that the errors in Alice occurred only when very serious or catastrophic problems occurred with the system. Recovering from wrong output expectedly was much easier in Alice than in BlueJ. This may be because of the immediacy of Alice's animated output in contrast to BlueJ where users do not see any visual representation of the change of state of the system.

Table 17 Mann and Whitney Test for Coping with Errors

	System Err	Output	SynErDrgDrp
Mann-Whitney U	329.00	296.50	401.00
Wilcoxon W	794.00	761.50	866.00
Z	-1.68	-2.20	-0.77
Asymp. Sig. (2-tailed)	0.093	0.028	0.44

Table 17 depicted the result of Mann and Whitney Test. To aid in differentiating the use of the two programming tools in learning the various object-oriented concepts, the following hypotheses were tested at the 5% significance level:

- The null hypotheses that there were no significant difference in recovering from system errors (with $z = 0.093$ & $p > 0.05$) and syntax errors (with $z = 0.44$ & $p > 0.05$) using Alice or BlueJ were accepted.
- The null hypothesis that there was significant difference in recovering from output errors using Alice or BlueJ was rejected with $z = 0.028$ and $p < 0.05$.

How student respondents coped with system error messages and drag & drop/syntax errors using Alice or BlueJ did not appear to differ but when it came to coping with output errors, the two groups were different. It may be that the difference was due to

relatively user-friendly errors in Alice rather than the absence of error information in BlueJ.

Table 18 Mean & Standard Deviation of Respondents Confidence in Learning Object-Oriented Concepts

Object-oriented Concepts	Alice		BlueJ	
	Mean	SD	Mean	SD
Class	2.47	1.14	2.20	1.03
Object	2.43	1.30	2.27	1.05
Method	2.20	1.38	2.37	1.13
Message Passing	2.90	1.12	2.73	1.05
Inheritance	2.77	1.16	2.67	1.12
Encapsulation	3.28	1.31	3.45	1.30
Polymorphism	3.34	1.17	3.55	1.27

Table 18 described how Alice and BlueJ increased respondents' confidence in learning object-oriented concepts. The differences in the means of the two groups did not exceed 0.5 and in all object-oriented concepts, Alice has a higher standard deviation than BlueJ except for Polymorphism. Among the object-oriented concepts, Encapsulation and Polymorphism gained lesser confidence with the use of either Alice or BlueJ. It can be argued that the values of the means and standard deviations being close together show that using Alice or BlueJ gave the respondents the same confidence level in learning object-oriented concepts.

Table 19 Mann and Whitney Test for Confidence in Learning Object-Oriented Concepts

	ConClass	ConObjct	ConMeth	ConMsgP
Mann-Whitney U	390.00	431.50	389.50	427.50
Wilcoxon W	855.00	896.50	854.50	892.50
Z	-0.94	-0.28	-0.93	-0.35
Asymp. Sig. (2-tailed)	0.35	0.78	0.35	0.73

	ConInherit	ConEncap	ConPoly
Mann-Whitney U	428.50	389.50	374.00
Wilcoxon W	893.50	824.50	809.00
Z	-0.33	-0.50	-0.74
Asymp. Sig. (2-tailed)	0.74	0.62	0.46

Table 19 presented the following results when Mann and Whitney Test was used to test if using Alice or BlueJ there was a difference with the student respondents' confidence of learning the different object-oriented concepts. The null hypothesis that there was no significant difference in confidence with the use of either Alice or BlueJ in learning the various object-oriented concepts was accepted at the 5 % significance level. These figures were obtained for the following object-oriented concepts: class with $z = 0.35$ and $p > 0.05$; objects with $z = 0.78$ and $p > 0.05$; methods $z = 0.35$ and $p > 0.05$; message passing $z = 0.73$ and $p > 0.05$; inheritance $z = 0.74$ and $p > 0.05$; encapsulation $z = 0.62$ and $p > 0.05$; polymorphism $z = 0.46$ and $p > 0.05$.

For all object-oriented concepts, there was no significant difference between the uses of Alice or BlueJ in increasing student respondents' confidence. The mean, standard deviations were calculated and Mann-Whitney test employed to answer the third research

question, i.e. to find out which of the two programming environments (graphical-based or text-based) offered more help in understanding object-oriented concepts. Finding out which of the two programming environments increased the confidence of the student respondents in learning object-oriented concepts was an important result which could be used to answer the third research question.

The respondents were also asked open-ended questions in the form of sentence completion. Below is the summary of their answers.

1. My most difficult experience using Alice/Java was when.....

Three (10%) Alice respondents and one (3.33%) Java respondent wrote that their coursework was difficult. The major concern for these Alice respondents was the difficulty in using the drag and drop interface, positioning the camera and synchronising objects. The most common difficult experience reported with Java was syntax and logical errors. The usual errors in Alice have to do with interaction with the interface whereas the errors in BlueJ concern programming activities themselves.

2. My most exciting experience using Alice/Java was when.....

For both questionnaires, Alice – four (13.33%), Java – two (6.67%), coursework completion was rated as their most exciting experience. Additionally, a respondent wrote that Java is not exciting at all.

3. My most rewarding experience using Alice/Java was when.....

Five (16.67%) Alice and three (10%) Java respondents wrote their most rewarding experience was to see the output of their coursework.

4. What have you learnt from Alice/Java?

The respondents recognised that both programming environments were used to learn various object-oriented concepts. Although, six (20%) Alice respondents put the emphasis on the concepts of class, object, methods, inheritance and instance. Two (6.67%) Java respondents emphasised class, attribute, methods and links with other classes. Three (10%) of Alice's respondents learnt also how to create simple animations through storyboarding, the importance of adding comments to their programs and UML. They also indicated their personal preference with a couple (6.67%) of them writing Alice is better, easier to learn and it was a good introduction to programming. Four (13.33%) of the respondents realised that programming is difficult and complicated but fun and rewarding. Two (6.67%) Java respondents were reminded to watch for syntax errors and they wrote Java made them less confident about programming.

5. What features are missing from Alice/Java?

Three (10%) Alice's respondents stated that they would like to receive clearer error messages and a better user interface. They would also like to create their own 3D objects and more sophisticated animations (e.g. like in the movies). For Java's features, two (6.67%) respondents wanted better and easy to understand help functions and a syntax library.

CHAPTER 7

CONCLUSIONS and SUGGESTIONS for FURTHER WORK

This chapter presents conclusions of the study, lists the contributions in the field of research and ask the questions for future research.

Summary

The study was conducted to answer the question “How do the different programming tools (graphical-based and text-based) are used by students to learn object-oriented concepts?” In answering the question, the researcher used a combination of Quantitative, Descriptive and Action Research in the design of the study. It used the survey method to gather the relevant data through questionnaires in one of the computing classes of Robert Gordon University entitled Object-oriented Programming Techniques. Random sampling was used to identify respondents to the questionnaires. There were a total of sixty respondents: thirty for graphical-based (Alice questionnaire) and thirty for text-based (Java questionnaire). For both programming tools: Alice and BlueJ, selections of these programming tools and learning environments were based on functionalities and their ubiquitousness in the academe. The following statistical tools were used to analyse the data gathered: Frequency Distribution and Percentage, Means as the measure of central tendencies and Standard Deviation as a measure of dispersion. Mann and Whitney test was used to assess whether two samples of observations come from the same distribution. The Statistical Package for Social Science (SPSS) has been used to generate the manipulated data. Generally, tables were used to visualise and present the quantitative data gathered from the survey.

Findings and Contributions

“Do learning object-oriented concepts differ using a graphical-based programming tool from using a text-based programming tool?”

The data gathered and its analysis show that the use of graphical-based or text-based programming tools in understanding object-oriented concepts do appear to differ. Moreover the Mann-Whitney Test found that use of the graphical-based (Alice) or the text-based (BlueJ) programming tools indeed lead to differences in understanding the object-oriented (OO) concepts of Message Passing, Encapsulation and Polymorphism. In understanding Class, Object, Method and Inheritance, there was no significant difference for these object-oriented concepts with the use of graphical-based (Alice) or text-based (BlueJ) programming tools. However, it should be remembered that a student's understanding of the OO concepts of Class, Objects, Methods, etc. which they expressed in the second questionnaire are built upon foundations laid in Alice. Consequently, the two questions may not start from the same base. It does show that more works need to be done on looking at how elementary OO concepts such as class, methods, etc. which students learn in the Alice environment, can best be transferred to the (somewhat) more sophisticated coding environment of BlueJ. There are other points which should be made. It may have been that the survey was taken too early in the second semester to allow a considered reply in terms of the latter OO concepts. Also, Alice does not really specify such concepts as encapsulation so questionnaire results given for this may not be accurate. Polymorphism is covered to a certain extent because all objects come with standard methods but care needs to be taken to interpret the results sensibly. A revision to the first research question, as shown above, might be helpful in the gathering of more appropriate data which would differentiate more clearly between the use of the two programming environments in learning object-oriented concepts.

This finding has a direct effect on how course curriculum for object-oriented paradigm can be designed, revised and updated appropriately. Knowing that students understand the concepts using graphical-based or text-based programming tools differently then we now have ideas when to use each of the programming tool given a scenario. For example, if we can identify that students are learning the object-oriented concepts (i.e Message Passing, Encapsulation and Polymorphism) better using a text-based programming tool then we can emphasise the use of these tools during the discussions and usage of the mentioned concepts. In general, if we can identify which object-oriented concept to emphasise when using graphical-based or text-based, then the teaching and learning of these concepts may be more effective and efficient. However, since this is an initial investigation, the researcher suggests a more systematic investigation of the OO concepts in both programming environments.

“Do graphical-based tools support text-based tools in learning object-oriented concepts? “

The data gathered were indicative of the idea that students in learning object-oriented concepts need to have a gentle introduction through the use of graphical-based programming tools and reinforce these learning using text-based programming tools. Using the mean for understanding the different object-oriented concepts using Alice and BlueJ, the values computed were not that far away from each other (e.g. the mean for Alice and BlueJ in understanding class was 1.80 and 1.77 respectively; the highest mean difference would be for Polymorphism 2.79 for Alice & 3.67 for BlueJ which is 0.88). This maybe because the student respondents have started with an animated output using Alice and then used BlueJ as their Java-based IDE. To put it another way, it may be the case that students should gain some confidence with the use of a visual programming tool like Alice before introducing object-oriented concepts, reinforce this with use of a Java-

based IDE, e.g. BlueJ, then complete the transition to Java. But the transition should be a rigorous one, meaning the students while enjoying the use of graphical-based programming tools like Alice would have to be reminded that the interface to coding in Java is of a different kind and that it may be initially more difficult although, in the long run, this level of difficulty is made up for by the flexibility and power of the interface. Thus, students should be prepared for an initial phase in which they study a programming language at a simpler level. It may be necessary to reduce the initial expectations of proficiency in Java (after Alice) in order to prepare them for the transition. It might also be noteworthy to emphasise that Alice is not a sophisticated animation tool that produces animations for movies. Students who were expecting more sophisticated animation software, like that used in the movies (were a little bit frustrated by the rather simple animations of Alice).

The study has given some information when in the course curriculum a graphical-based or a text-based programming tool be used or even when to combine the use of the two programming tools, and which object-oriented concepts to emphasise when using either graphical-based or text-based or again, combined programming tools.

In the respondents' completion of coursework, the Mann-Whitney test on the data indicated there was significant difference between the use of Alice and BlueJ. Student respondents found Alice easier to use in the completion of their coursework than using Java based on the mean values of the two programming tools which may be attributed to the visual output of Alice in the form of animations. This is an important point. Students value ease of completion of assessment and so the fact that they found it easier to complete an assessment using Alice suggests that they valued it as an educational tool. Also, the fact that CM1010 was an introductory module and CM1011 a follow-up module did not mean that coursework submitted in Alice was in some way more elementary than the BlueJ coursework. Indeed, the 3D animations produced in Alice were actually more

sophisticated than the corresponding Java courseworks and probably used more sophisticated programming instructions. The difference was that the students had to type the code for the CM1011 coursework whereas they could use the drag-and-drop interface for the Alice coursework.

The researcher feels a lot of work in this area is needed to confirm that indeed a graphical-based programming tool like Alice is helpful in using a text-based programming tool like BlueJ to transition successfully to Java. Thus, it is recommended that further work and data collection are needed to answer the question.

“Do graphical-based tools offer more help in understanding object-oriented concepts than text-based tools?”

The data gathered can not conclude whether graphical-based tools offer more help in understanding object-oriented concepts than text-based but the data can offer the answer that the confidence level of students were no different with the use of Alice or BlueJ. (Admittedly anecdotal) evidence suggests that the students are more confident with Alice than BlueJ. It is certainly the case that the complexity of tasks required for BlueJ was initially less than that required in the final weeks of Alice. It may be that the confidence of BlueJ is accumulated on a basis of confidence in Alice. However, the interpretation requires further investigation. The suggested questionnaire revisions may be useful in obtaining necessary data which would illustrate how the confidence of the student respondents has been increased with the use of the two programming tools.

For this study the researcher relied heavily on respondents' reflections and assessment how they have used the programming tools (i.e. Alice and BlueJ) and how their confidence were increased or improved to learn object-oriented concepts. Based on the respondents' assessments of how these two programming tools have helped them in learning object-oriented, they do not find any significant difference. The researcher feels

that a further investigation should be conducted to probe the answers to this question. Questions that would guide them to reflectively think about measuring their own capabilities maybe helpful to guide them in answering the question.

Suggestions for Further Work

Although tentative conclusions were arrived at for the research questions, the task is not complete yet. We cannot emphasise enough that improving the pedagogy of object-oriented programming is a cycle, which is continuously evolving, thus the following observations are made:

- Firstly, the ambiguities in research question number one could have been eliminated with more careful thought. The sub-questions could have been differentiated by category such as learning the user interface, adapting with screen messages, and coping with output errors. The respondents' computing background which investigated how students used the two programming environments could have been used to identify the various user levels (e.g. novice, has programming experience using Java in academic environment or professionally). A rewritten first research question could have taken the following form "1. Does the process of learning object-oriented concepts using graphical-based tools differ from using text-based tools in the following ways: 1.1 learning the various object-oriented concepts? 1.2 learning the user interface? 1.2 adapting with screen/error messages? 1.3 coping with wrong output? 1.4 completing your coursework?" Then the comparison could not only be the difference using the two programming environments in learning object-oriented concepts but also as the user types (e.g. the ratings of novice users compared to academic users compared to professional users).

- There is a need to be more specific in the language of the subsequent questionnaires so that the comparison can be made between equivalent concepts. For example, a more specific word for “easy” could be used or “easy” maybe qualified for each of the questions. It would be necessary to ensure that corresponding technical programming terms used in both questionnaires are understood by users to be the same. Some questions can be replaced by more appropriate versions e.g. terms familiarisation with the programming environments could have been better understood if it referred to on-screen or error messages for both programming environments. The data gathered from the computing background and profile of the students could have been processed to address alternative research questions (e.g. details of the student course and their ratings of how they have used both programming tools could be processed to find out whether their interest predisposed them to a graphical-based or text-based tool). The last question about respondents understanding object-oriented concepts could also make use of user levels and maybe followed with additional questions for each through a focus group or an interview.
- The processing of data could be extended to the final grades that the student respondents received from the module and the successor object-oriented modules. This data can be subjected to statistical analysis to better answer the third research question.
- An interview could have been conducted to follow-up some interesting results from the questionnaire, e.g. the length of programming exposures and issues surrounding mature students. This could start with a focus group and follow with a 1-on-1 interview for some specific points, such as wanting to know the code behind animations, and not wanting to be bothered by this code as long as the

output is what they have expected, this may indicate different programming personalities for student respondents.

- If action research were performed two possible perspectives can be derived. The researcher as the lecturer can monitor constantly the step-by-step process over varying periods of time using different instruments like case studies, focus groups, etc. This would have given the researcher additional insights over time of how the respondents used the two programming tools to help them learn object-oriented concepts by maybe conducting case studies of the two groups. Additionally, the researcher being the learner could have obtained a greater range of perspectives with the actual use of the two programming tools.
- There is a need to investigate the differences in lecturer-delivery and student-use of the concepts, which were identified as being more appropriate for graphical delivery.
- There is a need to investigate whether the results obtained from the survey are characteristics of all text-based/graphical-based development environments or whether Alice or BlueJ are unrepresentative of their respective types.
- Does having a multiplicity of learning environments confuse students more by giving them an animation, which is visually attractive but does not really convey the importance of the underlying structure of the object-oriented concepts that need to be learnt?
- How to prepare a robust self-assessment questionnaire so that learners may be given appropriate guidance on assessing their relevant strengths and weaknesses in learning object-oriented concepts?
- How to apply the self-assessment questionnaire properly or appropriately?

Results of the research may be helpful in identifying how visual learning can be used to reinforce learning object-oriented concepts with the use of a text-based tool. Thus, an important result of this research would be to identify a transition for the concepts, which need to be transferred from graphical-based learning to text-based learning. Answering these questions will make the assessment process more suited to the context of the particular classroom in which it is used and more flexible when used to investigate the process of learning and teaching object-oriented programming. Other question which could be addressed in the context of a more advanced research program is:

- How can student's reflection on programming experiences help them learn object-oriented concepts?

Findings of such research would contribute important insights into how learning, assessment and teaching object-oriented concepts can be improved.

BIBLIOGRAPHY

- Adams, J. (1996). Object-centered design: a five-phase introduction to object-oriented programming in CS1\–2, SIGCSE '96: Proceedings of the twenty-seventh SIGCSE technical symposium on Computer science education, Philadelphia, Pennsylvania, United States, pp. 78-82.
- Alphonse, C. & Ventura, P. (2002). Object orientation in CS1-CS2 by design, ITiCSE '02: Proceedings of the 7th annual conference on Innovation and technology in computer science education, Aarhus, Denmark, pp. 70-74.
- Anderson, D., Sweeney, D. & Williams, T. (1993). Statistics for Business and Economics, 5th ed. West Publishing Company.
- Armstrong, D. (2006). The quarks of object-oriented development, Commun. ACM, 49, 2, pp. 123-128.
- Bailie, F., Blank, G., Murray, K. & Rajaravivarma, R. (2003). Java visualization using BlueJ, J. Comput. Small Coll., 18, 3, pp. 175-176.
- Berghammer, R. & Huch, F. (2005). From functional to object-oriented programming: a smooth transition for beginners, FDPE '05: Proceedings of the 2005 workshop on Functional and declarative programming in education, Tallinn, Estonia, pp. 3-8.
- Bierre, K., Ventura, P., Phelps, A. & Egert, C. (2006). Motivating OOP by blowing things up: an exercise in cooperation and competition in an introductory java programming course, SIGCSE '06: Proceedings of the 37th SIGCSE technical symposium on Computer science education, Houston, Texas, USA pp. 354-358.

- Box, I. (2004). Object-oriented analysis, criterion referencing, and Bloom, ACE '04: Proceedings of the sixth conference on Australasian computing education, Dunedin, New Zealand, pp. 1-8.
- Brilliant, S. & Wiseman, T. (1996). The first programming paradigm and language dilemma, SIGCSE '96: Proceedings of the twenty-seventh SIGCSE technical symposium on Computer science education, Philadelphia, Pennsylvania, United States, pp. 338-342.
- British Psychological Society (2006). Ethical Principles for conducting Research with Human Participants, [http://www.bps.org.uk/Ethical Principles for conducting Research with Human Participants.mht](http://www.bps.org.uk/Ethical_Principles_for_conducting_Research_with_Human_Participants.mht), Date Accessed 12/07/2006.
- Burgess, G. A. & Hanshaw, C. (2006). Application of learning styles and approaches in computing sciences classes, J. Comput. Small Coll., 21, 3, pp. 60-68.
- Cable, A. (2001). Classroom-based assessment in an object-oriented programming course, J. Comput. Small Coll., 17, 1, pp. 259-264.
- Carnegie Mellon University (2006). What is Alice?, http://www.alice.org/index.php?page=what_is_alice/what_is_alice, Date Accessed 14/02/2006.
- Cohen, L. & Manion, L. (1989). Research Methods in Education, 3rd ed. London, Routledge.
- Danforth, S. & Tomlinson, C. (1988). Type theories and object-oriented programming, ACM Comput. Surv., 20, 1, pp. 29-72.
- Dann, W., Dragon, T., Cooper, S., Dietzler, K., Ryan, K., and Pausch, R. (2003). Objects: visualization of behavior and state, ITiCSE '03: Proceedings of the 8th annual conference on Innovation and technology in computer science education, Thessaloniki, Greece, pp. 84-88.

- DeClue, T. (1996). Object-orientation and the principles of learning theory: a new look at problems and benefits, SIGCSE '96: Proceedings of the twenty-seventh SIGCSE technical symposium on Computer science education, Philadelphia, Pennsylvania, United States.
- Dougherty, J. (2007). Concept visualization in CS0 using ALICE, J. Comput. Small Coll. }, 22, 3, pp. 145-152.
- Felder, R. & Silverman, L. (1988). Learning and Teaching styles in Engineering Education, <http://www4.ncsu.edu/unity/lockers/users/f/felder/public/Papers/LS-1988.pdf>, Date Accessed 28/10/2006.
- Graphpad Software (2006). How to Mann-Whitney Test, http://www.graphpad.com/help/Prism5/prism5help.html?the_results_of_a_mann_whitney_test.htm, Date Accessed 10/03/2007.
- Gordon, A. (2006). Tutorial on Alice, J. Comput. Small Coll., 21, 3, pp. 130-130.
- Gross, P. & Powers, K. (2005). Evaluating assessments of novice programming environments, ICER '05: Proceedings of the 2005 international workshop on Computing education research, Seattle, WA, USA, pp. 99-110.
- Hadjerrouit, S. (1998). Java as first programming language: a critical evaluation, SIGCSE Bull., 30, 2, pp. 43-47.
- Hadjerrouit, S. (1999). A constructivist approach to object-oriented design and programming, ITiCSE '99: Proceedings of the 4th annual SIGCSE/SIGCUE ITiCSE conference on Innovation and technology in computer science education, Cracow, Poland, pp. 171-174.
- Haaster, V. & Hagan, D. (2004). Teaching and Learning with BlueJ: an Evaluation of a Pedagogical Tool, <http://bluej.org/papers/2004-06-vanhaaster-hagan.pdf>, Date accessed 05/06/2007.

- Henriksen, P., & Kolling, M. (2004). greenfoot: combining object visualisation with interaction, OOPSLA '04: Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications, Vancouver, BC, Canada, pp. 73-82.
- Howard, R., Carver, C. & Lane, W. (1996). Felder's learning styles, Bloom's taxonomy, and the Kolb learning cycle: tying it all together in the CS2 course, SIGCSE '96: Proceedings of the twenty-seventh SIGCSE technical symposium on Computer science education, Philadelphia, Pennsylvania, United States, pp. 227-231.
- Hudak, P. (1989). Conception, evolution, and application of functional programming, ACM Computing Surveys 21, 3, pp. 359-411.
- Hughes, J. & Peiris, D. (2006). ASSISTing CS1 students to learn: learning approaches and object-oriented programming, ITICSE '06: Proceedings of the 11th annual SIGCSE conference on Innovation and technology in computer science education, Bologna, Italy, pp. 275-279.
- Irimia, A. (2001). Enhancing the introductory computer science curriculum: C++ or Java?, J. Comput. Small Coll., 17, 2, pp. 159-166.
- Kelleher, C., Pausch, R. & Kiesler, S. (2007). Storytelling alice motivates middle school girls to learn computer programming, CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems, San Jose, California, USA, pp. 1455-1464.
- Kempf, R. & Stelzner, M. (1987). Teaching object-oriented programming with the KEE system, OOPSLA '87: Conference proceedings on Object-oriented programming systems, languages and applications, Orlando, Florida, United States, pp. 11-25.
- Kolling, M. & Rosenberg, J. (1996). An object-oriented program development environment for the first programming course, SIGCSE '96: Proceedings of the

- twenty-seventh SIGCSE technical symposium on Computer science education, Philadelphia, Pennsylvania, United States, pp. 190-194.
- Kolling, M. & Rosenberg, J. (2000). Objects first with Java and BlueJ (seminar session), SIGCSE '00: Proceedings of the thirty-first SIGCSE technical symposium on Computer science education, Austin, Texas, United States, pp. 429.
- Kolling, M. & Rosenberg, J. (2001). Guidelines for teaching object orientation with Java, ITiCSE '01: Proceedings of the 6th annual conference on Innovation and technology in computer science education, Canterbury, United Kingdom, pp. 33-36.
- Kouznetsova, S. (2007). Using BlueJ and Blackjack to teach object-oriented design concepts in CS1, *J. Comput. Small Coll.*, 22, 4, pp. 49-55.
- Leavens, G. (1991). Introduction to the literature on object-oriented design, programming, and languages, *SIGPLAN OOPS Mess.*, 2, 4, pp. 40-53.
- Lewis, J. (2000). Myths about object-orientation and its pedagogy, SIGCSE '00: Proceedings of the thirty-first SIGCSE technical symposium on Computer science education, Austin, Texas, United States, pp. 245-249.
- Lister, R. & Leaney, J. (2003). First year programming: let all the flowers bloom, ACE '03: Proceedings of the fifth Australasian conference on Computing education, Adelaide, Australia, pp. 221-230.
- Lister, R., Berglund, A., Clear, T., Bergin, J., Garvin-Doxas, K., Hanks, B., Hitchner, L., Luxton-Reilly, A., Sanders, K., Schulte, C., & Whalley, J. (2006). Research perspectives on the objects-early debate, ITiCSE-WGR '06: Working group reports on ITiCSE on Innovation and technology in computer science education, Bologna, Italy, pp. 146-165.

- Luker, P. (1994). There's more to OOP than syntax!, SIGCSE '94: Proceedings of the twenty-fifth SIGCSE symposium on Computer science education, Phoenix, Arizona, United States, pp. 56-60.
- Madden, M. & Chambers, D. (2002). Evaluation of student attitudes to learning the Java language, PPPJ '02/IRE '02: Proceedings of the inaugural conference on the Principles and Practice of programming, 2002 and Proceedings of the second workshop on Intermediate representation engineering for virtual machines, 2002, Dublin, Ireland, pp. 125-130.
- Mazaitis, D. (1993). The object-oriented paradigm in the undergraduate curriculum: a survey of implementations and issues, 25, 3, pp. 58-64.
- McDermott, R., Brindley, G. & Eccleston, G. (2007). Software Design & Development using Alice, Robert Gordon University.
- Mitchell, W. (2000). A paradigm shift to OOP has occurred...implementation to follow, CCSC '00: Proceedings of the fourteenth annual consortium on Small Colleges Southeastern conference, Roanoke College, Salem, Virginia, United States, pp. 94-105.
- Mougin, P. & Ducasse, S. (2003). OOPAL: integrating array programming in object-oriented programming, OOPSLA '03: Proceedings of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, Anaheim, California, USA, pp. 65-77.
- Neubauer, B. & Strong, D. (2002). The object-oriented paradigm: more natural or less familiar?, J. Comput. Small Coll., 18, 1, pp. 280-289.
- Olan, M. (2004). Dr. J vs. the bird: Java IDE's one-on-one, J. Comput. Small Coll., 19, 5, pp. 44-52.

- Or-Bach, R. & Lavy, I. (2003). Students' understanding of object orientation, ITiCSE '03: Proceedings of the 8th annual conference on Innovation and technology in computer science education, Thessaloniki, Greece, pp. 251-251.
- Or-Bach, R. & Lavy, I. (2004). Cognitive activities of abstraction in object orientation: an empirical study, SIGCSE Bull., 36, 2, pp. 82-86.
- Osborne, M. (1992). The role of object-oriented technology in the undergraduate computer science curriculum, OOPSLA '92: Addendum to the proceedings on Object-oriented programming systems, languages, and applications (Addendum), Vancouver, British Columbia, Canada, pp. 303-308.
- Patterson, J. & Haddow, J. (2003). Teaching Java: using an object-oriented database and the BlueJ IDE, ITiCSE '03: Proceedings of the 8th annual conference on Innovation and technology in computer science education, Thessaloniki, Greece, pp. 273-273.
- Pokkunuri, B. P. (1989). Object Oriented Programming, SIGPLAN Not., 24, 11, pp. 96-101.
- Powers, K., Ecott, S. & Hirshfield, L. (2007). Through the looking glass: teaching CS0 with Alice, SIGCSE '07: Proceedings of the 38th SIGCSE technical symposium on Computer science education, Covington, Kentucky, USA, pp. 213-217.
- Pratt, T. & Zelkowitz, M. (1996). Programming Languages: Design and Implementation, 3rd ed. Englewood Cliffs, N.J.: Prentice Hall.
- Pugh, J., LaLonde, W. & Thomas, D. (1987). Introducing object-oriented programming into the computer science curriculum, SIGCSE '87: Proceedings of the eighteenth SIGCSE technical symposium on Computer science education, St. Louis, Missouri, United States, pp. 98-102.

- Rankin, Y., Lechner, T. & Gooch, B. (2007). Team-based pedagogy for CS102 using game design, SIGGRAPH '07: ACM SIGGRAPH 2007 educators program, San Diego, California, pp. 20.
- Rasala, R., Raab, J. & Proulx, V. (2001). Java power tools: model software for teaching object-oriented design, SIGCSE '01: Proceedings of the thirty-second SIGCSE technical symposium on Computer Science Education, Charlotte, North Carolina, United States, pp. 297-301.
- Rentsch, T. (1982). Object oriented programming, Sigplan Not., 17, 9, pp. 51-57.
- Robert Gordon University (2006). Object-Oriented Programming Techniques, http://www.rgu.ac.uk/prospectus/modules/disp_moduleView.cfm?Descriptor=CM1011, Date Accessed 20/10/2006.
- Robert Gordon University (2006). The Research Ethics Policy, <http://www.rgu.ac.uk/research/activity/page.cfm?pge=11627>, Date Accessed 09/07/2006.
- Roumani, H. (2006). Practice what you preach: full separation of concerns in CS1/CS2, SIGCSE '06: Proceedings of the 37th SIGCSE technical symposium on Computer science education, Houston, Texas, USA, pp. 491-494.
- Rowntree, D. (2006). A new way with words in distance education, http://www-iet.open.ac.uk/pp/D.G.F.Rowntree/words_in_de.htm, Date accessed 25/02/2007.
- Sanders, D. & Heeler, P. (2001). Introduction to BlueJ: a Java development environment for CS1 and CS2, Proceedings of the seventh annual consortium for computing in small colleges central plains conference on The journal of computing in small colleges, Branson, Missouri, United States, pp. 115-116.
- Sanders, D., Heeler, P. & Spradling, C. (2001). Introduction to BlueJ: A Java Development Environment, Consortium for Computing in Small Colleges, 16, 3, pp. 257-258.

- Schahczenski, C. (2000). Object-oriented databases in our curricula, Proceedings of the seventh annual CCSC Midwestern conference on Small colleges, Valparaiso Univ., Valparaiso, Indiana, United States, pp. 11170-176.
- Sebesta, R. (1996). Concepts of Programming Languages, 3rd ed. Reading, Mass.: Addison-Wesley Publishing Company.
- Seffah, A., Bari, M. & Desmarais, M. (1999). Assessing object-oriented technology skills using an Internet-based system, ITiCSE '99: Proceedings of the 4th annual SIGCSE/SIGCUE ITiCSE conference on Innovation and technology in computer science education, Cracow, Poland, pp. 71-74.
- Snyder, A. (1986). Encapsulation and inheritance in object-oriented programming languages, OOPLSA '86: Conference proceedings on Object-oriented programming systems, languages and applications, Portland, Oregon, United States, pp. 38-45.
- Special Interest Group on Computer Science Education (2005). Java Task Force Report, <http://www.sigcse.org/javataskforce.htm>, Date Accessed 12/02/2006.
- Stiller, E. & LeBlanc, C. (2003). Creating new computer science curricula for the new millennium, J. Comput. Small Coll., 18, 5, pp. 198-209.
- Tewari, R. & Gitlin, D. (1994). On object-oriented libraries in the undergraduate curriculum: importance and effectiveness, SIGCSE '94: Proceedings of the twenty-fifth SIGCSE symposium on Computer science education, Phoenix, Arizona, United States, pp. 319-323.
- Thomasson, B., Ratcliffe, M. & Thomas, L. (2006). Identifying novice difficulties in object oriented design, ITICSE '06: Proceedings of the 11th annual SIGCSE conference on Innovation and technology in computer science education, Bologna, Italy, pp. 28-32.

- Vilner, T., Zur, E. & Gal-Ezer, J. (2007). Fundamental concepts of CS1: procedural vs. object oriented paradigm - a case study, ITiCSE '07: Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education, Dundee, Scotland, pp. 171-175.
- Wegner, P. (1990). Concepts and paradigms of object-oriented programming, SIGPLAN OOPS Mess., 1, 1, pp. 7-87.
- Wei, F., Moritz, S., Parvez, S. & Blank, G. (2005). A student model for object-oriented design and programming, J. Comput. Small Coll., 20, 5, pp. 260-273.
- Wong, W. (2006). Graphical-And Text-Based Programming: Complementary, Not Competitive, Electronic Design, http://electronicdesign.com/Files/29/13241/13241_01.pdf, Date accessed 12/12/2006.
- Zhu, H. & Zhou, M. (2003). Methodology first and language second: a way to teach object-oriented programming, OOPSLA '03: Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, Anaheim, CA, USA, pp. 140-147.

APPENDIX A

QUESTIONNAIRE for Alice Users (Respondent's Self-Assessment of Alice)

I. INFORMATION TECHNOLOGY USAGE

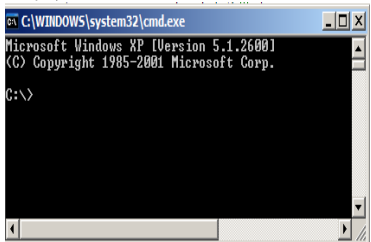
How long have you been programming?

- none at all
 less than 1 year
 more than 1 year

What was the first programming language you used? _____

What computer games do you play? _____

PLEASE CHECK THE COLUMN WHICH IS MOST RELEVANT TO YOUR ANSWER.

Questions	Everyday	At least once a week	At least once per fortnight	At least once per month	Less often	Never
How often do you use a computer?						
How often you use the command prompt (e.g. c:\>)? 						
How often do you use the following Internet services?						
World Wide Web (WWW)						
Electronic Mail (E-mail)						
Usenet (Newsgroups)						
Instant Messaging (IM)						
File Transfer Protocol (FTP) – Upload/Download Files						

II. ALL ABOUT ALICE

Have you heard of Alice before using it in the laboratory?

- Yes
 No

If yes, where did you hear it from?

How long have you used Alice?

- less than 3 months
 3 – 6 months
 7 – 9 months
 10 – 12 months
 more than 1 year

How long did it take you to get familiarised with Alice's user interface?

- less than 1 week
 1 - 2 weeks
 3 – 4 weeks
 more than 1 month

How familiar are you with the following terms?

Terms	Very Familiar (1)	(2)	(3)	(4)	Not at all (5)
Interactive					
3D Objects					
Visualisation					
Animation					
Drag & Drop					
Debugging					
World					
Create New Event					
Create New Variable					
Camera					

III. WORKING WITH ALICE

PLEASE RATE THE FOLLOWING STATEMENTS.

Statements	Easy (1)	(2)	(3)	(4)	Difficult (5)	Not Applicable
How easy is it to use Alice in completing your courseworks?						
How easy is it to use Alice in understanding the following object-oriented concepts?						
Class						
Object						
Method						
Message Passing						
Inheritance						
Encapsulation						
Polymorphism						
How easy is it to recover from the following in Alice?						
System error messages						
Output (animation) is not the movement you expected						
Unable to drag and drop						

Statements	Very much (1)	(2)	(3)	(4)	Not at all (5)	Not Applicable
How much has Alice increased your confidence in learning the following object oriented concepts?						
Class						
Object						
Method						
Message Passing						
Inheritance						
Encapsulation						
Polymorphism						

IV. OTHER QUESTIONS.

PLEASE COMPLETE THE FOLLOWING SENTENCES.

My most difficult experience using Alice was when ...

1. _____

2. _____

3. _____

My most exciting experience using Alice was when...

1. _____

2. _____

3. _____

My most rewarding experience using Alice was when...

1. _____

2. _____

3. _____

What have you learnt from using Alice?

1. _____

2. _____

3. _____

What features are missing from Alice?

1. _____

2. _____

3. _____

V. RESPONDENT'S INFORMATION

PLEASE CHECK ONE WHICH APPLIES TO YOU.

To which age category do you belong to?

- less than 18
- 18 - 21
- 22 - 25
- 26 - 30
- 31 - 35
- 35 +

Gender:

- Male
- Female

What is the highest level of Mathematics attained?

- Standard Grade _____
- Higher Grade _____
- Others (please specify) _____

Which course are you enrolled in?

- Internet & Multimedia
- Graphics & Animation
- Business & e-Commerce
- Computer Science
- Computing & Information
- Computing for Intelligent System

Are you willing to be contacted at a later date for a possible interview?

- Yes
- No

If yes, please indicate your

Email-address: _____

THANK YOU VERY MUCH!

APPENDIX B

QUESTIONNAIRE for Java Users (Respondent's Self-Assessment of Java)

I. INFORMATION TECHNOLOGY USAGE

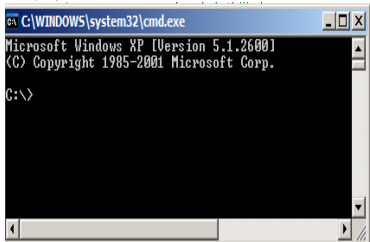
How long have you been programming?

- none at all
 less than 1 year
 more than 1 year

What was the first programming language you used? _____

What computer games do you play? _____

PLEASE CHECK THE COLUMN WHICH IS MOST RELEVANT TO YOUR ANSWER.

Questions	Everyday	At least once a week	At least once per fortnight	At least once per month	Less often	Never
How often do use a computer?						
How often you use the command prompt (e.g. c:\>)?						
						
How often do you use the following Internet services?						
World Wide Web (WWW)						
Electronic Mail (E-mail)						
Usenet (Newsgroups)						
Instant Messaging (IM)						
File Transfer Protocol (FTP) – Upload/Download Files						

II. ALL ABOUT JAVA

Have you heard of Java before using it in the classroom?

- Yes
 No

If yes, where did you hear it from?

How long have you used Java?

- less than 3 months
 3 – 6 months
 7 – 9 months
 10 – 12 months
 more than 1 year

Which editor are you using for Java? _____

Please write down other editors you know for Java. _____

How long did it take you to get familiarised with your Java's user interface?

- less than 1 week
 1 - 2 weeks
 3 – 4 weeks
 more than 1 month

How familiar are you with the following terms?

Terms	Very Familiar (1)	(2)	(3)	(4)	Not at all (5)
Command Prompt					
JVM					
Compiler					
Byte Code					
Run					
Projects					
IDE					
Java Libraries					
Java Editors					
Debugger					

III. WORKING WITH JAVA

PLEASE RATE THE FOLLOWING STATEMENTS.

Statements	Easy (1)	(2)	(3)	(4)	Difficult (5)	Not Applicable
How easy is it to use Java in completing your courseworks?						
How easy is it to use Java in understanding the following object oriented concepts?						
Class						
Object						
Method						
Message Passing						
Inheritance						
Encapsulation						
Polymorphism						
How easy is it to recover from the following in Java?						
System error messages						
Syntax errors						
Output (result) is not What you expected						

Statements	Very Much (1)	(2)	(3)	(4)	Not at All (5)	Not Applicable
How much has Java increased your confidence in learning the following object oriented concepts?						
Class						
Object						
Method						
Message Passing						
Inheritance						
Encapsulation						
Polymorphism						

IV. OTHER QUESTIONS.
PLEASE COMPLETE THE FOLLOWING SENTENCES.

My most difficult experience using Java was when ...

1. _____

2. _____

3. _____

My most exciting experience using Java was when...

1. _____

2. _____

3. _____

My most rewarding experience using Java was when...

1. _____

2. _____

3. _____

What have you learnt from using Java?

1. _____

2. _____

3. _____

What features are missing from Java?

1. _____

2. _____

3. _____

V. RESPONDENT'S INFORMATION

PLEASE CHECK ONE WHICH APPLIES TO YOU.

To which age category do you belong to?

- less than 18
- 18 - 21
- 22 - 25
- 26 - 30
- 31 - 35
- 35 +

Gender:

- Male
- Female

What is the highest level of Mathematics attained?

- Standard Grade _____
- Higher Grade _____
- Others (please specify) _____

Which course are you enrolled in?

- Internet & Multimedia
- Graphics & Animation
- Business & e-Commerce
- Computer Science
- Computing & Information
- Computing for Intelligent System

Are you willing to be contacted at a later date for a possible interview?

- Yes
- No

If yes, please indicate your

Email-address: _____

THANK YOU VERY MUCH!