

Spring 2009

Development of the Curriculum for the Introduction to Computer Science Course

Nancy Major
Regis University

Follow this and additional works at: <https://epublications.regis.edu/theses>



Part of the [Computer Sciences Commons](#)

Recommended Citation

Major, Nancy, "Development of the Curriculum for the Introduction to Computer Science Course" (2009). *All Regis University Theses*.
98.

<https://epublications.regis.edu/theses/98>

Regis University
College for Professional Studies Graduate Programs
Final Project/Thesis

Disclaimer

Use of the materials available in the Regis University Thesis Collection ("Collection") is limited and restricted to those users who agree to comply with the following terms of use. Regis University reserves the right to deny access to the Collection to any person who violates these terms of use or who seeks to or does alter, avoid or supersede the functional conditions, restrictions and limitations of the Collection.

The site may be used only for lawful purposes. The user is solely responsible for knowing and adhering to any and all applicable laws, rules, and regulations relating or pertaining to use of the Collection.

All content in this Collection is owned by and subject to the exclusive control of Regis University and the authors of the materials. It is available only for research purposes and may not be used in violation of copyright laws or for unlawful purposes. The materials may not be downloaded in whole or in part without permission of the copyright holder or as otherwise authorized in the "fair use" standards of the U.S. copyright laws and regulations.

Development of the Curriculum for the Introduction to Computer Science Course

by

Nancy Major
nmmajor24@msn.com

A Thesis/Practicum Report submitted in partial fulfillment of the requirements for the
degree of Master of Science in Computer Information Technology

School of Computer and Information Sciences
Regis University
Denver, Colorado

March 5, 2009

Abstract

This project proposes to reformat the curriculum for an Introduction to Computer Science course for high school students, currently taught as a one semester course. Several issues with the current course are addressed with recommendations for changes intended for the benefit of students at their school.

In the past five years, enrollment in the school's Advanced Placement (AP) Computer Science course has decreased from 50 students to 25 students despite no significant change in overall enrollment or student demographics. For the portion of those students enrolled in the course who have taken the Advanced Placement exam during the past four years, the passing rate was 50% to 100%. When students were encouraged to take the AP Computer Science A exam, a less rigorous exam, the passing rate increased.

The school has been known to develop a curriculum that best meets the needs of its students. The current Introduction to Computer Science course is not meeting their needs. This project addresses several key aspects of the course that could be changed to better prepare the students for the Advanced Placement (AP) Computer Science course and increase enrollment in both the introductory and advanced courses, particularly (this would be an added bonus) with respect to female students.

The key aspects with the Introduction to Computer Science course that this paper will address are as follows:

1. The effectiveness of the curriculum as an introduction to the Advanced Placement (AP) Computer Science, following the curriculum as outlined by the College Board.

2. The course objectives such as the educational philosophy of the course, how the students will be introduced to object-oriented programming using java, the programming language used in the AP Computer Science course, and choosing the software, textbook and supplemental materials that would best meet the needs of the students and support the course objectives.

3. The classroom teaching methodology. This would include, but not be limited to, the expectations of the students both in the classroom and as it relates to homework beyond class times, the nature of homework assignments, when and how much would be assigned on a daily basis, the types of assessments that would determine the students grades, and how these assessment would be graded.

4. Building student interest in the computer science field and demonstrating that every student is capable of basic programming skills.

Acknowledgements

I would like to thank my husband, Mark, who continues to support me in whatever I choose to do and inspires me to be the best I can be. I would also like to thank my beautiful four daughters, Sara, Emily, Lizzie and Rosie, who have, and will, accomplish more than I can dream about. I would also like to thank my students who continue to be an inspiration to me every day as they work through their studies and never give up.

Table of Contents

Abstract	ii
Executive Summary	1
Chapter 1 – Introduction	3
Problem Statement	3
Project Proposal	4
Scope of Project	4
Feasibility	5
Chapter 2 – Review of Literature and Research	8
Description of Existing Advanced Placement (AP) Curriculum	11
<i>AP Computer Science AB</i>	11
<i>AP Computer Science A</i>	13
Description of Existing Introduction to Computer Science Course	14
Research Resources	15
Chapter 3 – Methodology	16
Phase I: Analysis	16
Phase II: Design	16
Phase III: Implementation	17
Phase IV: Testing	17
Phase V: Maintenance	17
Timeline	18
Formats for Presenting Results and Deliverables	18
Chapter 4 – Project Analysis and Results	19
Functional Requirements	19
Goals	20
Course Objectives and Teaching Methodologies	20
Java Environment and Textbook	31
<i>IDE (Integrated Development Environment)</i>	32
<i>Software</i>	37
<i>Textbooks</i>	38
Lesson Plans and Daily Assignments	40
<i>Lesson 1</i>	42
<i>Lesson 2</i>	42
<i>Lesson 3</i>	53
<i>Lesson 4</i>	53
<i>Lesson 5</i>	54
<i>Lesson 6</i>	56
<i>Lesson 7</i>	57
<i>Lesson 8</i>	59
<i>Lesson 9</i>	59
<i>Lesson 10</i>	60
<i>Lesson 11</i>	61
<i>Lesson 12</i>	62
<i>Lesson 13</i>	63
<i>Lesson 14</i>	66

Assessments	68
Grading Rubrics.....	71
Motivation and Building Confidence	71
Chapter 5 – Project History	76
Goal Evaluation.....	76
Chapter 6 – Lessons Learned and Next Evolution of the Project	77
What I Learned From the Project Experience.....	77
What I Would Have Done Differently	77
Did the Project Meet Initial Expectations?	78
What Would Be the Next Stage Of Evolution for the Project If Continued? .	79
Recommendations	80
Summary.....	82
References.....	83
Appendix A	86
Appendix B	89
<i>Student and Parent Signature Confirming Understanding of Course Objectives</i>	89
Appendix C.....	90
<i>Downloading the SUN Java JDK (Java Development Kit) and Java API</i> <i>Documentation</i>	90
Appendix D.....	91
<i>Downloading JCreator, the IDE (Integrated Development Environment)</i>	91
Appendix E	92
<i>Lesson 1- Course Objectives and Classroom Expectations PowerPoint</i>	92
Appendix F	98
<i>Login Instructions for Blackboard Web Site</i>	98
Appendix G.....	99
<i>Lesson Plan for Day 4</i>	99
Appendix H.....	103
<i>Currents in Technology" Guidelines</i>	103
Appendix I	104
<i>Lesson Plan for Day 5</i>	104
Appendix J.....	107
<i>Lesson Plan for Day 6</i>	107
Appendix K.....	114
<i>Lesson Plan for Day 7</i>	114
Appendix L	121
<i>Lesson Plan for Day 10</i>	121
Appendix M.....	124
<i>Lesson Plan for Day 11</i>	124
Appendix N.....	127
<i>Lesson Plan for Day 12</i>	127
Appendix O.....	133
<i>Quiz 1, Unit 1</i>	133
Appendix P	134
<i>Quiz 1, Unit 1 – Answer Key</i>	134

Appendix Q	135
<i>Unit 1 Test, Unit 1</i>	135
Appendix R	139
<i>Test, Unit 1 – Answer Key</i>	139

List of Figures

Figure 1: Download Page for Java Development Kit	43
Figure 2: JDK 6 Update 12 Download Page.....	43
Figure 3: Select Correct Java SE Download for Machine	44
Figure 4: Download Selected SE Development Kit	44
Figure 5: Java JDK Path File	45
Figure 6: Download Java SE 6 Documentation.....	45
Figure 7: Download Page for JCreator	46
Figure 8: JCreator Setup Wizard.....	47
Figure 9: JCreator Project/Project Settings.....	48
Figure 10: JCreator Required Libraries tab	48
Figure 11: JCreator name and add jar file folder	49
Figure 12: JCreator Add Archive selection	49
Figure 13: JCreator jar files list	50
Figure 14: JCreator making set of jars available	50
Figure 15: "All About Me" Prompt	51
Figure 16: Lesson 5 program, Hello, World!.....	55
Figure 17: Lesson 5 assignment "The Great Oz has spoken!"	55
Figure 18: Lesson 5 Program DrawSquare and output.....	57
Figure 19: Assignment DrawHouse code	58
Figure 20: Assignment DrawHouse output.....	59
Figure 21: SmileyFace Assignment Code & Output	60
Figure 22: Lesson 10 Benzene Program	61
Figure 23: Lesson 12 Karel J. Robot Program Draw H.....	63
Figure 24: Lesson 13 Karel J. Robot Retrieve Newspaper.....	64
Figure 25: Lesson 13 Karel J. Robot Pick Up Groceries.....	65
Figure 26: Karel J. Robot Repetitive Code.....	65
Figure 27: Lesson 14 Karel J. Robot Mountain Assignment.....	66
Figure 28: Lesson 14 Karel J. Robot Figure8 Assignment.....	67

List of Tables

Table 1: Introduction to Computer Science Course Daily Content and Assignment Schedule, Week 1.....	41
Table 2: Introduction to Computer Science Course Daily Content and Assignment Schedule, Week 2.....	56
Table 3: Introduction to Computer Science Course Daily Content and Assignment Schedule, Week 3.....	62

Executive Summary

This paper proposes to develop and design a new curriculum for the Introduction to Computer Science course at the instructor's school that will use the java programming language. The goal of the project is to write the course objectives, the first few assignments for the course and how those topics would be taught, assessments and rubrics for those topics and how they would be graded, and address some ways to instill interest, motivation and confidence in the students in learning the subject matter.

The scope of the project will be the development of the first unit of curriculum. Even though the curriculum itself is the most important piece of this course, the underlying problem occurring at the instructor's school is developing an interest in the field itself. This project will focus on ways to develop and maintain this interest in the students. Given this focus and scope, the project will only address the following key issues to implement this course: course objectives, teaching methodologies, programming environment and textbooks, assignments and assessments, grading rubrics, and a learning environment structured to promote confidence and interest in computer programming.

Since this introductory course is not critical to completing the Advanced Placement (AP) Computer Science course, it can be viewed as an enhancement to the curriculum and any material covered would further assist students in their success in future courses. This gives greater flexibility in adjusting the course on an as needed basis. If successful, the project can continue along the same path. If not successful, the project can be reassessed and changed to adjust to the needs of the students. The

uniqueness of the introductory course allows the instructor to cover only the material that time will allow. It is important that the instructor's methodology be flexible and grounded in the understanding that the initial approach taken may not be the way this course ultimately ends up being taught. It may change during the semester or from semester to semester.

Because the Advanced Placement Board sets the AP curriculum, a standard curriculum guide is already in place for the Advanced Placement Computer Science course. The College Board does, however, allow the instructor flexibility with regard to teaching methods. The course is taught using java as the programming language. To ensure that the students in the introductory course are prepared for Advanced Placement work, java instruction will be required in this course as well.

This project proposes the continued use of student workbooks as a curriculum guide as teachers use in other mathematics courses and current computer science courses. These workbooks will model the current department workbooks in that they will contain assignments for students as well as be a resource containing notes for the courses.

The instructor is confident that this proposal will produce a curriculum that will meet the school's high standards. Through the curriculum development process, the goals will be to increase the students' level of preparation for the Advanced Placement Computer Science course and to increase the level of interest in computer science courses among the student population.

Chapter 1 – Introduction

Children are our future. This is seen as particularly true by many high school teachers as they meet young people in their classrooms. A common objective among many teachers in educating these young minds is to give them the opportunity to participate in courses from a wide range of academic areas including computer science. The Advanced Placement (AP) Computer Science course is one of these courses. This course provides students the opportunity to enroll in a computer science course for college credit. To better prepare students for the Advanced Placement course, the instructor's school offers the Introduction to Computer Science and Intermediate Computer Science courses, each being one semester in length.

1.1 – Problem Statement

The number of students registering for Advanced Placement courses at the instructor's school has steadily increased in the last five years. For example, five years ago, 254 students enrolled in Advanced Placement Calculus. Currently, there are 300 students enrolled in Calculus. However, the number of students enrolled in Advanced Placement Computer Science has decreased. The school offers two sections of Introduction to Computer Science each year with approximately 20-25 students in each section. Approximately one-half of these students enroll in the Advanced Placement Computer Science course. Those that do not register for the AP course cite reasons such as the material is too difficult, they do not feel adequately prepared, or they found it is a subject in which they do not have an interest.

From a class size of 25 of those students who do enroll in the Advanced Placement Computer Science, only seven to 11 students take the AP Computer Science exam offered by the College Board in May. This project examines why this is happening and recommends changes in the course to promote interest and confidence in computer science within the school's student population.

1.2 – Project Proposal

This project proposes to develop a new approach to the teaching of the Introduction to Computer Science course that will use the java programming language. A goal of this course is to instill an interest in computer science and expose students to the possibility of computer science as a career or, at the very least, prepare the students for the possibility that their chosen academic area in college may require a computer science course for which they would be more prepared.

One overarching goal for the project is to evaluate and present ways to teach the students object-oriented programming, to instill interest, and motivate students to continue studying computer science, and to build student confidence in their computer science abilities. A second broad goal is to write the course objectives as well as sufficient assignments, assessments, and rubrics to provide a solid foundation for the restructuring of this course.

1.3 – Scope of Project

The scope of the project will be the first unit of the curriculum. The main topic of the first unit will be the introduction to object-oriented programming. This introduction may include understanding objects, classes and methods of object-oriented programming and writing a simple java program. These topics will be

covered at an introductory level and not at the depth that they would be covered in the Intermediate Computer Science or Advanced Placement course.

Even though the curriculum itself is the most important aspect of this course, the underlying problem occurring at the instructor's school is developing interest in the computer science field itself. This project will focus on exploring ways to develop and retain such interest in the students. Consequently, the project will only be addressing key issues as they pertain to the beginning of this course (i.e., in the first unit). These issues are course objectives, teaching methodologies, programming environment and textbooks, assignments and assessments, grading rubrics, and structuring a learning environment to promote confidence and interest in computer programming. Under the rationale proposed here, at the end of the first unit, these items would be reassessed and the next unit's methods would be subject to changes that could better enhance the course, including the learning environment and the level of student interest.

1.4 – Feasibility

The Advanced Placement College Board uses java as the programming language to teach object-oriented programming. Java was chosen for many reasons one of which was to force teachers in the Advanced Placement curriculum to use an object-oriented approach. The Advanced Placement Board establishes the Advanced Placement curriculum so a standard curriculum guide is already in place for the Advanced Placement Computer Science course. The drop in enrollment in the Advanced Placement Computer Science course, particularly at a time when technology has become such a critical industry, demands investigation. It may be that

student needs have changed, or there may be other contributing factors to be discovered. A study is necessary to explore this decline in enrollment. The result of the study will help lay the foundation for an alternative approach to the introductory course to build interest from the students in computer science, thus increasing not only the enrollment in the Advanced Placement Computer Science course but the success rate on the Advanced Placement exam for these students as well. Therefore, the operational feasibility of this project is justified.

Furthermore, since these courses have always had student workbooks as a curriculum guide (as with many other mathematics courses at the instructor's school), the Introductory Computer Science course will continue this practice as well. These workbooks have been successfully employed in the curriculum in several of the other courses in the mathematics department. The workbooks would be built after the completion of revisions in the course through one semester. Workbooks are put together through the district's printing services and sold to students through the bookstore at a nominal fee to the student. The cost of the workbook is covered by the mathematics department for those students who cannot afford the fee. Economically, the feasibility of this project has been justified.

Finally, the mathematics department at the school has determined that the operational and economic feasibility of this project is justified based upon the school's historical record of success with Advanced Placement exams in general, and particularly with the Advanced Placement Computer Science exam, which the school wishes to continue. Seven years ago, the school had 17 students that took the Advanced Placement Computer Science AB exam, the more advanced exam of the

two Advanced Placement Computer Science exams offered by the College Board. Seven of those scored a five, the highest possible score, on the exam and eighty-two percent of those students passed with a score of three or above. Four students took the Advanced Placement Computer Science A exam, a less rigorous exam. These students all passed with a score of three or above but no one scored a five.

Six years ago the school had 22 students take the AB exam and 3 students take the A exam. On the AB exam, seven students scored a five and ninety-one percent of those students passed the exam. On the A exam, one student scored a five and all of them passed the exam.

Last May, only four students took the AB exam and three students took the A exam. Two of those students scored a five on the AB exam, no one scored a five on the A exam. However, all seven students passed their exam with a score of three or above. The project is also justified by the growing concern of the administration and faculty to best meet the needs of the students and to best prepare the students for college-level work.

Chapter 2 – Review of Literature and Research

A tremendous amount of support is available to assist teachers motivated to increase the effectiveness of computer science courses. The seminal issue will be discerning what best fits the needs of the students currently sitting in the classroom. For the Advanced Placement Computer Science Course, a wealth of information is available on the College Board Website that is named AP Central, <http://apcentral.collegeboard.com/apc/Controller.jsp>. At AP Central, an instructor can find the course requirements and support in the form of documents written by College Board members and instructors across the nation and in the form of discussion boards. This Web site encourages educators to use any methodologies, ideas and activities that are posted there. The atmosphere of the College Board community is to share ideas to help other instructors to better help their students. Trees (2009) gives the following scenario of what makes a great teacher.

An interview with a “smart” computer:

Question: What is a “teacher”?

Answer: One whose occupation is to instruct (Merriam-Webster dictionary).

Question: What is an “instructor”?

Answer: One that instructs (Merriam-Webster dictionary)

Question: What is a “professor”?

Answer: One that teaches or professes special knowledge of an art, sport, or occupation requiring skill (Merriam-Webster dictionary)

Let's try again. An interview with AP CS [computer science] teachers:

Question: What is a “teacher”?

Answer: A teacher is one who **shares** knowledge.

Question: What makes a “good” teacher “great”?

Answers: A sincere interest in students

Organization

Experience

Ability to explain concepts MANY ways

Patience

...

...

A bag of tricks.

Question: What do you mean “bag of tricks”?

Answer: You know... teaching techniques, activities, and more.

Question: Where do you get these “tricks” to put in your bag?

Answer: Well, why from other **teachers**, of course!

Articles posted on the AP Central web site are there to help fellow educators.

Authors are educators themselves who have found something that works for them and their students. A teacher becomes “great” when they research these ideas and find the ones that best fit the teacher and his or her students’ needs. These articles

are written by professionals who have tried many things in their classrooms, some of which have been successful and some of which have not. By listening to fellow educators, teachers can discern what may work for them. This strategy will help minimize the likelihood that the initial group of students to whom a teacher first presents a new course will wind up as guinea pigs subjected to the many mistakes teachers make along the way with new material. Students deserve to have their instructors well prepared. This project intends to make a computer science instructor's first experience teaching a computer science course a positive experience, with the instructor having confidence in being well prepared and having the ability to truly help his or her students being the best they can be.

Support for educators for teaching methodologies is also available through various books. Literature includes, but is not limited to, best practices to ensure high school students are prepared for college-level work. Studies have been done from many venues to answer the questions on how best to prepare students for college. Conley (2005) has written a book about his studies of successful college students and what it took to get them prepared for their college education. Barth et al. (2005) have written about professional learning communities and how they can assist students to become successful learners.

Although the course that is being discussed in this document is not the Advanced Placement Computer Science Course, the introductory course needs to address the issue of best preparing students to enter the Intermediate Computer Science course and subsequently the Advanced Placement course.

2.1 – Description of Existing Advanced Placement (AP) Curriculum

Because the Advanced Placement Board sets the AP curriculum, a standard curriculum guide is already in place for the Advanced Placement Computer Science course. The College Board does, however, allow the instructor flexibility with regard to teaching methods. The course is taught using java as the programming language. One reason why java was chosen was to force teachers in the Advanced Placement curriculum to use an object-oriented approach. To ensure that the students in the introductory course are prepared for Advanced Placement work, java instruction will be required in the proposed introductory course as well.

The current Advanced Placement Computer Science course can be taught with different curriculums. One is labeled “AB” and the other “A”. Both AB and A are described in the following sections; generally, A is taught with fewer topics and less rigor than AB. Both of these curriculums require the use of java as the programming language.

2.1.1 – AP Computer Science AB

The AB course covers material taught in most college courses spanning a year’s time. Scoring a 4 or a 5 on the Advanced Placement Computer Science AB exam often qualifies college-bound students with the potential of earning a full first-year college computer science credit. The AB course covers a wide-range of topics, and all these topics are covered at some depth. These topics, as described by the College Board in the course description (AP Central, 2009, p. 7) are object-oriented program design, program implementation, program analysis, standard data structures,

standard algorithms, and computing in context, each of which is briefly discussed, below.

Object-oriented program design includes program and class design. Within this topic, the student is taught about abstract data types, inheritance, polymorphism, class instances and class attributes, class hierarchy, Java Library Classes, constructors and methods, interface, and encapsulation.

Program implementation involves teaching the students basic fundamental programming constructs of the language, teaching about information hiding, procedural abstraction, declarations of classes, variables and methods, interface declarations, recursive data structures, dynamically allocated structures, and control constructs. Input and output varies by what approach the instructor deems appropriate for his or her course. No particular approach is dictated by the Advanced Placement curriculum.

Program analysis involves teaching the students about debugging programs, teaching them how to throw runtime exceptions, tracing and testing programs for correct output, and analyzing a program for efficiency.

Standard data structures not only teach the students about numerical representations and primitive data types, but also include the topics of data structures defined by classes, arrays (objects) and ArrayLists (classes). Also included are stacks, queues, priority queues, sets, maps, linked lists, trees and binary trees, heaps and hash tables.

Standard algorithms involves arrays, for example searching arrays with standard searches, sequential or binary searches and understanding which search

would be the best to use under certain circumstances. Iterators and hash tables are included in the understanding of standard algorithms.

Computing in context develops the student's knowledge of the main hardware components including how the software interacts, knowledge about the operating system, compiler, single-user or stand-alone systems, etc.

Ciezki (2008) summarizes the content of the Advanced Placement Computer Science Course by saying it

includes all of the topics covered in Advanced Placement Computer Science A, with more formal and in-depth study of algorithms, data structures, design, and abstraction. Students design, implement, and test computer-based solutions to a variety of problems in diverse application areas. These solutions are implemented using java and are comparable to those in the introductory sequence of courses for computer science majors offered in college and university computer science departments. The course prepares students for the AB-level Advanced Placement Computer Science Exam.

2.1.2 – AP Computer Science A

Students entering this course usually have no prior knowledge of programming. The focus on this course is developing problem-solving skills. Not all the topics that are covered in AB will be covered in A. The overlapping topics are taught at less depth. At the instructor's school, all of the students interested in taking the Advanced Placement Computer Science course have only the option of taking AB. However, at the end of this course, students have the option of taking the A Computer Science Exam rather than the AB Computer Science Exam. Generally,

students who select this option feel that they are not ready or prepared adequately to sit for the AB Exam. One prerequisite for this course is either having prior computer programming skills or having successfully passed the two introductory courses, Introduction to Computer Science and Intermediate Computer Science. These two semester-long courses are meant to be a thorough preparation for the AB curriculum. In addition, the math courses required as prerequisites are Algebra I and Geometry.

2.2 – Description of Existing Introduction to Computer Science Course

Currently, the topics for the Introduction to Computer Science course include classes, object-oriented programming, primitive data types, object behavior, libraries and APIs, simple input and output, loops, decisions, recursion and strings. There is time within the semester to assist the students in developing an understanding of object-oriented programming through role plays. Some articles are assigned for reading regarding programming and discussed through the discussion board on the Blackboard Web site or in class.

Since this introductory course is not critical to completing the Advanced Placement (AP) Computer Science course, it can be viewed as an enhancement to the curriculum, and any material covered would further assist students in their success in future courses. This gives greater flexibility in adjusting the course on an as needed basis. If successful, the project can continue along the same path. If not successful, the project can be reassessed and changed to adjust to the needs of the students. The uniqueness of the introductory course allows the instructor to cover only material that time will allow. In other math courses, a standard curriculum is outlined and

instructors need to ensure the courses cover the mandatory topics. In this introductory course, the instructor would not be bound to adhere to those standards.

The instructor's school has always put the needs of the students as the number one priority, and even math courses where there is a standard curriculum are often faced with making decisions about what absolutely needs to be taught and what can be eliminated so that the students are prepared adequately for their next course. Members of the mathematics department constantly discuss the impact of changes in the student body on curriculum as they see more students better prepared and, yet, still see more students needing to learn at a slower pace. This course would be no different in that regard. It is important that the instructor's methodology be flexible and grounded in the understanding that the initial approach taken may not be the way this course ultimately ends up being taught. It may change within the semester or from semester to semester.

2.3 – Research Resources

Several resources, other than the existing curriculum, will be utilized to determine the curriculum employed in this course. They are computer science textbooks regarding java programming (e.g., see Litvin, & Litvin, 2001), Advanced Placement College Board standards, Web sites from Advanced Placement teachers such as those included in or referenced at AP Central, and employees in the industry currently using java programming.

Chapter 3 – Methodology

The instructor will be using an iterative approach where each unit of the curriculum will be taken through all the phases of analysis, design, implementation, testing and maintenance. However, due to constraints within the instructor's teaching responsibilities, the instructor will not be able to actually work through the implementation, testing and maintenance stages. Nevertheless, the instructor will address how these stages would have been completed.

3.1 - Phase I: Analysis

Phase I will include observing the Introduction to Computer Science course currently in place and reviewing its teaching methodologies and curriculum. Based on the objectives of the Advanced Placement curriculum and based on research within the College Board community, the curriculum for the prerequisite course will be defined. The course objectives will be formulated for the Introduction to Computer Science course. These course objectives will be formulated by researching current trends in education, reviewing the general academic abilities of the students who typically enroll in the Introduction to Computer Science course at the instructor's school, and researching the goals of the school already in place for the student population.

3.2 - Phase II: Design

During this phase, the design of the daily classroom format will be outlined. Will there be warm-ups? What will the quizzes and tests look like, and how

frequently will they be administered? How will the assessments be graded? These are just some of the questions that will be answered during this phase.

Assignments, particularly programming problems to be used by the students to aide in developing their programming skills and their basic understanding of programming principles, will also be written during Phase II. Quizzes, test and warm-ups will be written during this phase, as well. Further discussions will be held about what teaching approach will be used to present the content of the new unit to the students.

3.3 - Phase III: Implementation

Implementation will consist of the written document outlining the proposed curriculum and an explanation of the methodologies and structure of the course. Through research, course objectives will be written that will become the foundation for the curriculum taught through the semester. If the instructor were to teach this course in the future, the instructor would implement the course objectives, teaching methodologies, assessments, and rubrics at that time.

3.4 - Phase IV: Testing

Testing will not be completed for this project since the instructor is not assigned to teach this course at the school. The project will address how the instructor would have completed these stages.

3.5 - Phase V: Maintenance

If the instructor were to teach the course, maintenance would be carried on by reviewing each unit upon its completion. This review would include assessment of the students' mastery of the material as well as an evaluation of their response to and

enthusiasm for the subject. Once studied, adjustments would be made to the next unit.

3.6 – Timeline

The project will be completed over a period of 8 weeks. During this period of time, research will be completed that supports the course objectives, teaching methodologies, software, textbooks, assessments, and rubrics used in the Introduction to Computer Science course. Decisions will be made as to the software and textbook that will be used for the course. Assignments, assessments, and rubrics will be written for the first unit of the course.

3.7 – Formats for Presenting Results and Deliverables

All homework assignments and worksheets will be compiled into a student workbook. This workbook will be the beginning of a student workbook for the course that would be available for purchase from the school store or posted on the school's Blackboard Web site for access by the students. All quizzes, tests, warm-ups and notes will be compiled into a teacher binder. The binder will become a resource for the instructor.

Chapter 4 – Project Analysis and Results

In general, developing curriculum for a high school course involves reviewing what is already in place, researching other available curriculums, studying what has and what has not been successful and researching the learning ability of the student population. Course development is a top-down approach in that the capstone course in a subject area dictates the course objectives for prerequisite courses.

4.1 – Functional Requirements

This project proposes the continued use of student workbooks as a curriculum guide the same as teachers use in other mathematics courses and current computer science courses. When developing courses at the instructor's school, the first time a course is taught, regardless of the standard curriculum outlined by the department, the course is taught 2-3 weeks at a time with the instructor staying 2 weeks ahead of the timeline. Then, based on the response from the students and reassessing the students' learning progress, the curriculum may change, resulting in more time being spent on a particular topic than originally planned or further homework assignments or assessments which may be written as needed. Workbooks for the following year will be developed based upon these assignments if they are assessed to be useful learning tools for the students. Every time a course is taught in subsequent years, these workbooks will be reassessed and rewritten as needs change. These workbooks will model the current department workbooks in that they will contain assignments for students and be a resource for notes for the course.

Therefore, the requirements for the project are:

1. A sampling of a student workbook containing the assignments written for the first unit.
2. A teacher notebook containing course objectives, master copies of all assessments and rubrics, some appropriate notes or lesson plans, or both, and a timeline for the topics covered.

4.2 - Goals

To develop a viable curriculum for the Introduction to Computer Science course, the goals are as follows:

1. To ensure that lessons plans and assignments align with course objectives.
2. To adequately prepare the students for the Advanced Placement Computer Science AB course.
3. To increase interest in the computer science program at the instructor's school as shown by the number of students enrolled in these courses (this is beyond the scope of this project and cannot be determined.)
4. To increase the percentage of students receiving a score of 4 or higher on the Advanced Placement Computer Science exam (this is beyond the scope of this project and cannot be determined.)

4.3 – Course Objectives and Teaching Methodologies

To develop a model for the Introduction to Computer Science course, the instructor has first examined and observed the course that is currently in place at the school. The instructor sat in the course daily for approximately six weeks, taking notes, observing the computer science course environment and students, particularly

how the students responded to the content, and the behavior, attention and knowledge of the students. The instructor also participated in some of the assignments by completing them outside of the class period.

The current Introduction to Computer Science course is set so that students complete all of their work during class time. The course meets daily for fifty-one minutes in the computer lab. Some days, direct instruction is given by the instructor. The rest of the time is for students to work on programs at their computers. The instructor is available to students to answer questions as needed. The program is designed so that the majority of work required to complete the course may be completed during the class period. A minimal amount of work is assigned to be completed at home rather than in the classroom. Occasionally, a few students will not have completed their assignment during class time and therefore are required to complete the work during their off-periods or at home.

Weighted grades are used with percentages assigned to the categories of programming assignments, quizzes, final exam and a general category labeled “other activities” that includes assessments such as discussions, worksheets, etc.

The programming language used is the java language.

Conley (2005) writes about what we can do, as high school teachers, to prepare our students for success in college.

Conley (2005) states,

High schools do not usually design their four-year instructional programs to be intellectually coherent. The focus is more on content coverage than on student development; classes and requirements do not consciously develop

student cognitive abilities or key learning skills. However, when Advanced Placement courses are given as capstone courses at the end of a carefully designed sequence that develops students' key knowledge and skills, these classes can help ready the students for college success. The advantage of conceiving Advanced Placement as a capstone for the curriculum is that the entire four years of the high school experience can then be aligned and organized around students progressively mastering core skills and concepts.

(p. 52)

To understand the students at the instructor's school and see what has worked to help the students be successful, it merits viewing the honors mathematics program. Students in this program will advance through Honors Geometry, Honors Algebra II and Trigonometry and Honors PreCalculus. After completing this sequence, the students are then prepared to take Advanced Placement Calculus BC, a capstone course. The mathematics courses in the honors track at the instructor's school are built with a foundation of teaching the students mastery of core skills and concepts. In addition to this foundation, the instructors of these courses work together to develop teaching methods that aid students in developing cognitive abilities and key learning skills. The success of this program is shown in the passing rate of the students taking Advanced Placement Calculus BC. The honors students are expected to work at a higher level and pace in the courses leading up to the Advanced Placement course. When taking Calculus, then, they not only have been taught the necessary mathematics skills to help them be successful in this course, but they have

been taught reasoning and study skills and have been taught conceptual understanding of the material.

The main focus of the Introduction to Computer Science course will remain to be preparing the students for success in the future Advanced Placement course, the school's capstone course for the computer science curriculum. The Advanced Placement Computer Science AB course is considered an introductory college-level course. The key to developing any courses leading up to the Advanced Placement Computer Science course would be to keep in mind how to help the students become college-ready so they may become successful in the Advanced Placement Computer Science course. The college-level course is different than the high school course. For this paper, the instructor will refer to the Advanced Placement course as the college-level course and the Introduction to Computer Science as the high school course.

According to a paper written by Conley (2007) for the Bill & Melinda Gates Foundation, it is important to look at the differences between what is expected of a student in a college course and what is expected of a student in a high school course. Knowing what is expected of a student in college will help high school instructors focus on skills necessary for the students to increase the probability of success in college.

Conley (2007) writes

The college instructor is more likely to emphasize a series of key thinking skills that students, for the most part, do not develop extensively in high school. They expected students to make inferences, interpret results, analyze conflicting explanations of phenomena, support arguments with evidence,

solve complex problems that have no obvious answer, reach conclusions, offer explanations, conduct research, engage in the give-and-take of ideas, and generally think deeply about what they are being taught. (p. 6)

Conley (2007) has used the term “key cognitive strategies” to identify the behaviors, learning styles and thinking skills necessary for students to be successful in a college-level course. He continues to explain specific key cognitive strategies which are briefly explained as follows:

1. Intellectual openness: The student is curious about ideas, thoughts and concepts within the classroom setting and is open to hearing all points of views from the instructor, other members of the class or other educational sources.

2. Inquisitiveness: The student questions ideas presented in class and the student is always asking why or looking for support for the reasons as presented in class.

3. Analysis: The student is able to compare different resources and analyze different ideas and thoughts to come to his or her own conclusion about the subject matter.

4. Reasoning, argumentation, proof: The student is able to argue a point of view and support that argument using valid proof.

5. Interpretation: The student is able to compare different points of view, explain the similarities and differences and explain these verbally or in writing.

6. Precision and accuracy: The student is able to judge the level of precision or accuracy needed in problems and reach that level accordingly. The student knows how to check for accuracy.

7. Problem solving: The student is able to apply a number of different strategies to reach the desired answer to a problem.

The instructor's school has implemented Conley's (2007) "Four Key Dimensions of College Readiness" and is using this model as a basic model for all courses within the school. The first level (or dimension) is the key cognitive strategies as mentioned above. These strategies are important goals for setting up the Introduction to Computer Science Course. (p. 12)

This course serves two purposes. One is as a preparation for the Advanced Placement course and an introduction to future courses of computer science that the student wishes to take to further the knowledge of computer science. The other is simply to provide a positive exposure to computer science that may or may not lead to further courses for the student. Because of this, there is a wide-range of abilities of the students who register for this course. Keeping this in mind, the course will need to be designed to meet the needs of both types of students – those with goals of pursuing the Advance Placement Computer Science course and those students who simply would like to understand more about computers or explore the possibility of an interest in computer science. The course goal will be for all students, if they so choose, no matter the ability at which they entered the course, to gain the necessary knowledge to enable them to advance into the Intermediate Computer Science course. At the intermediate level, they would continue their studies and enter the Advanced Placement Computer Science course, a college-level course.

Advanced Placement courses are very rigorous with intense work, critical thinking skills and daily homework equaling anywhere from one to three hours daily.

The introductory computer course needs to ease the student into this format in preparation for the possibility of the student furthering his or her education with the Advanced Placement Computer Science course. However, the needs of the other type of student will also need to be addressed.

Considering these various degrees of learning styles and goals, the course will be set up with daily assignments to be completed at home. Assignments completed outside of the classroom time are necessary to prepare the students for college-level course work. In a college-level course, students are required to complete hours of homework for each course. To continue this course as it stands, without much outside homework, is not preparing the students for the necessary course load required at the higher level. Conley (2005) stated, "It is not uncommon for students to experience difficulty in an AP course, not because they are intellectually incapable of mastering the content but because they have not been expected to work at that level or pace before." (p. 52)

Because of the diverse learning styles of the students in this course, the homework should target approximately 45-60 minutes each night. The assignments will focus on different skills, the main one being programming. Other skills will be writing, reading and researching to target various key cognitive strategies.

Since the main focus of homework will be programming, all students should have access to a computer at home or in computer labs on the school campus. Most, if not all, the students that register for this course have computers available to them at home. They are academically capable of setting up the proper computer environment on their home computer to accomplish what needs to be done with assignments on a

daily basis. For those students without access, there are computers in many labs throughout the school campus that the student can use to complete homework outside of class time in addition to the actual computer lab where the class is held.

The teaching strategies that will be used will help develop the key cognitive strategies as mentioned above. Students will use analytical techniques to solve applied problems. This will be through teacher-directed instruction allowing time for students to fully understand the process. Students are encouraged to question the approach as it is being developed, and the teacher will involve the students in developing the process.

Students will be encouraged to work collaboratively on their assignments. Extensive use of the school district's Blackboard Web site will be employed to not only post assignment schedules, worksheets, assignment answers, etc., but to encourage all students to participate in online discussions. Students who participate in the online discussions strengthen their communication skills while assisting other students in understanding computer science topics. Assistance or additional explanation by the instructor will be given as needed.

Evan et al. (2006) have listed the following items, reviewed in the following paragraphs, to make learning more rigorous and relevant. These items are alignment of the curriculum, using real-world examples, student awareness of the goals of the course, building student confidence, and diversifying instruction. Some of the highlights of this list are setting high expectations and challenging the students academically. When an instructor expects more from the students, they will rise to the occasion and meet that challenge.

Ensuring that the curriculum is aligned to help prepare the students for college demonstrates what the student will need to know and at what level of rigor they will need to perform to be successful in college. Developing projects within the curriculum or asking the students to answer more in-depth questions can help the students to think at a higher level.

Using real-world examples within the students' assignments builds interest in the students and helps them recognize the relevance of the content they are studying. This is particularly true in a more diverse classroom where the students come from various backgrounds. It is also more difficult for the instructor to creatively design assignments that will reach students of diverse backgrounds. What can aide the instructor in accomplishing this task is to become more familiar with the backgrounds of his or her students, attend diversity trainings offered within the educational community, and seek help from colleagues.

Students appreciate knowing the goals of the course. This helps them understand their role in achieving the goals if the goals are clearly stated by the instructor up front. Lesson objectives posted in the classroom on a daily basis assist the students in understanding what they should be able to accomplish before exiting the class for that day.

Students build confidence by knowing that an instructor is available for help when needed. Keeping up with the grading and giving the students this feedback in a timely manner helps the students keep track of where they stand academically. Each assignment can inform the students of mistakes that may have been made. If assignments are returned in a timely manner, the students have the opportunity to

correct what they have misunderstood or change their learning style and/or study habits to correct these mistakes on future assignments or assessments.

Every student may learn differently or test differently. Providing alternate means by which to assess a student's progress gives the student, and the instructor, a more informative and complete picture of the student's progress. (p. 37)

Keeping the above items from Evan et al. (2006) as a philosophy for the Introduction to Computer Science Course will help ensure the course maintains its rigor and that the students will be well prepared for the Advanced Placement Computer Science Course.

In summary, then, the course objectives for each student are as follows:

1. The student will set up a working java environment on his or her computer at home. This will include any software that is to be used in class such as Karel J. Robot (Bergin, 1997) and JCreator.

2. Homework will be assigned on a daily basis. These assignments are to be completed outside of class time and are due the next class meeting unless stated otherwise.

3. The student will be asked to justify his or her answers in class, as well as in homework, quizzes and tests. The student should be prepared to explain his or her justification or techniques verbally and in writing. Explaining in complete, but brief, sentences using the proper notation will be emphasized.

4. The student will be required to follow these in-class expectations:

- a. Bring all needed material to class. Textbook, paper, pencils and notebook are to be brought to class everyday.

- b. Be in your seat and ready to work when the bell rings and remain in class the full period.
- c. Obtain permission before speaking or leaving your seat.
- d. **Respect...**
 - yourself.** Always do your best, believe in yourself, and do your own work.
 - your classmates.** Listen to, appreciate the differences in, and support each other.
 - your teachers.** We have chosen this profession because of our passion to help you.
- e. Seek extra help when needed and take responsibility for material missed.
- f. Listen carefully when instructions or presentations are being given, taking good notes.
- g. Respect the property of other people and the school.
- h. Ask questions and be an active participant in the day's activity.
- i. Exercise integrity. The instructor expects that you will always function honestly and with integrity on every test and every assignment, as well as with every person in the class. Every time you take a quiz or a test you are taking two assessments. The first has to do with the mathematics; the second has to do with your character. If you can only pass one of them, make it the second.

The teaching methodologies to be employed by the instructor are as follows:

1. The Introductory Computer Science course will be taught to promote the key cognitive strategies needed to prepare students for college-level work. These strategies, used to promote a high level of student learning, include: a thorough understanding of the basic concepts, principles, and techniques of computer science, conceptual understanding of computer science concepts, applying learned concepts to “real world” situations, efficient problem solving strategies, evaluating the validity of solutions, and communicating understanding through written and verbal forms.

2. Students will learn through lectures, demonstrations, discussions, programming assignments, quizzes, exams and other activities.

3. With each new concept taught, the instructor will emphasize what it means to answer the problem and properly support that answer.

4. Quizzes will occur every Friday on any previous material in either the multiple-choice format or the free-response format of the AP exams. This is to help promote the consistent learning required for the students to maintain a high level of understanding.

The course objectives and teaching methods used for the course will be given to the students and their parents on the first day of the course. These will also be posted on the Blackboard Web site utilized by the school (see Appendix A).

4.4 – Java Environment and Textbook

Decisions for the course will need to be made concerning the java environment, the IDE (Integrated Development Environment) and the textbook(s) to

be used. The java environment is a free download from www.java.sun.com.

Instructions for the students are in Appendix C.

4.4.1 – IDE (Integrated Development Environment)

There are a few choices to consider for the IDE. Because of cost restraints, the IDE should be a free download. The system should be somewhat easy for students to install on their home computers. Also of consideration is the ease at which students can recover from mistakes or the extent to which the instructor would be able to assist the students in this process.

Currently, the introductory course and subsequent computer science courses are using JCreator. There are many options for free programming environments (IDEs) on the market. Other possibilities for use in the computer lab and for use on the home computers of the students are BlueJ, JBuilder, Eclipse and NetBeans. These are the more commonly used IDE's at the high school level and the more frequently discussed, referred to or used by the College Board community. There is also the choice of using java from the Command Line. However, for beginning computer science students, at a high school level, use of the Command Line would be expecting the students to perform at a level that would confuse them more than help them.

Another IDE, DrScheme, was discussed on AP Central as producing favorable results with students. North (2008) writes about her success using this IDE with struggling students and the program it is designed for – TeachScheme! She cites several examples of students who had found little success in computer science not to mention little success in other courses as well. Some of those cases involve girls who would have never imagined computer science as a career. Once they caught on, using

DrScheme, they continued on in the program and some have even continued into the computer science field through college and beyond. North (2008) believes this can be attributed to the step-by-step guide provided by Dr. Scheme and the fact that creativity is emphasized. This allows students to move from concrete to abstract thinking.

The disadvantage of using this type of program for students at the instructor's school is that it does not employ the use of java programming. Since one of the school's goals is to prepare the students for the Advanced Placement Computer Science course, the capstone computer science course, it logically follows that the students would be better served to begin their programming experience in the language used in the capstone course. This decision should be made with the understanding that the students are academically capable of meeting the challenges of a higher level of thinking. North (2008) does comment on her students also having difficulty in their math courses and having poor algebra skills. The students of the instructor's school registering in the computer science courses have been successful in their math courses. The prerequisite for registering for the computer science courses are Algebra I and Geometry. In addition, the math courses at the school, in general, are known to be more rigorous than math courses at other schools. Therefore, it stands to reason that the students at the school registering in the computer science courses, even at an introductory level, are capable of the rigor of using java.

North's (2008) comments about her students' success should not be taken lightly. There is truth in the fact that students, given the proper amount of practice time and simplifying the learning process to their abilities, can produce success. This

is an important piece in choosing the proper IDE for the classroom as well as applying to the curriculum as well.

Comments by the Advanced Placement Computer Science community on JBuilder have not promoted its use. JBuilder Enterprise is promoted in the programming community but it is not free. Turbo JBuilder is built on open source Eclipse and can easily be enhanced with powerful development tools. Little information is available for this in the Advanced Placement Computer Science community. Discussions from the instructors of the Advanced Placement Computer Science Course revolve around BlueJ and JCreator with occasional conversations supporting other IDEs such as jGRASP.

jGRASP is a free IDE that educators have said works well in the teaching environment. Cross (2008) writes about jGRASP's ability to "generate CSDs (Control Structure Diagrams). The diagrams are very useful in tracing flow of control throughout a program." Cross (2008) does agree that "BlueJ is the most sound pedagogical tool". But he sees the use of the CSDs in jGRASP as setting that IDE apart from BlueJ.

Greenfoot is an IDE that simplifies the process of writing games and simulations in the java programming language. The instructor's school has tried to use this in the computer labs. However, the technology department will only allow compiling over the home directory and not on the C-drive. Greenfoot requires compiling on the C-drive. Therefore, it is not an IDE that can be considered for the Introduction to Computer Science course.

Eclipse and NetBeans are popular IDEs with programmers in that they are robust and allow programmers to perform complicated tasks. However, the weakness in using these for beginning programming students at the high school level is that it becomes difficult to recover students from mistakes. For the Introduction to Computer Science course, it is advised to look for a simple IDE that would introduce the students to programming and allow the instructor to guide them through the process and the mistakes that may happen along the way.

JCreator Lite is a simple IDE that takes up little disk space. It is the current IDE that is being used in the Advanced Placement Computer Science course at the instructor's school. BlueJ was also tried with the students in the past. Students, however, found BlueJ to be too elementary and didn't feel that it challenged them academically.

Since JCreator is currently being used in the computer lab with the computers already loaded with this IDE, JCreator should be viewed as a possibility for the Introduction to Computer Science course. The fact that some students found BlueJ too elementary may be attributed to those students having an ability to learn programming at a higher level than most students in other schools who register for a beginning course.

BlueJ has the capability of running methods right from the object or class icons without writing a driver program. Educators of computer programming argue that using BlueJ teaches students more about how to use this program than about java programming. Friesen (2005) states that BlueJ simplifies the programming process in that the students would not have to understand classpaths and public static void main

(String[] args). The debugger in BlueJ also simplifies that process which promotes the use of the debugger by the students. The debugger assists the students' learning process in that it encourages the students to look for mistakes and move on to learning the basics of programming. In general, the simplicity of BlueJ's features allows the students to not be overwhelmed with learning the IDE and focus on learning java.

The web site for downloading BlueJ, <http://www.bluej.org/index.html>, provides extensive support. The help menu provides answers to technical questions. Also available are tips of the week and a discussion forum. For those students wishing to further their computer science education by registering for the Advanced Placement Computer Science course, the web site offers a download for the project, GridWorld, used in the AP curriculum.

There are some strong arguments in favor of using BlueJ for the Introduction to Computer Science course. The history of students at the instructor's school finding BlueJ too elementary continues to be an argument against use for this course. BlueJ's opening window prompts the student to set up classes and objects much like a class diagram would be used by programmers. The students do not type their own code. Students at the instructor's school enter this course with an interest in computer science and with a firm foundation from their mathematics courses. Therefore, for the Introduction to Computer Science course at the school, the instructor has determined that the course will begin with the use of JCreator as the IDE. Instructions for the students will be given the first week of the unit.

4.4.2 – *Software*

Karel J. Robot (Bergin, 1997) is software currently being used in the school's Computer Science courses. The software allows students to manipulate robots in a simple world of streets and avenues. The robots can be moved through this world, outlined on a grid, and perform simple tasks such as carry a beeper, a small plastic cone that emits a quiet beeping noise, pick up or put down a beeper, and locate a beeper. According to Slater (2008), one of Karel J. Robot's strongest assets is its ability to introduce students to java and object-oriented programming. Since this software is already owned by the instructor's school and it has proven to be a useful tool in teaching students object-oriented programming, it is economically feasible and operationally feasible to continue its use in the Introduction to Computer Science course.

Other software for use in the computer science courses has been considered at the instructor's school. The Alice software is one of those considered. Alice was developed by Carnegie Mellon University to aide beginning programmers. It is java-based and uses graphics to create programs. It is written in such a way that makes it interesting and fun for young programmers to learn about simple programming techniques. Alice is promoted as a program to reach girls and involve them in computer science. Taft (2007) reviews some studies done using Alice and how the results of those studies indicated Alice assisted young students in learning programming. Alice does seem, however, to be too elementary for high school students as the graphics used included dinosaurs, penguins, bugs, monkeys or fairies.

Since there are also boys in the Introduction to Computer Science course and there are various levels of ability, Alice will not be considered an option for this course.

Game Maker is another software package that has been tried in the computer lab at the instructor's school. It simplifies the process of making computer games. No coding is needed but learning the Game Maker language is required. The language uses loops, branching and tools that reference objects. Thus it is a good way to interest students in programming and present to them some primitive programming techniques. However, the technology department at the school has not been able to connect the sound cards in the computers in the lab thus preventing the software from working properly. The software may be considered at a later date providing the technology issue has been resolved.

4.4.3 – Textbooks

The school's Computer Science courses currently use a textbook by Quesenberry (2006). Due to economic feasibility, the use of this textbook will continue to be used in the Introduction to Computer Science course. It is a comprehensive textbook of the Advanced Placement Computer Science curriculum. It includes electronic versions for posting on the Blackboard Web site. The textbook is divided into 33 lessons. Each lesson includes a content outline with a detailed explanation of each content area. The lesson ends with an assignment for the students to complete.

Educators understand that one textbook will not fit the needs of the students and the goals of the course. The best an educator can do is use a textbook that will fit most of the needs of the course. Then the educator will employ the use of worksheets

and assignments developed from other resources. Therefore, it is worth mentioning here other textbooks that may provide resources and possibly be replacements in the future. Also, if it becomes economically feasible and the progress of the students show an alternative textbook would better fit the needs of the students, other textbooks should be reviewed.

Litvin & Litvin (2001) have coauthored a textbook that introduces students to java programming. The textbook explains the details of java programming mainly through well-written chapters and through the use of examples. It begins with explaining hardware and then proceeds to explain software development before beginning object-oriented programming. Object-oriented programming is taught using three levels of the language, i.e. syntax, the conceptual object-oriented programming level, and the intermediate layer of library classes, at one time. Even though the students are not expected to understand all three of these levels from the very beginning of instruction, the students are expected to take as faith that the ideas will eventually come together and understood fully as more is taught through the course. A companion web site is provided with the textbook that contains further explanations and code for case studies, labs, and exercises. A teacher's disk is available with solutions for those courses using the textbook in the classroom.

A textbook that specifically focuses on the Advanced Placement exam is one written by Lewis, Loftus, & Cocking (2004). Its strengths are that it is written with the academic level of a high school student in mind in that there are more color-coded figures and less writing. Each chapter ends with a summary of key concepts and self-review questions with the answers available. The exercises at the end of the chapter

contain various forms of problems such as multiple choice, true or false and short answer questions. There is also a very adequate set of programming projects from which to choose. If the textbook is used in the course, there are supplementary materials available for the students and instructor such as online assistance, instructor's manual, solutions, test bank and lab manual.

An interesting approach is found in a textbook by Barnes & Kolling (2003). This textbook begins with Chapter 1 covering object-oriented programming rather than the traditional hardware introduction covered in other textbooks. They use an iterative approach in their style in that the main topic, objects, is introduced first, then each subsequent discussion of a new topic will incorporate previous topics. BlueJ is the IDE used throughout the book and the authors write extensively about the justification for their use of BlueJ (p. xix). Exercises for the students are scattered through the chapters. Each chapter ends with a concept summary list. As with other textbooks, online support is available, a CD comes with the textbook for the students to download the java environment and BlueJ, and additional teacher support is available in the form of electronic copies of the examples in the textbook, slides to teach the course, etc.

4.5 – Lesson Plans and Daily Assignments

Assignments will be given to the students daily. Some assignments will take more than one day to complete and therefore time will be given to the students to work on those assignments in class. The assignments will be compiled into a table. The table will contain the content that will be discussed in class each day as well as the assignment for the student to complete that evening. Each student will be given a

hard copy of the table. It will also be posted on the Blackboard Web site. Table 1 is for the first week of the Introduction to Computer Science course.

Table 1: Introduction to Computer Science Course Daily Content and Assignment Schedule, Week 1

LESSON	CONTENT	ASSIGNMENT
1	<ul style="list-style-type: none"> Review Course Objectives with students Instructions on enrolling on Blackboard Web Site 	<ul style="list-style-type: none"> Enroll in course on Blackboard Web Site Tour Site and find the hidden message.
2	<ul style="list-style-type: none"> Instructions on downloading java and JCreator Role Play (Group 1) 	<ul style="list-style-type: none"> Download java and JCreator onto home computer Respond to prompt posted on Blackboard Web Site in discussion group (All About Me)
3	<ul style="list-style-type: none"> Review any difficulties students had with home computers Role Play (Group 2 & 3) 	<ul style="list-style-type: none"> Respond to prompt posted on Blackboard Web Site in discussion group (All About Me)
4	<ul style="list-style-type: none"> Introduce Classes, Objects, Methods 	<ul style="list-style-type: none"> Read Article on Kutztown, PA. (link on Blackboard); follow "Currents in Technology" Guidelines.
5	<ul style="list-style-type: none"> Continue discussion of object-oriented programming Introduce JCreator 	<ul style="list-style-type: none"> Write small program named Wizard; print "The Great Oz has spoken!"

MicroSoft PowerPoint (PowerPoint) lessons will be developed for the instructor's use when needed. In most cases, the suggested use for the PowerPoint slides is as a reference for the instructor. These slides are not designed to be used as a presentation to the students unless otherwise specified. The instructor will guide the students through the lesson with questions and answers rather than letting the students anticipate what is going to be discussed by seeing what is on the slide beforehand.

4.5.1 – Lesson 1

On the first day of the course, the instructor will review the course objectives and classroom expectations with the students. Each student will receive a hard copy which will be signed by the instructor (see Appendix A). Attached will be a form for the student's parents to sign indicating they have read and understand the objectives (see Appendix B). A PowerPoint presentation will be utilized to review the objectives with the students on the first day (see Appendix E).

Also on the first day, the instructor will take the students through the process of registering for the Blackboard Web site (see Appendix F). The Blackboard Web site will be an instrumental tool for learning throughout the semester. The students will participate in discussions, reference notes for the course, and view assignments on this site. The students' assignment for the first night is to tour the Blackboard Web site for the hidden message placed there by the instructor. They are to record what the message is and where they found it and hand it in for the next class.

4.5.2 – Lesson 2

The second day of instruction will begin with demonstrating to the students how to download the Java Development Kit (see Appendix C) and JCreator (see Appendix D). The students will also receive hard copies of these instructions and they will be posted on the Blackboard Web site. The steps and screen captures are as follows:

1. Go to <http://java.sun.com> and find the download page (see Figure 1).



Figure 1: Download Page for Java Development Kit

2. Click on “Java SE Development Kit (JDK) 6 Update 12” (see Figure 2).

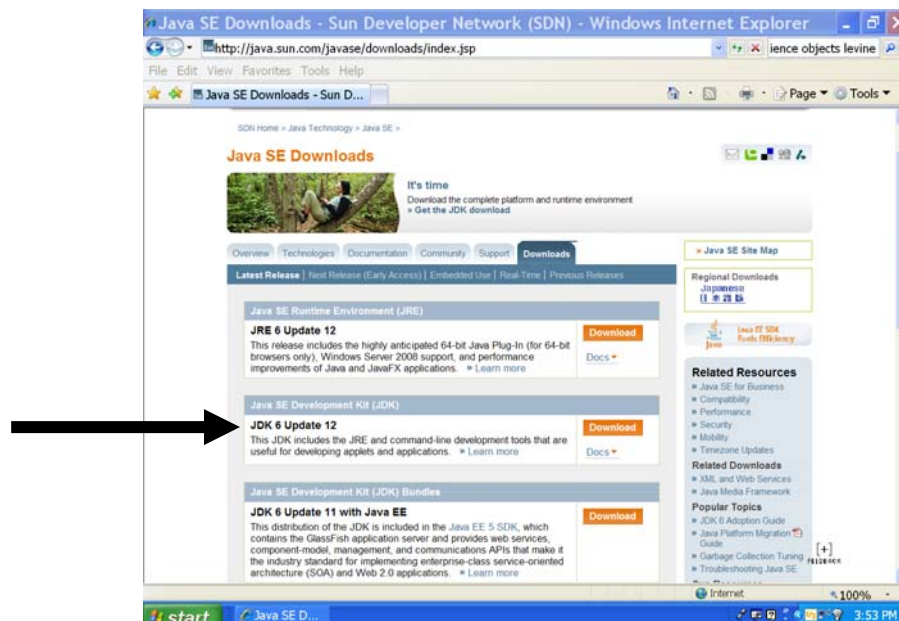


Figure 2: JDK 6 Update 12 Download Page

3. Select the correct download of Java SE for your machine. For Windows, select the Windows (all language) download of JDK (see Figure 3).

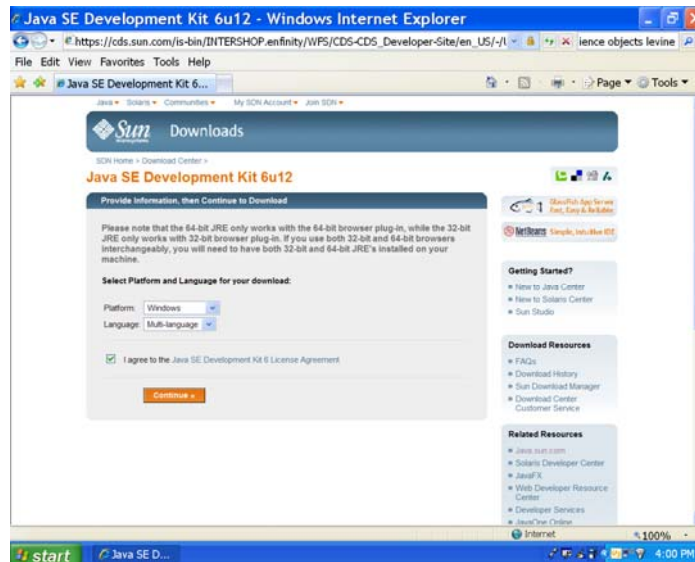


Figure 3: Select Correct Java SE Download for Machine

4. Click on Continue. Click on the file name (see Figure 4). Click RUN. Follow installation instructions.



Figure 4: Download Selected SE Development Kit

5. By default, it should be in “Java” within “Program Files” on your C: drive
(see Figure 5).

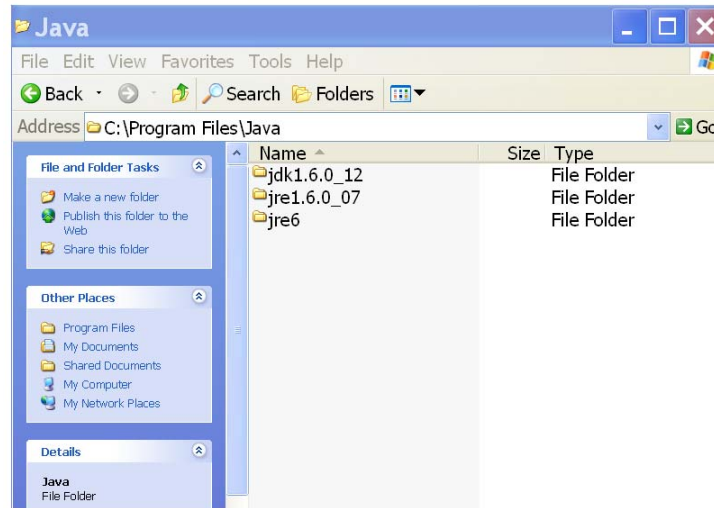


Figure 5: Java JDK Path File

6. Go to <http://java.sun.com/javase/downloads/index.jsp> and download the Java SE 6 Documentation (see Figure 6).



Figure 6: Download Java SE 6 Documentation

7. Select English and the agree box. Click Continue.
8. Click on the file name as was done in Step 4 above. Click Save.
9. Save in C:\Program Files\Java\jdk1.6.0_12.

The instructions for downloading JCreator are listed below. The students will also need to copy jar files into the library for use during the course.

1. Go to <http://www.jcreator.com/download.htm> and go to the download page (see Figure 7).

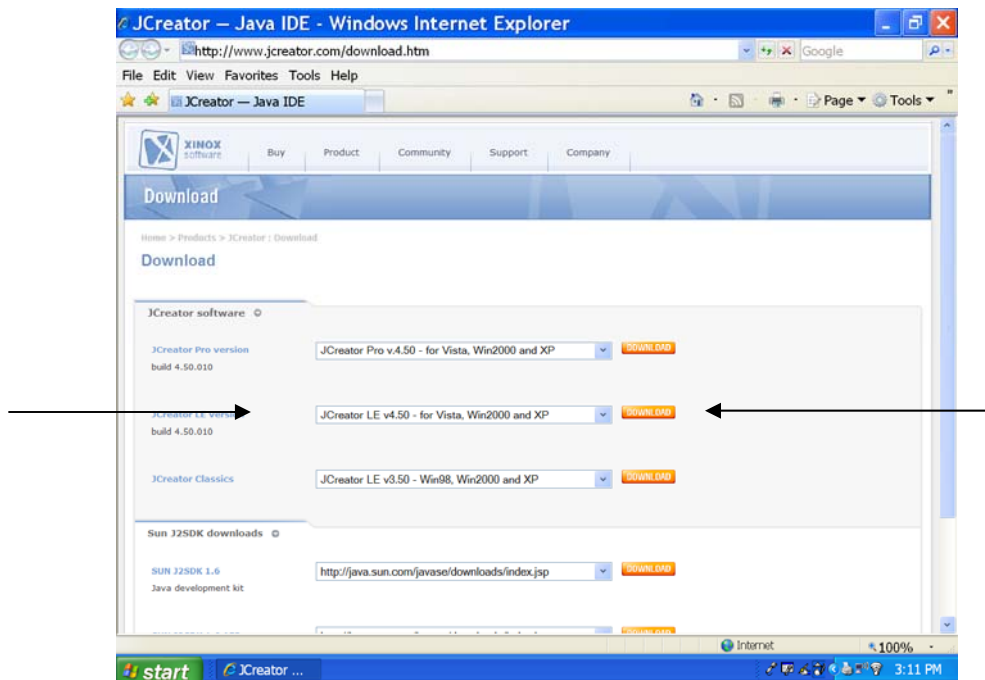


Figure 7: Download Page for JCreator

2. Click on Download. There are installation instructions on the JCreator web site. Use these instructions to configure default properties. When JCreator is launched, follow the Setup wizard (see Figure 8) and set the paths to C:\...\jdk.1.6.0_12\ and C:\...\jdk1.6.0_12\docs.

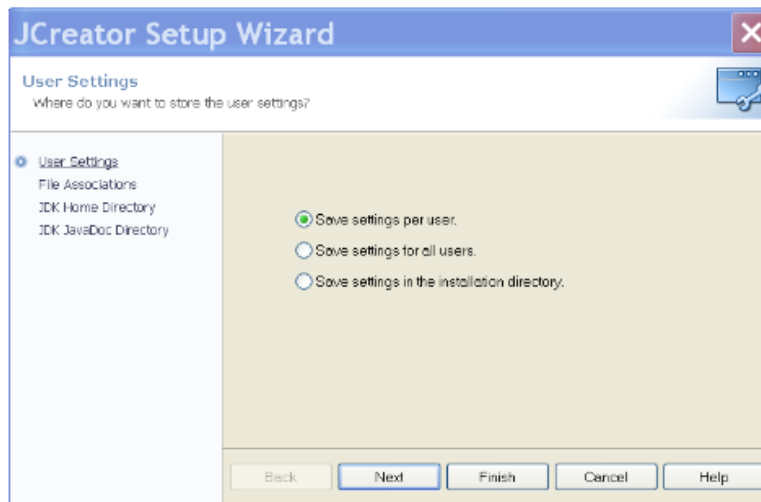


Figure 8: JCreator Setup Wizard

3. Go to the Blackboard Web site. Click on the tab marked “Course Notes”.
Click on the link for “Installation Directions”. Right click on the jar file and select “Save Target As”. Save to a folder on the computer or save to the desktop.
4. In order to use libraries (packages) in JCreator, open a workspace with a project. Go to Project/Project Settings (see Figure 9) and click on the Required Libraries tab (see Figure 10). To add a library, click New, click on the Classes tab and enter a name of your choice for the set of jars in the dialog box (see Figure 11). Click on Add and choose Add Archive (see Figure 12). Browse to the location that you saved the jar files in Step 3. Double click on the file until the file or folder is in the list. Repeat until all of the jar files from Blackboard are in the list (see Figure 13). Check the box next to the folder name to make it available in any project (see Figure 14).

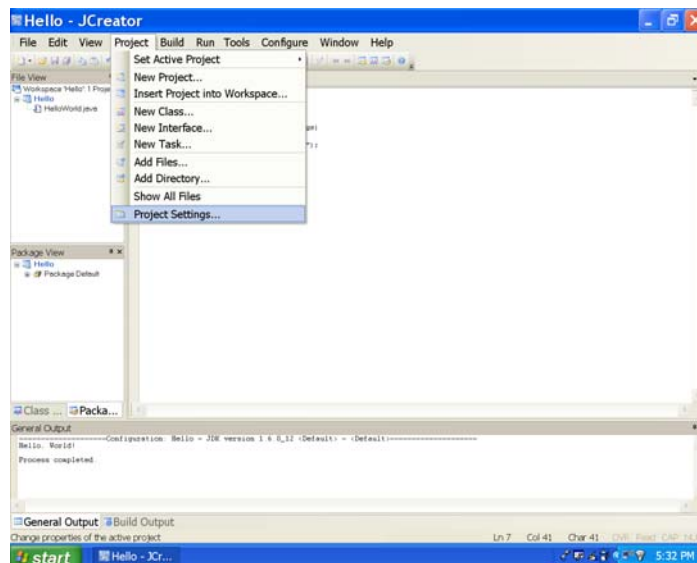


Figure 9: JCreator Project/Project Settings

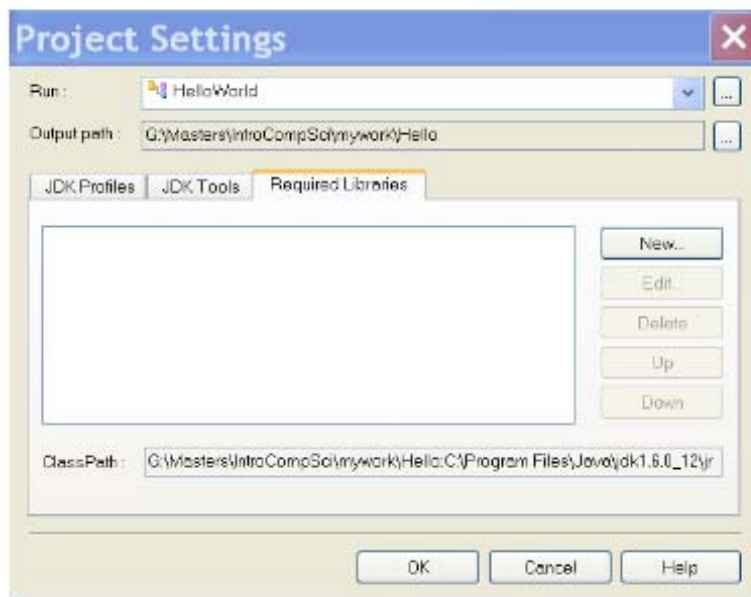


Figure 10: JCreator Required Libraries tab

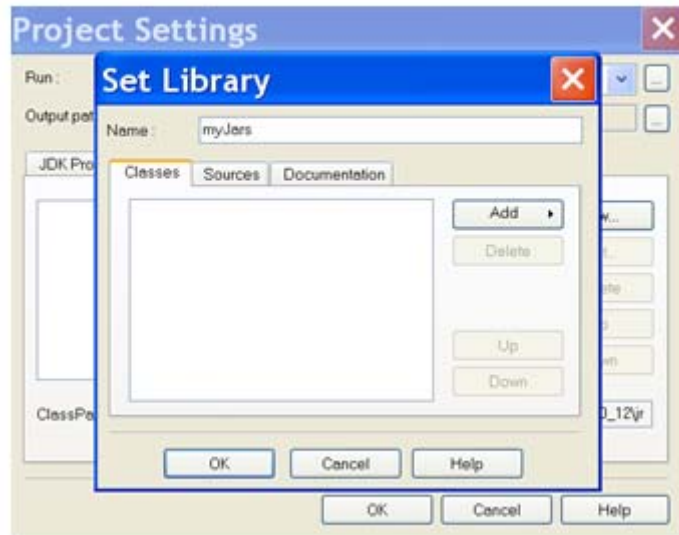


Figure 11: JCreator name and add jar file folder



Figure 12: JCreator Add Archive selection

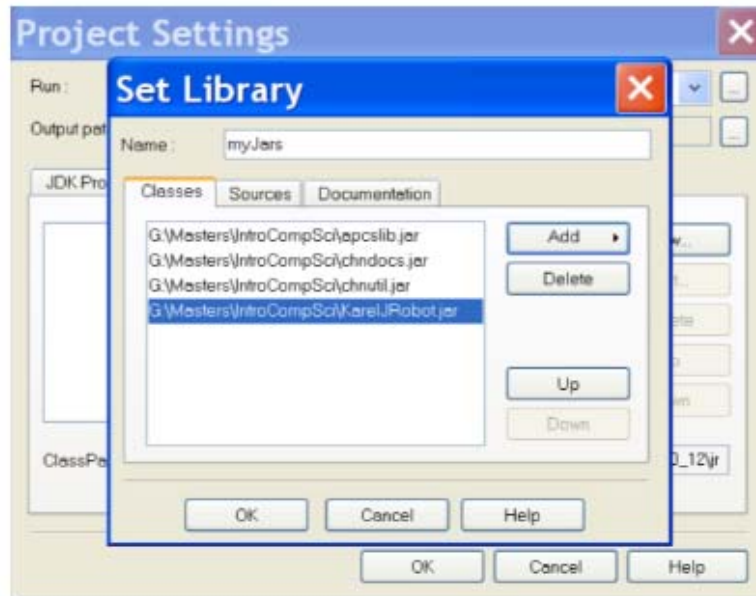


Figure 13: JCreator jar files list

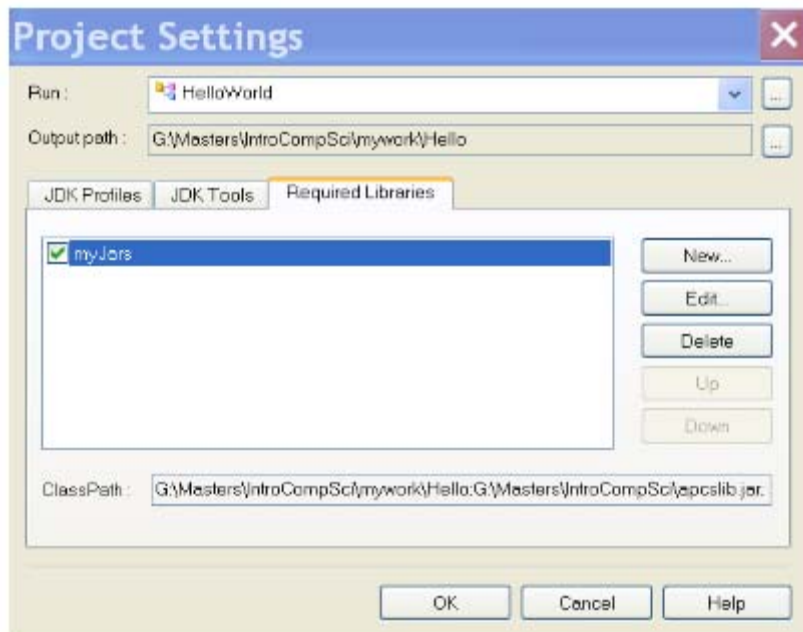


Figure 14: JCreator making set of jars available

After the instructor has demonstrated the process of downloading both the java JDK and JCreator to the computer, the students will be divided into three groups. One group will participate in the role play activity while the other two groups will

begin their assignment for that day’s lesson. The assignment to be completed in class is for the students to write information about themselves on the Blackboard Web site. They will be following the prompts in Figure 15, but do not necessarily have to answer all the prompts. They also may add information not necessarily asked for but rather information they would like the instructor to know. All students will be assigned the process of downloading the java JDK and JCreator to their home computers.

Slide 1

The slide features a blue header with the word "Information" in white. Below the header is a list of six prompts, each preceded by a light blue circular bullet point. To the right of the list are seven horizontal lines for writing.

- On Blackboard, answer prompt “All About Me”
- Write about yourself and your family.
- Why you are taking this course.
- What you hope to learn in this course.
- Other school activities.
- Activities outside of school.
- Favorite movie, book, song, etc.

Figure 15: "All About Me" Prompt

While group two and three are working at their computers on the “All About Me” prompt, the instructor will lead the first group in the role play activity. The role play to be used is written by Levine (2007). It is used to introduce beginning programming students to classes, objects and methods. Each student is assigned a role such as an acrobat, a blackboard, a calculator, a bamboozler, a lazy calculator or a decider. They are given a paper that describes what they can and cannot do and how they are to do it. For example, Levine (2003) writes an acrobat’s instructions as:

When you are asked to **Clap**, you will be given a number. Clap your hands that many times.

When you are asked to **KneeBend**, you will be given a number. Stand up and sit down that many times. Note that if you are told “2”, then you will stand up twice AND sit down twice.

When you are asked to **Count**, you will reply (verbally) with the total number of exercises you have done. Note that Clap-ping four times counts as four exercises, and KneeBend-ing twice counts as two. If you have done these things (and only these things) then your reply should be “6”. (p.1)

The instructor has a script to follow that prompts the students through their roles. The instructor can ask a student, by name, to construct himself or herself as an acrobat. Then the instructor can call on the student to perform methods such as “Acrobat, clap four”. The instructor can also ask for things that they do not know how to do or an incorrect instruction such as asking a class, instead of an object, to perform certain tasks. Cutler (2005) states how he just has to say “Acrobat clap” during the school year and his students will understand that a mistake was made from someone calling a method with the class name instead of the object name (p. 4). After the students adjust to their roles, the instructor can also begin to introduce java code. For example, when a student has been called to perform a method, the instructor would write on the board “luke.clap(4)”. The role play activity helps the students visualize writing programming and makes it fun in the process.

4.5.3 – Lesson 3

The third day of the course will begin with problem solving any difficulties the students encountered while downloading the java JDK, JCreator or the jar files. The discussion will involve all of the students as they try to help troubleshoot problems other students were having with the process.

The rest of the lesson will look like lesson 2 in that the other two groups will be led through the role play activity by the instructor. Any group not involved in role play, will be working on their post on the Blackboard Web site using the prompts from “All About Me.”

4.5.4 – Lesson 4

The objective for lesson 4 is for the students to be introduced to the definitions of objects, classes and methods. The lesson plans will review the student lesson found in the textbook by Quesenberry (2006) which is being used for the course. This lesson is A1 – Introduction to Object-Oriented Programming (OOP). In the textbook, the purpose of this lesson is stated as follows: “The purpose of this lesson is to give you a feel for object-oriented programming and to introduce its conceptual foundation.” (p. 1). The students will be directed to the location on the Blackboard Web site where to find this lesson as well as all lessons from this course. If the student has chosen to purchase the hard copy of the textbook from the school store, they will be directed to the lesson in that copy. A PowerPoint file (see Appendix G) may be used as an aide for the instructor in leading the students through the discussion.

The homework for the evening will be to read the article by Rubinkam (2005) on Kutztown, PA. The link will be available to the students on the Blackboard Web site. The guidelines and directions outlined by N. Quesenberry (personal communication, January 30, 2009) for this assignment are described in Appendix H. These same guidelines and directions will be used throughout the semester for an assignment pertaining to reading and responding to articles from the media.

4.5.5 – Lesson 5

The objective for lesson five is to introduce the students to a small program and programming syntax (see Appendix I). The students will see how comments are used in a program, why they are used and will be given examples of what types of comments are useful. The syntax for class definition will be introduced as well as the main method. The students will see how to configure JCreator and will work through the steps at their computers with the guidance of the instructor. The instructions from Litvin (2003) will be referenced to assist the students in configuring JCreator on their home computers. The instructor will demonstrate the writing of a small program called “Hello, World!” This program will instruct the students about class definition, main method and will also introduce the `println` method (see Figure 16).

The assignment for lesson five is to write a small program named Wizard that will print out the phrase “The Great Oz has spoken!” (see Figure 17). This will ensure that the students have configured their home computers properly and can run JCreator. Since it also will require them to type, compile and run the program, they will experience how to find errors, correct them, rebuild and run the program

successfully. The students will be required to hand in a printout of the program from their home computers.

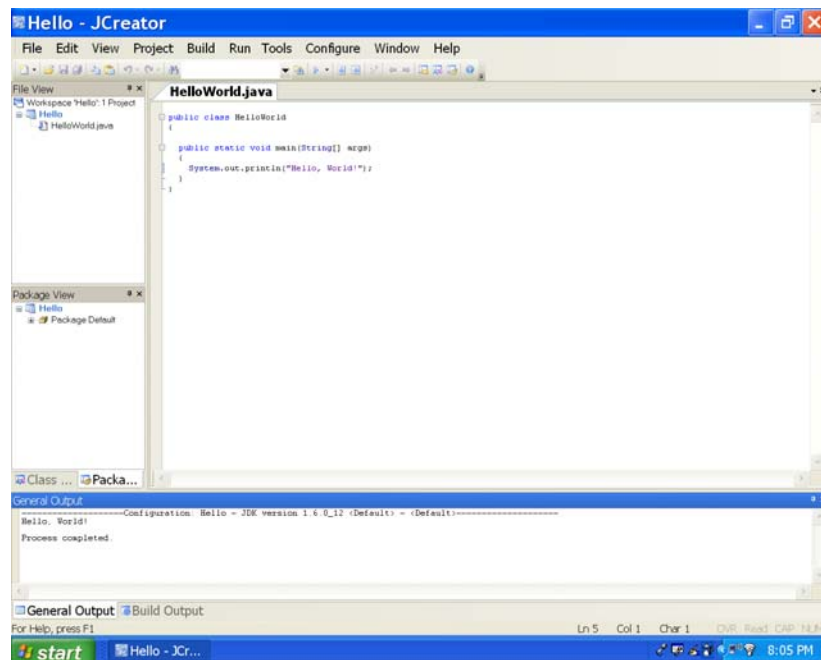


Figure 16: Lesson 5 program, Hello, World!

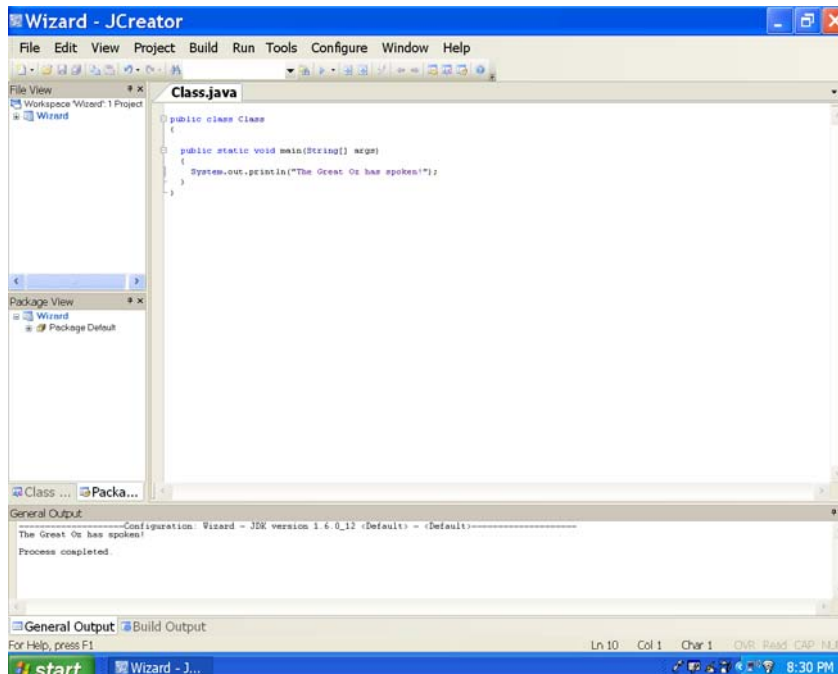


Figure 17: Lesson 5 assignment "The Great Oz has spoken!"

4.5.6 – Lesson 6

The objective of lesson 6 is to connect what the students did in role play to java programming and code. The students will participate in a discussion on what each person was asked to do during role play. The instructor will guide the students into seeing the connection on what was done during role play and how that is written into java code. A PowerPoint file (see Appendix J) may be used as an aide for the instructor in the discussion.

The next week’s assignments will be distributed to the students (see Table 2). Lab time will be given so that the students may work at their computers on programs. Students who have already completed the assignments without difficulty will be asked to assist students who have not been able to complete the assignments due to errors in programming. The instructor will also guide the students in finding errors.

Table 2: Introduction to Computer Science Course Daily Content and Assignment Schedule, Week 2

LESSON	CONTENT	ASSIGNMENT
6	<ul style="list-style-type: none"> • Go over Role Play • Lab time; answer questions; • Clarify concepts 	<ul style="list-style-type: none"> • None
7	<ul style="list-style-type: none"> • DrawSquare Program 	<ul style="list-style-type: none"> • DrawHouse
8	<ul style="list-style-type: none"> • Lab time 	<ul style="list-style-type: none"> • Continue work on DrawHouse
9	<ul style="list-style-type: none"> • Lab time 	<ul style="list-style-type: none"> • ICT Worksheet A1.1 • SmileyFace
10	<ul style="list-style-type: none"> • Review Differences Between Objects and Classes 	<ul style="list-style-type: none"> • Quiz 1 • Worksheet, Benzene • ICT Worksheet A2.1 • ICT Worksheet A2.2

4.5.7 – Lesson 7

The objective of lesson 7 is to expand on what has already been learned about objects, classes, and methods. The discussion will consist of the instructor and students designing a program together, called DrawSquare (see Figure 18). The students will be following along on their computers and writing the code as instructed. The instructor will lead the discussion as outlined in Appendix K (Quesenberry, 2006), explaining object-oriented programming and the use of JCreator as this program is being developed. It is not expected, at this time, for the students to understand all the coding instructions. The goal is to understand more about object-oriented programming.

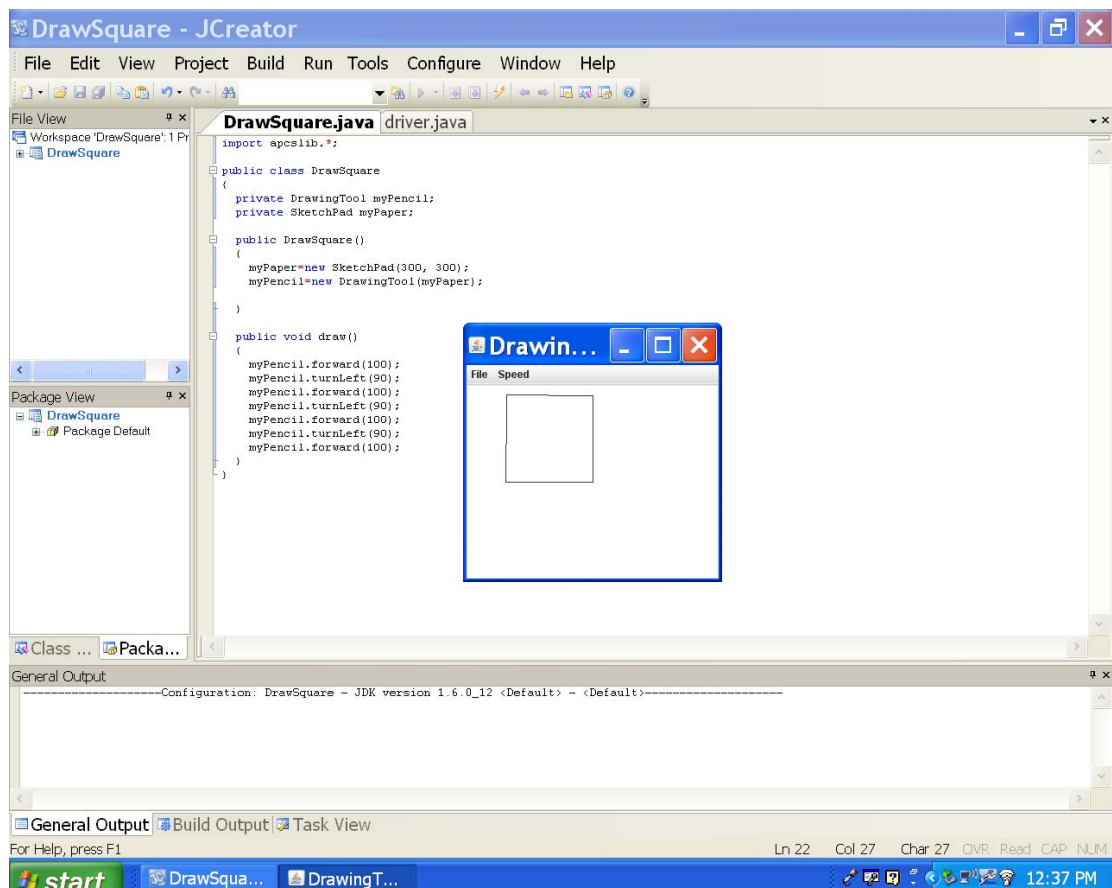
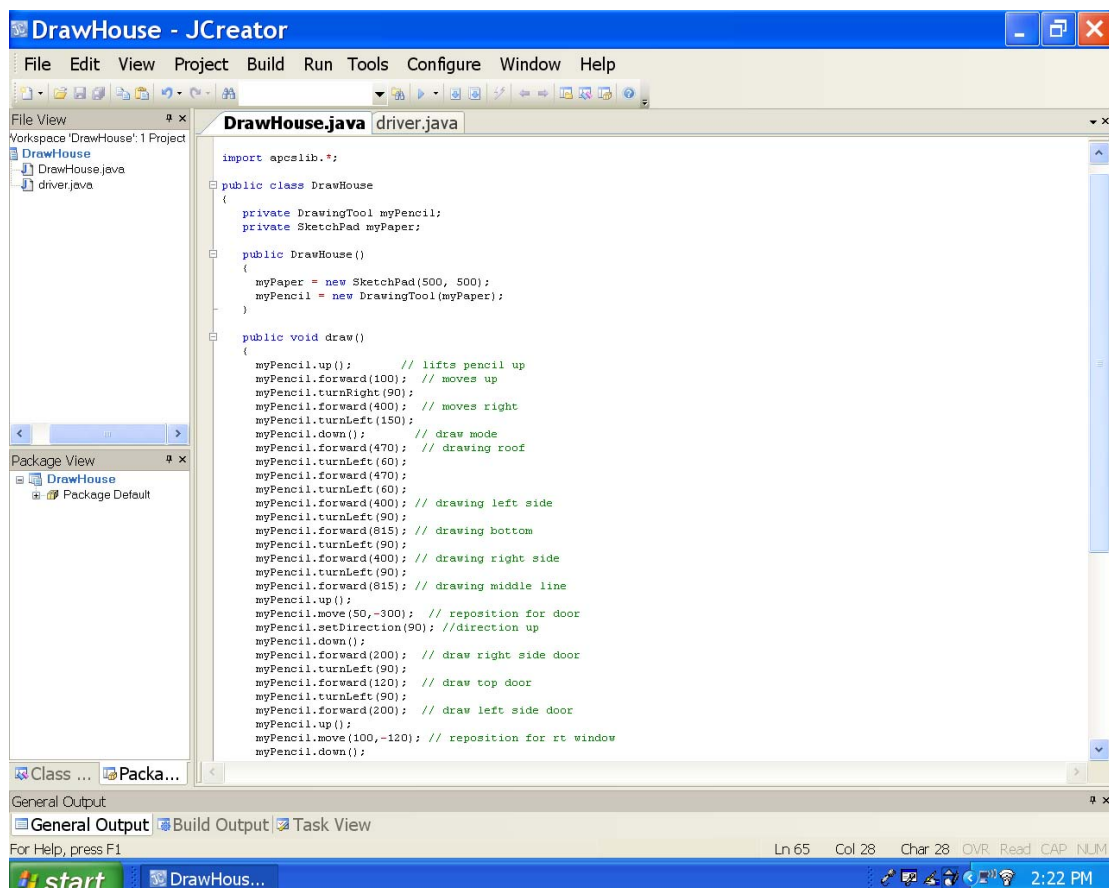


Figure 18: Lesson 5 Program DrawSquare and output

The students will receive two handouts for this lesson. The first handout lists the DrawingTool class specifications from the Institute of Computer Technology (ICT) curriculum that will be used in developing the program. This handout lists some of the features of the class, such as the object DrawingTool, constructor methods, accessor methods, and modifier methods. The second handout is for the students' assignment. The assignment is for the students to create a program, called DrawHouse (see Figure 19), which does what the name implies (see Figure 20).



```
import apcslib.*;

public class DrawHouse
{
    private DrawingTool myPencil;
    private SketchPad myPaper;

    public DrawHouse()
    {
        myPaper = new SketchPad(500, 500);
        myPencil = new DrawingTool(myPaper);
    }

    public void draw()
    {
        myPencil.up(); // lifts pencil up
        myPencil.forward(100); // moves up
        myPencil.turnRight(90);
        myPencil.forward(400); // moves right
        myPencil.turnLeft(150);
        myPencil.down(); // draw mode
        myPencil.forward(470); // drawing roof
        myPencil.turnLeft(60);
        myPencil.forward(470);
        myPencil.turnLeft(60);
        myPencil.forward(400); // drawing left side
        myPencil.turnLeft(90);
        myPencil.forward(815); // drawing bottom
        myPencil.turnLeft(90);
        myPencil.forward(400); // drawing right side
        myPencil.turnLeft(90);
        myPencil.forward(815); // drawing middle line
        myPencil.up();
        myPencil.move(50,-300); // reposition for door
        myPencil.setDirection(90); //direction up
        myPencil.down();
        myPencil.forward(200); // draw right side door
        myPencil.turnLeft(90);
        myPencil.forward(120); // draw top door
        myPencil.turnLeft(90);
        myPencil.forward(200); // draw left side door
        myPencil.up();
        myPencil.move(100,-120); // reposition for rt window
        myPencil.down();
    }
}
```

Figure 19: Assignment DrawHouse code

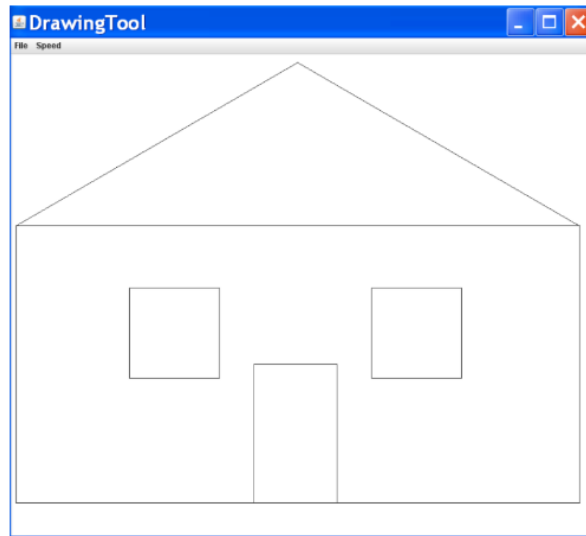


Figure 20: Assignment DrawHouse output

4.5.8 – Lesson 8

Since the majority of class time on lesson 7 was spent with discussion and demonstration, lesson 8 will be lab time for the students to continue to work on their projects of developing the program DrawHouse. During this time, the instructor will be available to help the students. The instructor will also encourage students to help each other and encourage them to try the optional piece of the assignment which is to add a door and windows.

4.5.9 – Lesson 9

Lesson 9 will be another day of lab time where the students will extend their understanding of the DrawingTool and SketchPad objects. They will use the lab time to complete the next worksheet assignment from the ICT curriculum. Two questions on this worksheet give the students code. They are to read through the code and draw

the figure generated by following the code directions. The remaining three questions have the students write code (see Figure 21).

4.5.10 – Lesson 10

The first part of lesson 10 will be a discussion, as outlined in Appendix L, about the differences and similarities between objects and classes (Quesenberry, 2006). The remaining class time will be used for lab work. During this time, the students will be completing a program that draws the abbreviated symbol for the chemical benzene (see Figure 22). There will be two worksheets from the CIT curriculum for the students to complete for homework. The first worksheet reviews objects and classes and contains short-answer questions. The second worksheet also contains short-answer questions and reviews object-oriented programming.

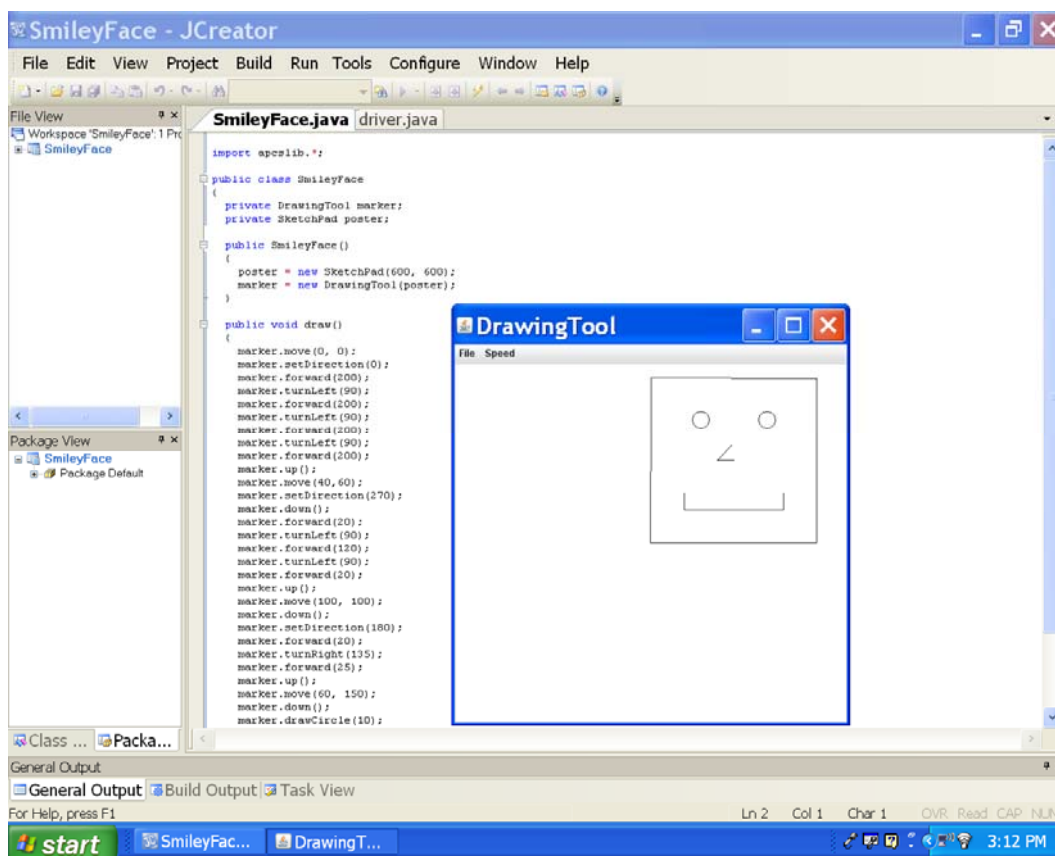


Figure 21: SmileyFace Assignment Code & Output

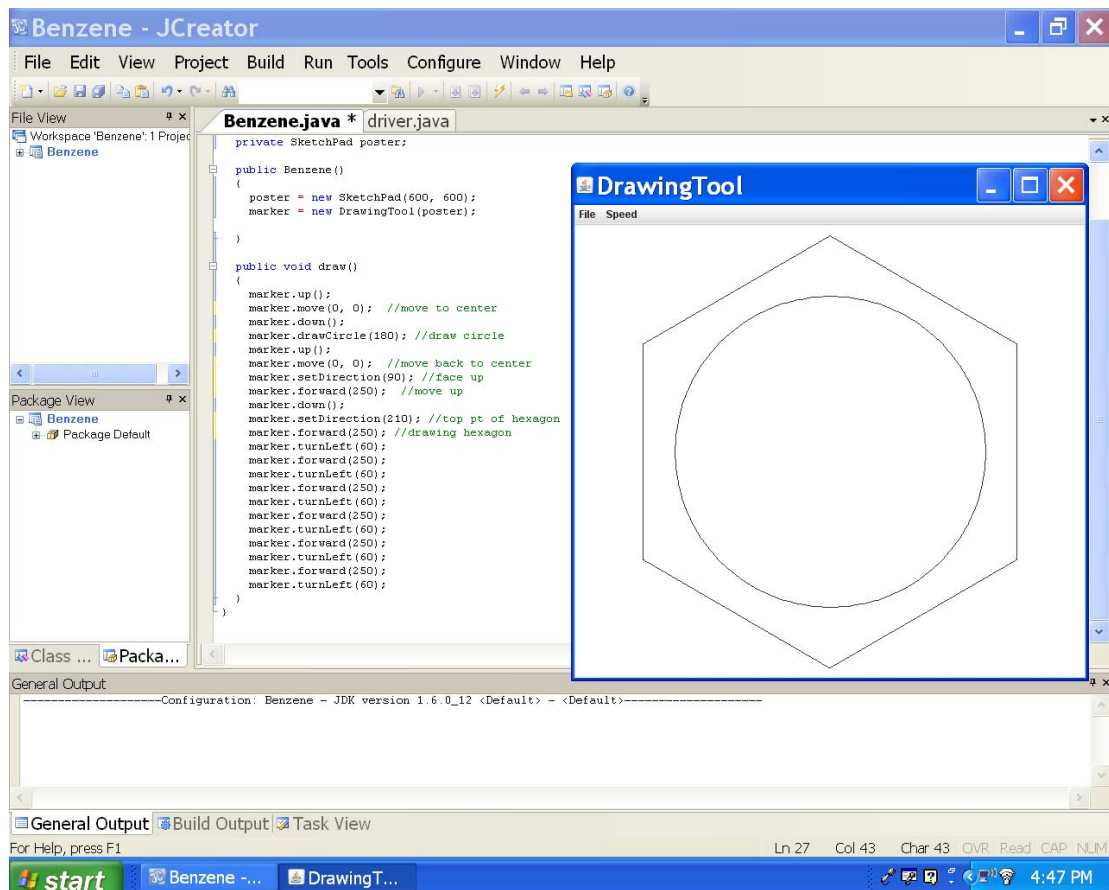


Figure 22: Lesson 10 Benzene Program

4.5.11 – Lesson 11

At the start of the third week, the students will be given the assignments in a table format (see Table 3). This week, the course will begin using the Karel J. Robot class (Bergin, 1997). There are many advantages in using Karel J. Robot for teaching object-oriented programming. The visual feedback helps the students see their mistakes and successes. The variety of robot tasks helps the students in understanding major concepts such as inheritance, polymorphism, abstraction, encapsulation, object-oriented programming design, recursion, and iteration. The students enjoy using the programming as it filters out java details so they can focus

on the major concepts. As the semester goes on, the instructor can spiral back through those topics introducing more and more detail as required.

Lesson 11 begins with a PowerPoint presentation to the students (see Appendix M) about the Karel J. Robot World. The textbook for Karel J. Robot comes with Problem Sets for each chapter. For homework, the students will answer the questions from the Problem Set for Chapter 1.

Table 3: Introduction to Computer Science Course Daily Content and Assignment Schedule, Week 3

LESSON	CONTENT	ASSIGNMENT
11	<ul style="list-style-type: none"> • Karel J. Robot, Chapter 1 	<ul style="list-style-type: none"> • Karel J. Robot Chap 1 Questions
12	<ul style="list-style-type: none"> • Karel J. Robot, Chapter 2 	<ul style="list-style-type: none"> • Initials • Karel J. Robot Chap 2, #1 & #2
13	<ul style="list-style-type: none"> • Lab time 	<ul style="list-style-type: none"> • Karel J. Robot Chap 2, #5 (newspaper) & #7(grocery)
14	<ul style="list-style-type: none"> • Lab time 	<ul style="list-style-type: none"> • Karel J. Robot Chap 2, #6 (mountain) • Karel J. Robot Chap 2, #9 (Figure 8)
15	<ul style="list-style-type: none"> • A1 to A2 of ICT, Karel J. Robot 	<ul style="list-style-type: none"> • Unit Test 1

4.5.12 – Lesson 12

The objective for lesson 12 is to understand the Karel J. Robot language and to write programs that instruct robots to perform simple obstacle avoidance and beeper transportation tasks. A PowerPoint presentation will be used to instruct the students in this goal (see Appendix N). After the presentation, the instructor will lead the students in writing a program to “draw” the letter “H” using beepers (see Figure 23). The assignments for the students will be to answer questions one and two from

the Karel J. Robot Problem Set of Chapter 2 (Bergin, 1997) and write a program that will “draw” their initials.

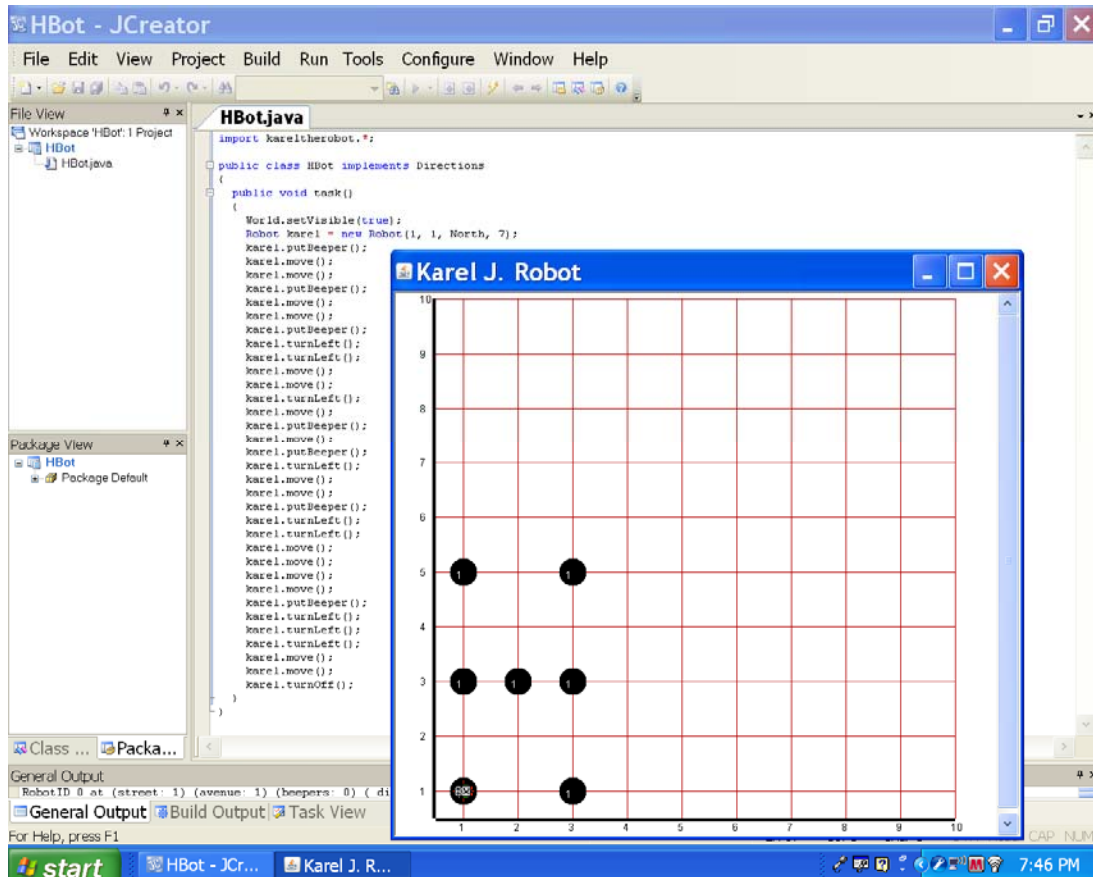


Figure 23: Lesson 12 Karel J. Robot Program Draw H

4.5.13 – Lesson 13

The students will have lab time to work on two Karel J. Robot assignments (Bergin, 1997). These two assignments require the students to copy a file from the Blackboard Web site into the “classes” folder so that the walls of the diagram display on the applet (see Figure 24). The first assignment is to write a program for the robot to retrieve a newspaper, which is a beeper, from outside his house. Then he is to return to the position from which he started.

The second assignment finds the robot with missing groceries from a ripped grocery bag. The students will write a program to help the robot retrace his steps and pick up the missing groceries (see Figure 25). There are four grocery items, which are beepers that the robot will retrieve.

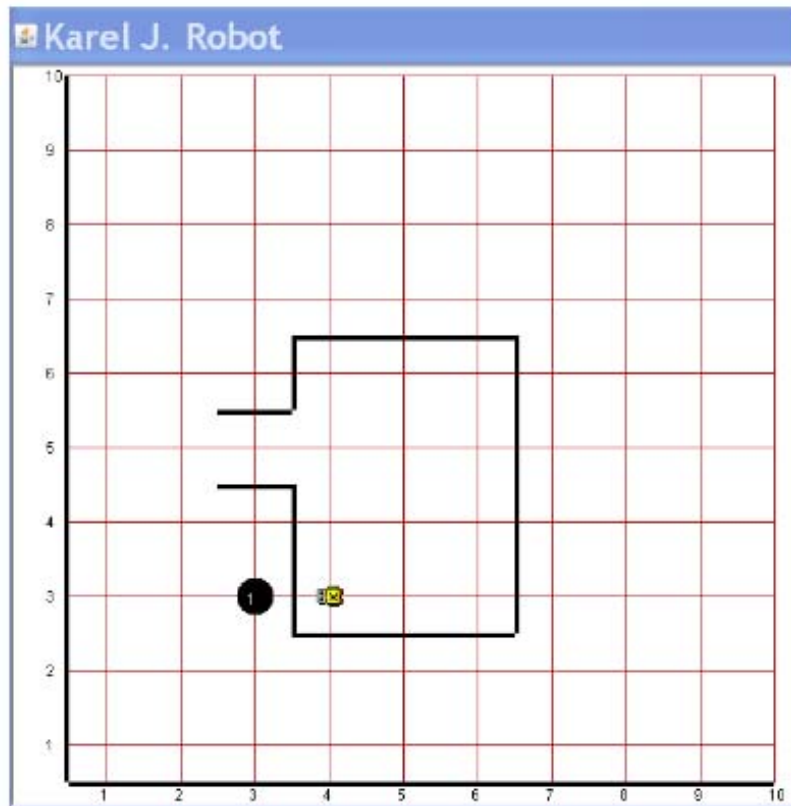


Figure 24: Lesson 13 Karel J. Robot Retrieve Newspaper

The students will notice how repetitive the code has become (see Figure 26) as they type the same instructions multiple times. This will introduce Chapter 3 of the Karel J. Robot textbook (Bergin, 1997) where the students will learn how to define new classes with new instructions that can be called with one code line instead of the repetitive code of Chapter 2.

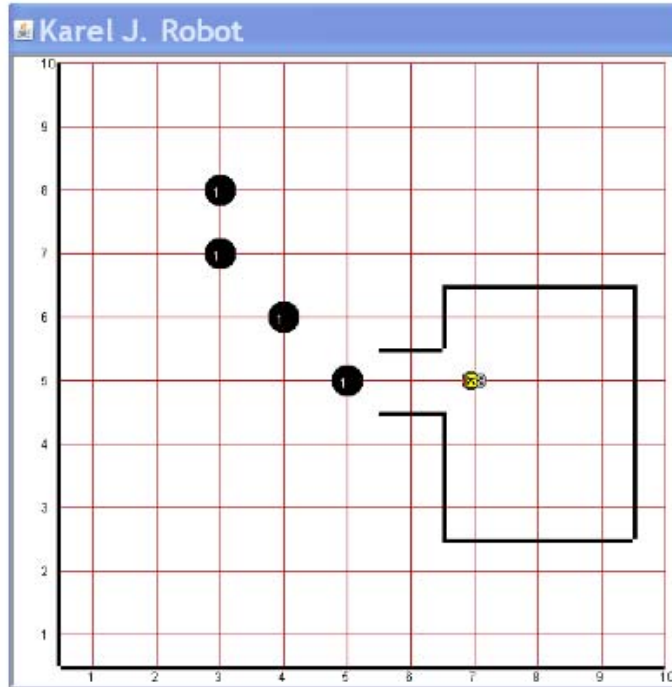


Figure 25: Lesson 13 Karel J. Robot Pick Up Groceries

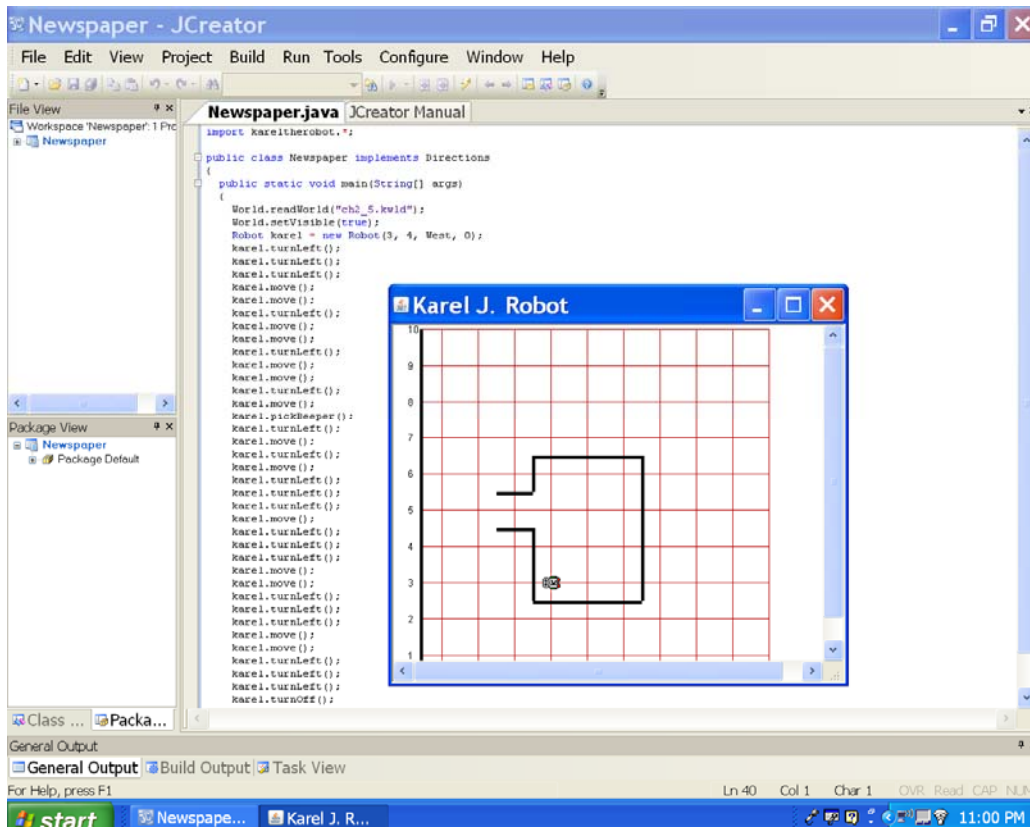


Figure 26: Karel J. Robot Repetitive Code

4.5.14 – Lesson 14

This last lesson of the unit will be continued practice for the students' programming skills. The first part of the lesson will begin with a review for the quiz the following day. With the remaining class time, the students will work on two more programs to write from the Karel J. Robot textbook. The first program has the robot climbing a mountain where he will place a beeper at the summit (see Figure 27). The students will guide the robot, through the program, along the side of the mountain. The robot cannot jump so he must travel close by the side of the mountain. This program gives the students practice placing beepers in a designated area.

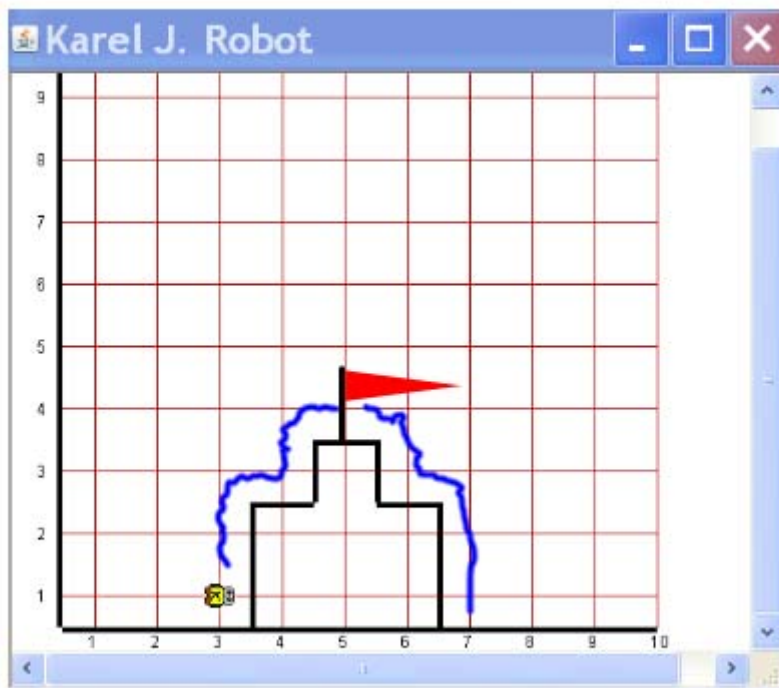


Figure 27: Lesson 14 Karel J. Robot Mountain Assignment

The second program is to walk the robot through a figure eight design around two beepers (see Figure 28). The challenge presented to the students is to write a program with as few instructions as possible. The objective of this assignment is to

begin the use of classes to be introduced in Chapter 3 of the Karel J. Robot textbook (Bergin, 1997).

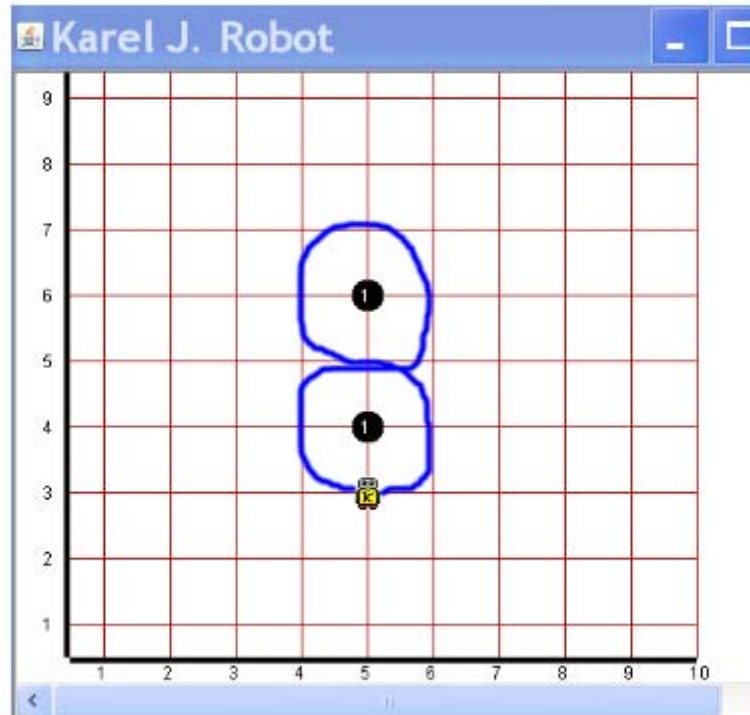


Figure 28: Lesson 14 Karel J. Robot Figure8 Assignment

The remaining lessons will be developed along a similar structure keeping in mind the objectives of the course. Key cognitive strategies will to be used to promote a high level of student learning. This can be accomplished through in-depth preparation of lesson plans, leading discussions with the students that promote students thinking on a higher level, and providing opportunity for exploration and experiencing achievement. Through balancing instruction, lab time, fun exercises and assignments that apply learned concepts to the “real world” situations and providing opportunities for communication of understanding through written and verbal forms, the lessons can instill a thorough understanding of the basic concepts, principles, and techniques of computer science.

4.6 – Assessments

Every Friday, with the exception of the first week of the course or when there is a unit test, a brief quiz will be administered to the students. Questions on the quiz will highlight the key concepts discussed through the week. The types of questions may be true or false, multiple choice or short answer. The quiz will take the students no more than half of the 51-minute class to complete.

Unit tests will take the entire class time. The types of questions may be multiple choice, short answer or long response. The concepts of the first unit are more about understanding the basics of object-oriented programming than about programming structure. Therefore, the first quiz and test will not be at as high a level of thinking as subsequent quizzes and tests through the course. The format will model the Advanced Placement exam in that there will be multiple choice questions and free response questions. The College Board has released some exam questions so that the instructor can write questions similar to those found on the AP exam. Test-taking strategies for each type of question will be discussed through the course.

The Institute of Computer Technology curriculum (Quesenberry, 2006), the textbook used for the Introduction to Computer Science course, also provides multiple forms of tests. The tests are electronic so that the instructor can make changes to better assess what has been discussed in the course for the unit. The answer keys are also provided.

The quiz for the first unit (see Appendix O) is shorter than the typical quiz since not as much content has been covered in the first two weeks of the course. (See Appendix P for the answer key). The questions are short answer format.

The unit test (see Appendix Q and Appendix R for test and answer key) has more content as more discussions and more programming practice has been completed at this time of the course. The questions are still not at the depth of those that will be seen on future unit tests in the course. The programming questions as released by College Board (2006) are questions that may be used on later tests. The following is a sample question from the released College Board questions (p. 24):

Consider the following code segment.

```
Int k = 1;
While (k < 20)
{
    If ((k % 3) == 1)
        System.out.print(k + " ");
    K++;
}
```

What is printed as a result of executing this code segment?

- (A) 2 5 8 11 14 17
- (B) 3 6 9 12 15 18
- (C) 1 4 7 10 13 16 19
- (D) 1 3 5 7 9 11 13 15 17 19
- (E) 2 4 6 8 10 12 14 16 18 20

Attending Advanced Placement conferences helps an instructor by facilitating the receiving and sharing of many aspects regarding the teaching of a computer

science course, such as writing and grading assessments. Being an active member of the Advanced Placement discussion board for computer science also facilitates knowledge in this area. With all of the resources available to the instructor including the textbook, College Board, and other educators, assessments can be written for the Introduction to Computer Science course that will be thought-provoking, promote critical thinking and adequately measure the students' achievement in the course.

4.7 – Grading Rubrics

Grading rubrics in computer science can be difficult and time-consuming for the instructor. Programming is the most important part of the computer science curriculum. Therefore, programming assignments are weighted 40% of the total grade (see Appendix A for assessment percentages). Educators have posted their programming rubrics on the College Board Web site to share with other AP Computer Science instructors. The rubrics range from the relatively simple and fast-grading format to the more detailed format.

The rubric favored for the Introduction to Computer Science course is a rubric with more details. Carter (2009) writes about this rubric from Sarah Fix of the Career Center in Winston-Salem, North Carolina. There is a total of 20 points in this rubric divided equally between four categories. The categories are correctness, design, style and documentation, and efficiency. Each category further subdivides into a zero to five rating based on what is to be graded with that particular category. For example, efficiency is graded a five for “the algorithm of choice used and correctly implemented” down to a zero for “grossly inefficient, unnecessary repetition of steps, etc.” Carter (2009) explains how the students submit their program as a MicroSoft

Word document. Then the grading is completed using Word's "Track Changes", the scoring is typed in at the end of the document and sent back to the student.

This grading rubric makes everything clear for the students. The students will understand what it takes to write good code by each of the categories and the subdivisions. The instructor's comments will help the students know how to make corrections and improve on the next program they will write.

There may be programs where a more condensed rubric is all that is necessary. The rubric can be created to fit the exact dynamics of the program. A program that has the students calculating the solving of a quadratic equation may be an 8-point rubric, with one point each for heading, prompts for input, correct calculations, one point for each of three outputs, correct layout of output, and use of the square root function and the power function. The instructor may adjust rubrics of assignments as deemed necessary.

Grading of the Advanced Placement Computer Science exam is often a topic at College Board workshops and conferences. The speaker teaches those that attend what to look for in a program and how to grade it by the Advanced Placement standards. The more the instructor of the Introduction to Computer Science course can do to help the students adjust to the format of the AP exam, the more comfortable the students will become with their skills by understanding what is required to score high on the rubric scale.

4.8 – Motivation and Building Confidence

Motivation can play a key part in a student's success in a course. Generally, students who elect to enroll in a high school computer science course, do so as their

choice. They may have an interest in computer science, believe that they will be good at computer science or suppose that there will be little work and a lot of fun. All of that can be true. But when the course challenges them at a level they were not expecting, the students can be disillusioned, lose confidence and therefore lose interest in the course. It is important for the instructor to take steps to avoid this from happening.

Motivation starts with the instructor believing that every student can be successful. Words of encouragement to the students such as “You can do it!” and “I will be there to help” and “I believe in you” can go a long way in steering students toward success. Supportive statements such as these and the actions taken by the instructor help the students feel that they are not alone in the struggles to get through some more difficult concepts.

In the classroom, the instructor encouraging students to ask questions without the risk of embarrassment can promote more students’ participation in discussions and consequently, feelings of accomplishments. Even an incorrect answer can be turned into a positive experience by using the answer as a catalyst for other ideas or different ways to approach the problem. Saying things such as “You are partially correct, which gives me a great idea” and “Who else has considered this viewpoint and would like to expand on that” can help a student see that the answer wasn’t entirely wrong. Rather, the answer may aide another into something they may not have considered before hearing the student’s response. This builds confidence in the students and helps them realize that every answer is a “good” answer.

It's important for the instructor to be aware of students who are in need of help. Students who are already invested in the course and love programming are also the students who are the first to seek assistance. It's very easy for an instructor to become surrounded by students who have questions but could probably answer those questions on their own or given some brief hints. The students who are quiet and unsure of their abilities are also the students who are not at the head of the line seeking the instructor's help. They are not quite sure how to ask for that assistance and do not want to ask in front of students for fear of feeling stupid.

Subtle ways of offering assistance to the quiet student can be employed so as not to make the student feel they are not capable. First, the instructor's routine while the students are working in the computer lab should be moving around to all areas of the computer lab. The instructor should not spend long periods of time with any student, but rather make brief observations or assistance with many students throughout the class time. Students become quickly aware of instructors favoring only the "smart kids". With this process, it won't be as obvious when the instructor visits a student at his or her computer that truly is in need of assistance. Then the instructor can offer suggestions of "try this and see what happens" and "I'll come back to see what you have done with this" shows the student the instructor has confidence in them. And they'll feel great knowing that they've achieved positive results with only a little help. For a student, it is so gratifying to see that program run successfully on that computer screen. In what other course can a student achieve instant gratification after completing an assignment?

A group of participants from an Advanced Placement Computer Science summer institute posted tips on AP Central (2004) for teaching success in the computer science classroom. The suggestions are very helpful for an instructor to motivate the students and help the students view computer science as interesting and fun. One suggestion is to never touch the student's mouse. Rather, let the student be in the driver's seat. Most students are visual learners and letting them physically work through the process will not only help them understand the material better, but also help them remember it better. Secondly, lecture as little as possible and let the students practice hands-on computer skills. Another suggestion is to set up the classroom using the golden rule of programming. With one instructor in a computer lab of 25 students, it is difficult to be all places at all times. The students need to learn to help each other. Therefore, the golden rule is to ask another student first before seeking the help of the instructor. These are just a sample of ten suggestions posted on AP Central (2004).

Enrollment of girls in computer science courses is a problem at the high school level and at the college level. And thus, there are few women employed in the computer science industry. The participants at the AP institute (AP Central, 2004) suggested a very simple strategy to help girls feel like they belong in the computer science classroom. Their suggestion was to not refer to the class as "you guys". Using this phrase makes the girls feel like a computer science course is only meant for boys. Treating the girls equally by avoiding phrases such as this and assisting all students equally will help the girls feel like they are on an even level with others in the course.

Building motivation and confidence in students is something every instructor has to deal with in every discipline. Since the enrollment in the school's computer science courses have dropped in the last few years, there is a particular need to build motivation and confidence in the Introduction to Computer Science course to keep the students interested and to encourage them to enroll in the Intermediate Computer Science course, the next computer science course in the sequence at the instructor's school. The above strategies will ensure that the students will experience success and feel confident in their abilities to continue on to the next computer science course.

Chapter 5 – Project History

This project began as a result of the need to better prepare the students in the computer science program at the instructor's school for the Advanced Placement (AP) Computer Science Course and the exam administered each May. In addition, the need to increase enrollment in these courses was a factor in developing this curriculum. For a school the size of this school (approximately 3,500 students) and for a school that places a strong focus on the Advanced Placement curriculum, there should be a stronger interest from the students to enroll in the computer science courses. The need for a new curriculum for the Introduction to Computer Science Course was established.

5.1 – Goal Evaluation

The project will be evaluated by the following criteria:

1. All proposals of teaching methods will be supported through research.
2. Lesson plans and assignments align with course objectives, particularly the course objectives of the Advanced Placement Computer Science curriculum.
3. The number of students taking computer science courses has increased. (This will not be available within the scope of this project. Interest in course work normally would take at least two years to build.)
4. The percentage of students receiving a score of 4 or higher on the Advanced Placement Computer Science exam has increased. (This is beyond the scope of this project and cannot be determined.)

Chapter 6 – Lessons Learned and Next Evolution of the Project

The development of a curriculum is a long and arduous process. It is also one with great rewards along the way. This project is only one small step, albeit a very important step, in the beginning of that development process. The students' experiences, reactions, learning styles, and progress will all play a part in predicting the next direction of this project.

6.1 - What I Learned From the Project Experience

I have learned many things from this project experience. Drawing on my prior experience of developing curriculum for high school mathematics courses and putting that together with what I have learned in computer science design in my courses at Regis University, I have merged the two together to analyze what is best for the students enrolling in the Introduction to Computer Science course at the instructor's school. I have learned how to research and design a course that best meets the needs of the students, that there is a multitude of resources available for educators to assist them in this process, and that analysis of the current situation at hand is important in discerning what will work for the instructor, the students and the learning environment. What I have learned can be applied to the development of other mathematics courses and can assist me in aiding new teachers through this process.

6.2 – What I Would Have Done Differently

One of the processes I would have done differently is to have observed other high school computer science courses. For this project, I am not actually teaching the

course nor are there plans in the future for me to teach the course. Therefore, it was not feasible for release time in the current school day to view computer science courses at other schools. Release time would be granted, however, if teaching a course such as this for the first time. Additional conversations with those educators in other schools would also be beneficial to hear what they have found to be successful in meeting the needs of their students.

Also of benefit would be attending an Advanced Placement conference. These conferences are taught throughout the country several times during the school year. Having attended these for the Advanced Placement Calculus BC course, I have found them to be extremely valuable in providing insight into a variety of areas such as teaching methodologies, assignments that help the students understand the material, assessments, and grading.

6.3 – Did the Project Meet Initial Expectations?

The initial expectations of the curriculum being developed and supported through research were met. Because of the research completed, the curriculum developed has a strong foundation on which to begin the course semester. The lesson plans, assignments, and assessments have been constructed with the understanding of the students' needs and the objectives for the course. Many tools are in place to build confidence in the students. The textbook to be used is easy for the students to read and understand. The Integrated Development Environment and other software, such as Karel J. Robot (Bergin, 2007), to be used in the course are designed for the beginning computer programming student. With increased confidence within the students, their interest in the computer science will increase as well. The instructor's

knowledge of key cognitive strategies will aide in asking the students the appropriate questions which will lead the students in thinking on a higher level. With these items firmly in place, the instructor can begin teaching the course with confidence that the students will be provided a sound learning environment.

6.4 – What Would Be the Next Stage Of Evolution for the Project If Continued?

The next stage of evolution for the project would begin on the first day of the course. The instructor would be observing how the students comprehend the material being taught. Conversations would continue between the instructor and his or her students to give the students the opportunity to express their thoughts on how the learning environment is working for them. Adjustments can be made through the first unit. These adjustments may include spending more time on a particular content or perhaps spending less time if the students feel things are moving along too slow. The instructor or students may feel additional explanation or practice may be warranted and change the number of days spent on a particular content area. More importantly, the time spent covering the first unit would also indicate how the subsequent units will need to be written. The instructor will be analyzing assignments and assessments to see if the students are mastering the content.

If time permits during the semester, the instructor may be able to veer away from curriculum that is not included in the capstone course, Advanced Placement Computer Science, and develop units that have a computer science base but not required in the AP curriculum. These items may include game development, systems analysis and design, or web design. Items such as these may help the students find computer science more enjoyable and easier to understand.

At the end of the first unit, the instructor may develop a survey utilizing Google docs, an account that the instructor's school has in place for all instructors, where the students could respond to questions concerning the results of learning that has taken place through the first unit's curriculum. Some questions that could be asked are the following: Did you feel you performed as well as you expected in this past unit? Why or why not? Did the instructor spend an adequate amount of time on the material? Is the time too little or too much? Would more individual assistance from the instructor be helpful? Would more lab exercises, role play activities, programming assignments be helpful? The students could complete the survey privately at their computers in the computer lab or at home. The instructor could then quickly and efficiently compile the results and make adjustments to the next unit accordingly.

When one full semester of the course has been taught, all worksheets, notes, including reading material, instructions and assignments will be compiled into a notebook. The printing service for the district will bind these with a cover to be sold to the students of future courses through the school store. All of the instructor's notes, lesson plans, PowerPoint slides, assessments and answer keys will be organized into a three-ring binder for the instructor's use. After each year this course is taught, the student and instructor's books will be updated.

6.5 – Recommendations

One recommendation is for the instructor to always “try things out” first, if at all possible. If there is a decision to be made on software, take some time to use it on the computer before using it with the students. It may not be what the instructor had

in mind. It is better to find this out before purchasing the software or having the technology department load it onto all of the computers and then finding out it does not support the instructor's objectives.

Secondly, the instructor should always work through programs before assigning them to the students. This is invaluable in finding problems that may arise during instruction or questions or problems that the students may run into while doing the programming themselves. As an added bonus, it is a great way for the instructor to become more knowledgeable about programming, too!

The most important recommendations for the instructor is to make use of the educational opportunities and support available for computer science instructors. The amount of material and support can be overwhelming, at times, and often confusing. There is so much information that it becomes difficult to discern what is best for the instructor and the instructor's students. But an instructor should never stop learning and reevaluating what they do. Reexamining the curriculum and changing it is a vital way to keep things fresh, exciting and up-to-date to fit the needs of the students.

The educational community believes very strongly in professional learning communities (PLCs). Some schools have built time into the weekly schedule for educators to meet and discuss ways to meet students' needs. Even if time is not built in by the school, it is important to make this time. This can include attending classes or conferences, researching through books, magazines or the internet, visiting other schools or participating in online discussion boards. All ideas from other sources are valuable to research and consider. They are from experienced educators that have, at one time, been new to the computer science education field, and are willing to share

what they have learned. Do not be afraid to try something new. If it doesn't work for your students, then change it. Be creative and believe in what you do and the students will respond. Remember, that if an instructor enjoys and is excited about what is taught, then the students will enjoy the class, be excited and learn as a result.

6.6 – Summary

The purpose of this project was to develop curriculum for the Introduction to Computer Science course at the instructor's school. The curriculum that would be implemented, should the instructor ever be asked to teach the course, would prepare the students for the Advanced Placement Computer Science course and would instill confidence in the students that they have the ability to complete said course successfully.

References

- AP Central. *AP Computer Science AB Syllabus 3*. Retrieved October 28, 2008, from http://apcentral.collegeboard.com/apc/public/repository/Comp_Sci_AB_Syllabus_3.pdf
- AP Central. (2004). *Ten Tips for Teaching Success*. Retrieved February 17, 2009, from http://apcentral.collegeboard.com/apc/members/courses/teachers_corner/45470.html
- Barnes, D., & Kolling, M. (2003). *Objects First With Java, A Practical Introduction Using BlueJ*. London: Pearson Education Limited.
- Barth, R., DuFour, R., Eaker, R., Eason-Watkins, B., Fullan, M., Lezotte, L., Reeves, D., Saphier, J., Schmoker, M., Sparks, D., & Stiggins, R. (2005). *On Common Ground*. Bloomington, IN: National Educational Service.
- Bergin, J., Stehlik, M., Roberts, J., & Pattis, R. (1997). *Karel++, A Gentle Introduction to the Art of Object-Oriented Programming*. New York: John Wiley & Sons, Inc.
- BlueJ – The interactive java environment[computer software]. (2009). Retrieved February 1, 2009, from <http://www.bluej.org/index.html>
- Carter, D. (2009). *Program Assignment Grading*. Retrieved February 17, 2009, from http://apcentral.collegeboard.com/apc/members/courses/teachers_corner/45331.html
- Ciezki, R. (2008). *Computer Science AB Course Perspective*. Retrieved October 28, 2008, from <http://apcentral.collegeboard.com/apc/public/courses/descriptions/4346.html>
- College Board. (2009). *Computer Science A and Computer Science AB Course Descriptions*. Retrieved January 19, 2009, from http://apcentral.collegeboard.com/apc/public/repository/ap09_compsci_course_desc.pdf
- College Board. (2006). *The 2004 AP Computer Science A and Computer Science AB Released Exams*. New York: College Board.
- Conley, D. T. (2005). *College Knowledge*. San Francisco, CA: Jossey-Bass.
- Conley, D. T. (2007, March). *Toward a More Comprehensive Conception of College Readiness*. Eugene, OR: Educational Policy Improvement Center.

- Cross, J. (2008, June). *jGRASP Update New Technology*. Retrieved January 21, 2009, from <http://apcentral.collegeboard.com/apc/Pageflows/TeachersResource/viewResourceDetail.do?source=tr&resourceId=4486>
- Evan, A., Huberman, M., Means, B., Mitchell, K., Shear, L. & Shkolnik, J., et al. (2006, August). *Evaluation of the Bill & Melinda Gates Foundation's High School Grants Initiative, Bill & Melinda Gates Foundation*. Retrieved January 6, 2009, from <http://www.gatesfoundation.org/learning/Documents/Year4EvaluationAIRSRI.pdf>
- Friesen, J. (2005, August). Java Tech: The Sweet Song of the BlueJ, Part2. Retrieved January 20, 2009, from <http://today.java.net/pub/a/today/2005/08/30/bluej.html>
- Levine, D. (2003). Role Playing in an Object-Oriented World. Retrieved February 16, 2009, from <http://www.cs.sbu.edu/dlevine/RolePlay/roleplay.html>
- Lewis, J., Loftus, W., & Cocking, C. (2004). *Java Software Solutions for AP™ Computer Science*. Boston: Addison-Wesley.
- Litvin, M., & Litvin, G. (2001). *Java Methods, An Introduction to Object-Oriented Programming*. Andover: Skylight Publishing.
- Litvin, M., & Litvin, G. (2003). *How to Use JCreator LE*. Retrieved February 21, 2009, from <http://www.skylit.com/javamethods/faqs/jcreator4.html>
- North, K. (2008). *The "Write" Tool for Introductory Computer Science Courses*. Retrieved October 28, 2008, from http://apcentral.collegeboard.com/apc/members/courses/teachers_corner/22778.html
- Quesenberry, N. (2006). *Java Curriculum for Advanced Placement™ Computer Science, Version 2.0 Revision*. Institute of Computer Technology (ICT).
- Rubinkam, M. (2005, August 9). Students charged with computer trespass. *USA Today*, Retrieved February 19, 2009, from http://www.usatoday.com/tech/news/2005-08-09-kutztown-hackers_x.htm
- Slater, D. (2008). *Karel J. Robot UPDATE NEEDED – NEW RELEASE*. Retrieved February 8, 2009, from <http://apcentral.collegeboard.com/apc/Pageflows/TeachersResource/viewResourceDetail.do?source=tr&resourceId=4026>

- Taft, D. (2007). *Girls Ask Alice for Programming Skills*. Retrieved February 8, 2009, from <http://www.eweek.com/c/a/Application-Development/Girls-Ask-Alice-for-Programming-Skills/>
- Trees, F. (2009). *Sharing Strategies for Teaching AP CS*. Retrieved January 19, 2009, from the AP Central Web site:
http://apcentral.collegeboard.com/apc/members/courses/teachers_corner/45322.html
- Trees, F., Cutler, R., Wittry, D., Kmoch, J., Hromcik, J., Litvin, M., Nevison, C., Astrachan, O., Carter, D. (2005). *The Teaching Series: Special Focus in Computer Science, Object-Oriented Design*. College Board.
- Xinox Software. *JCreator*[computer software]. Retrieved February 15, 2009, from <http://www.jcreator.com/>

Appendix A

(NAME OF SCHOOL)
INTRODUCTION TO COMPUTER SCIENCE
(INSTRUCTOR'S NAME)

COURSE OBJECTIVES, GRADING AND CLASSROOM EXPECTATIONS

OFFICE: (Room number)
OFFICE PHONE #: (phone number)
OFFICE HOURS: (off periods)
e-mail: (instructor's email address)
Blackboard Web Site: (web address)
Powerschool Login: (web address)

COURSE DESCRIPTION: *Introduction to Computer Science* is a one-semester course that serves as an introduction to Computer Science to students of all grades. This course covers programming techniques using the java language and the Object Oriented Programming paradigm. Topics include classes, object-oriented programming, primitive data types, object behavior, libraries and APIs, simple input and output, loops, decisions, recursion and Strings. Students will be introduced to the Karel J. Robot class written by Joseph Bergin and use it to solve problems throughout the semester. Techniques of problem solving will be stressed heavily as well as ethics as they relate to technology.

All topics will be practiced in the programming language java. This course is taught in a computer lab where each student works on an individual PC computer using JCreator. This course meets 5 days a week for 51 minutes each day.

The prerequisites are CP Algebra I or Integrated I.

MATERIALS:

All materials are available on the Blackboard Web site listed above. An optional purchase of the workbook is available in the school store for \$4.00.

TEXT: Quesenberry, Nancy. *Java Curriculum for Advanced Placement (TM) Computer Science*, version 2.0 revision. Institute of Computer Technology (ICT), 2006.

ASSESSMENT: Grades are made up of programming assignments (40%), quizzes (25%) a final exam (20%) and other activities (15%). Other activities include relevant discussions on Blackboard, worksheets, research and role-playing. Quizzes include multiple-choice questions and free response questions that are in the AP format.

The points from all these items will be averaged together and the following grading scale will be used: 90-100% (A), 80-89% (B), 70-79% (C), 60-69% (D), 0-59% (F)

TEACHING METHODS: This course will be taught to promote the key cognitive strategies needed to prepare students for college-level work. These strategies used to promote a high level of student learning include: a thorough understanding of the basic concepts, principles, and techniques of computer science, conceptual understanding of computer science concepts, applying learned concepts to “real world” situations, efficient problem solving strategies, and evaluating the validity of solutions.

Students will learn through lectures, demonstrations, discussions, programming assignments, quizzes, exams and other activities. The students have a quiz every Friday on any previous material in either the multiple-choice format or the free-response format of the AP exams.

CLASSROOM BEHAVIORS & EXPECTATIONS:

1. Bring all needed material to class. Textbook, paper, pencils and notebook are to be brought to class everyday.
2. Be in your seat and ready to work when the bell rings and remain in class the full period.
3. Obtain permission before speaking or leaving your seat.
4. **Respect...**
 - yourself.** Always do your best, believe in yourself, and do your own work.
 - your classmates.** Listen to, appreciate the differences in, and support each other.
 - your teachers.** We have chosen this profession because of our passion to help you.
5. Seek extra help when needed and take responsibility for material missed.
6. Listen carefully when instructions or presentations are being given, taking good notes.
7. Respect the property of other people and the school.
8. Ask questions and be an active participant in the day's activity.
9. Exercise integrity. I expect that you will always function honestly and with integrity on every test and every assignment, as well as with every person in the class. Every time you take a quiz or a test you are taking two assessments. The first has to do with the mathematics; the second has to do with your character. If you can only pass one of them, make it the second.

HOMEWORK:

Homework will be assigned daily. Homework is due the next class period after it is assigned unless otherwise stated.

*Answers only are not acceptable. **SHOW ALL OF YOUR WORK.***

It is VERY important to consistently do your homework. Always come to class with any questions from each assignment. We will go over questions after presentation of new material.

ABSENCES AND LATE WORK:

For every excused absence, you will be allowed two days to complete any make-up work. It is the student's responsibility, not the teacher's, to find out what was missed during the absence. A student can do this by asking another student in class, by looking at the assignment sheet or by checking the web site.

If you miss the day before a test or quiz, be prepared to take the exam as scheduled. If you miss the day of the test, be prepared to take the test the day you return to school. If you missed several days before a test, you will need to come in during a free period to make it up; this will need to be done within one week of the absences.

If you have not been absent but failed to turn in an assignment, late work will be accepted one week past the due date for **half** credit. Any work not made up will be graded as a 'zero'. If you have a pre-arranged absence, you need to see me before you are gone and the work that I give you will be due when you return. This includes all tests, quizzes, programming problems and homework.

GETTING HELP:

My off periods are listed on the front. Come see me if you need help or if you have any questions. Don't hesitate to ask your peers for help. Please don't wait until you're lost before you come for help.

Teacher's Signature: _____

Appendix B

Student and Parent Signature Confirming Understanding of Course Objectives

<p style="text-align: center;">(NAME OF SCHOOL) INTRODUCTION TO COMPUTER SCIENCE, (INSTRUCTOR'S NAME) COURSE OBJECTIVES, GRADING AND CLASSROOM EXPECTATIONS</p> <p>I have read and understand the course objectives, grading and expectations for Introduction to Computer Science.</p> <p>Student's printed name: _____</p> <p>Parent's Signature: _____</p> <p>Student's Signature: _____</p>
--

Appendix C

Downloading the SUN Java JDK (Java Development Kit) and Java API

Documentation

1. Go to <http://java.sun.com> and find the download page.
2. Click on “Java SE Development Kit (JDK) 6 Update 12” (this may change over time).
3. Select the correct download of Java SE for your machine. For Windows, select the Windows (all language) download of JDK.
4. Click on Continue. Click on the file name. Click RUN. Follow installation instructions.
5. By default, it should be in “Java” within “Program Files” on your C: drive.
6. Go to <http://java.sun.com/javase/downloads/index.jsp> and download the Java SE 6 Documentation.
7. Select English and the agree box. Click Continue.
8. Click on the file name as was done in Step 4 above. Click Save.
9. Save in C:\Program Files\Java\jdk1.6.0_12.

Appendix D

Downloading JCreator, the IDE (Integrated Development Environment)

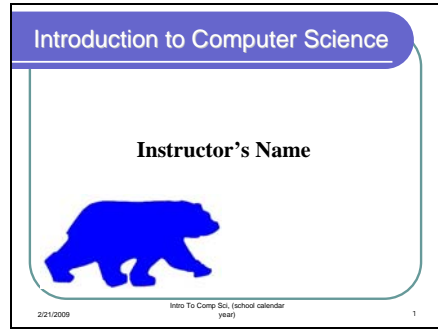
1. Go to <http://www.jcreator.com/download.htm> and go to the download page.
2. Click on Download. There are installation instructions on the JCreator web site.

Use these instructions to configure default properties. Save these instructions for help in later setting up workspaces and projects. When JCreator is launched, follow the Setup wizard and set the paths to C:\...\jdk.1.6.0_12\ and C:\...\jdk1.6.0_12\docs.
3. Go to the Blackboard Web site. Click on the tab marked “Course Notes”. Click on the link for “Installation Directions”. Right click on the jar file and select “Save Target As”. Save to a folder on the computer or save to the desktop.
4. In order to use libraries (packages) in JCreator, open a workspace with a project. Go to Project/Project Settings and click on the Required Libraries tab. To add a library, click New, click on the Classes tab and enter a name of your choice for the set of jars in the dialog box. Click on Add and choose Add Archive. Browse to the location that you saved the jar files in Step 3. Double click on the file until the file or folder is in the list. Repeat until all of the jar files from Blackboard are in the list. Check the box next to the folder name to make it available in any project.

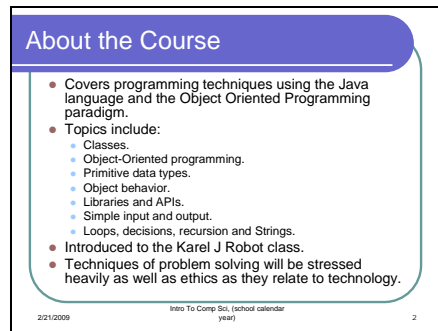
Appendix E

Lesson 1- Course Objectives and Classroom Expectations PowerPoint

Slide 1



Slide 2



Slide 3

Teaching Methods

- This course will be taught using these strategies:
 - A thorough understanding of the basic concepts, principles, and techniques of computer science.
 - Conceptual understanding of computer science concepts.
 - Applying learned concepts to "real world" situations.
 - Efficient problem solving strategies.
 - Evaluating the validity of solutions.

2/21/2009 Intro To Comp Sci, (school calendar year) 3

Slide 4

Teaching Methods (cont.)

- Students will learn through:
 - Lectures.
 - Demonstrations.
 - Discussions.
 - Programming assignments.
 - Quizzes.
 - Exams and other activities.

2/21/2009 Intro To Comp Sci, (school calendar year) 4

Slide 5

Materials Needed

- All materials are available on the blackboard website.
- Optional: Purchase Notes/Worksheets notebook from school store - \$4.00
- 3 ring binder/docket/folders.
- Optional: Install Java, JCreator and Karel J Robot on home computer
 - Alternative: Use off-periods to complete homework in computer lab.

2/21/2009 Intro To Comp Sci, (school calendar year) 5

Slide 6

Assessments

- Grades are made up of:
 - Programming assignments (40%)
 - Quizzes (25%)
 - Quizzes include multiple-choice questions and free response questions that are in the AP format.
 - Final exam (20%)
 - Other activities (15%). Other activities include:
 - Relevant discussions on Blackboard,
 - Worksheets,
 - Research and
 - Role-playing.

2/21/2009 Intro To Comp Sci, (school calendar year) 6

Slide 7

Grading

- The points from all these items will be averaged together and the following grading scale will be used:
 - 90-100% (A)
 - 80-89% (B)
 - 70-79% (C)
 - 60-69% (D)
 - 0-59% (F)

2/21/2009 Intro To Comp Sci, (school calendar year) 7

Slide 8

Homework Assignments

- Assigned daily, including Friday.
- Some assignments will be handed in on paper.
- Some assignments will be submitted electronically through Blackboard (e.g. discussion posts).

2/21/2009 Intro To Comp Sci, (school calendar year) 8

Slide 9

Homework Assignments (cont.)

- Homework sheet will be handed out at beginning of each quarter.
- Also, available on Blackboard and Bulletin Board in the lab.

2/21/2009 Intro To Comp Sci, (school calendar year) 9

Slide 10

Classroom Behaviors & Expectations

- Bring all needed material to class. Textbook, paper, pencils and notebook are to be brought to class everyday.
- Be in your seat and ready to work when the bell rings and remain in class the full period.
- Obtain permission before speaking or leaving your seat.

2/21/2009 Intro To Comp Sci, (school calendar year) 10

Slide 11

Classroom Behaviors & Expectations (cont.)

- **Respect...**
 - **yourself.** Always do your best, believe in yourself, and do your own work.
 - **your classmates.** Listen to, appreciate the differences in, and support each other.
 - **your teachers.** We have chosen this profession because of our passion to help you.

2/21/2009 Intro To Comp Sci, (school calendar year) 11

Slide 12

Classroom Behaviors & Expectations (cont.)

- Seek extra help when needed and take responsibility for material missed.
- Listen carefully when instructions or presentations are being given, taking good notes.
- Respect the property of other people and the school.
- Ask questions and be an active participant in the day's activity.
- Exercise integrity.

2/21/2009 Intro To Comp Sci (school calendar year) 12

Slide 13

Absences

- Excused Absences.
 - It is your responsibility to find out what you missed.
 - Two days for every day absence to make up homework.

2/21/2009 Intro To Comp Sci (school calendar year) 13

Slide 14

Absences (cont.)

- Excused Absences.
 - If you miss the day before a test, be prepared to take the exam as scheduled.
 - If you miss the day of the test, be prepared to take the test the day you return to school.
 - If you missed several days before a test, there is one week to make up the test. (Make-up test are a little harder).

2/21/2009 Intro To Comp Sci (school calendar year) 14

Slide 15

Absences (cont.)

- **Unexcused Absences.**
 - Assignments, tests and quizzes are counted as a zero.
 - 1st Unexcused Absence – conference with you.
 - 2nd Unexcused Absence – parent is contacted and referral to dean.

2/21/2009 Intro To Comp Sci, (school calendar year) 15

Slide 16

Homework for Tonight

- Bring back Signed Expectation Form (worth 10 homework points).
- Log on to Blackboard from home and "tour" site.

2/21/2009 Intro To Comp Sci, (school calendar year) 16

Slide 17

Close

"There's little you can do with a can't do" attitude, and everything you can do with a "can do" frame of mind. Just scratch the surface of every successful enterprise and you'll see a whole collection of "cans."

Todd Siler, author of "Think Like a Genius"

2/21/2009 Intro To Comp Sci, (school calendar year) 17

Appendix F

Login Instructions for Blackboard Web Site

To login to Blackboard:

1. Type in the following web address: (web address) or connect via the school's website.
2. Click on Login.
3. Put in your User ID and password and click on login.

To change your password:

4. On the left, under the Tools menu, click on "Change Password and Settings".

To enroll in Introduction to Computer Science:

5. Next you need to click on the red tab at the top of the page that says, "Classes".
6. Click on "XXHS Math" from the list on the right side.
7. Find the "Introduction to Computer Science" class taught by your instructor. **Do not click on the name of the class! Look to the right side and click on "Enroll".**
8. Then select "Submit."
9. Then click "OK."

Take some time to look through the various parts of the course:

- ❖ Announcements: Daily/ Weekly/ Monthly updates via BlackBoard
- ❖ Teacher: Information about the teacher of this course
- ❖ Class Information: syllabus & course information
- ❖ Homework Calendar: calendar for each chapter's assignments
- ❖ Worksheets: copies of worksheets
- ❖ Course Notes: Detailed explanation of the course content
- ❖ Communication: email and discussion board
- ❖ External Links: links to helpful sites
- ❖ Atomic Learning: free district online resource for software tutorials

Appendix G

Lesson Plan for Day 4

Summary of AI of the ICT Textbook (Quesenberry, 2006)

Slide 1

Lesson 4

- Objective: To introduce students to the definitions of objects, classes and methods.
- To give students a feel for object-oriented programming.

Slide 2

Object-oriented programming (OOP)

- Makes programs more closely model the way people think about and deal with the world.
- Program consists of a collection of interacting objects.

Slide 3

Definition of object

- Can be described in terms of its attributes and behaviors.
- Examples from role play:
 - acrobat
 - blackboard
 - calculator

Slide 4

Definition of object (continued)

- An attribute is anything that describes an object.
 - For example, a calculator can be described by the keys it has.
- A behavior is anything an object does.
 - For example, an Acrobat claps, does kneeBends and counts.

Slide 5

Definition of object (continued)

- Instance - reference of a particular object.
 - For example: bob was an acrobat, charlie was a blackboard and sue was a calculator.
- There can be more than one instance of an object.
 - Recall we had two acrobats in role play.

Slide 6

Encapsulation

- Each object protects and manages its own information.

Slide 7

Definition of a class

- Provides definition of the objects.
- Describes:
 - How the object behaves.
 - What kind of information it contains.
 - How to create objects.
- Defines methods of the objects.

Slide 8

Inheritance

- Classes created from other classes use inheritance.
- Definition of one class is based on another class.
- Form of software reuse.

Slide 9

Definition of methods

- Methods define the operations that can be performed on objects.
- We invoke or call a method.
- Object can only receive a message defined within its class.
 - Remember when you were asked to do things not defined in role play.

Slide 10

Definition of methods (continued)

- Example of method:
`System.out.println("Hello, World!");`
- A call to the `println` method of the `System.out` object built into Java.
- Prints what appears between quotation marks.

Appendix H

"Currents in Technology" Guidelines

From N. Quesenberry (personal communication, January 30, 2009)

Purpose: The purpose of these assignments is twofold: First, it is important to be aware of what is happening today in the field of technology. What are the trends, both personally and professionally? What is behind or driving these trends? Who are the "pace setters" and where are they going, what have they done or planning to do? Why? Where? Secondly, it is essential to be able to put your thoughts down on paper – to be able to communicate, no matter what field you are in, via the written word.

Thus, over the course of this semester, I will from time-to-time assign either an article for you to read or have you find an article which discusses a current topic in technology. These assignments will be completed outside of class time.

Each "Currents in Technology" will have a point value of 15 points and will be graded on content (5 points), readability/structure (5 points), and spelling/grammar (5 points). The article which you wrote about also needs to be attached to your paper.

Format: Use the following format to guide your responses:
Name, Currents in Technology #1, 2, 3...., Title of your article, Author, Source, Date of source. **All assignments should be word processed (12 pt. type), length of one page.**

Structure:

- Introductory paragraph – *Tell us what you are going to tell us.*
- Body paragraphs – *Tell us.*
- Summary or concluding paragraph – *Tell us what you told us.*

Use complete sentences and quotations where necessary – *So and So said "....."*

Content: Try and address the following topics in your response but don't limit yourself:

Who, What, When, Why, How, and Where.

Appendix I

Lesson Plan for Day 5

Slide 1

Lesson 5

- To introduce some java coding syntax; i.e. comments, class definition, main method and print method.
- To introduce JCreator with introductory program "Hello, World!"

Slide 2

Java program

- A collection of instructions that performs a particular task on a computer.
- The purpose is to solve a problem.

Slide 3

Documentation

- Any written comments or documents.
- Comments:
 - Start line with // symbols.
 - Ignores text after // to end of line.
 - Comments between /* and */ are ignored and can run for multiple lines.
 - Don't affect what program does.
 - Helps anyone reading the code understand what that code will do.

Slide 4

Class definition

- Runs from first opening brace ({) to final closing brace (}).
 - Provides a name for the class.
- Start names of classes with capital letters.

Slide 5

Main Method

- Always preceded by the words public, static, and void.
- This is where processing begins.
- Implements the behavior of the objects.

Slide 6

JCreator

- Click File, New, Project, Empty Project and Next.
- Type the name "Wizard".
- Be sure the JDK box is checked. Click Next.
- Default should be checked. Click Finish.

Slide 7

JCreator (continued)

- Click File, Class.
- Type Wizard in the name display box.
- Check Public and Generate main method.
- Click Finish.
- Type in code.
- Build and Run.

Appendix J

Lesson Plan for Day 6

Slide 1

Lesson 6

- Objective: to connect what the students did in role play to Java programming and code.

Slide 2

Role Play Review

- What actions were the acrobats asked to do?

Slide 3

Answer

- clap a specific number of times.
- kneeBend a specific number of times.
- count the number of exercises performed.

Slide 4

Acrobat class

- These are the rules for the methods.
- These are behaviors that the object can perform:
acrobat()
void clap(int i)
// clapped i times

Slide 5

Acrobat class (continued)

- Methods (continued).
void kneeBend(int i)
// i kneeBends
int count()
/* returns the number of exercises performed */
- Don't worry about understanding the details of the code – they will be covered more later.

Slide 6

Creating an Acrobat Object

- This defines bob as an Acrobat.

```
Acrobat bob = new Acrobat ( );
```

Slide 7

Methods for Acrobat

- This is how we tell bob to:

```
// clap 3 times  
bob.clap(3)  
  
// Do 5 kneeBends  
bob.kneeBend(5)  
  
// count  
bob.count( )
```

Slide 8

Blackboard class

- What actions was a blackboard asked to do?
- How would you give the rules for a blackboard?

Slide 9

```
Answer

Blackboard( )
  /* This is the special method with exactly the
  same name as class constructor */
void drawSquare( )
  // Object draws square in designated space
void drawCircle( )
  // Object draws circle in designated space
```

Slide 10

```
Answer for Blackboard (cont.)

void drawText(String s)
  // Object draws text in designated
  space
void clear( )
  // Clears designated space
```

Slide 11

```
Define a Blackboard Object

Blackboard charlie = new Blackboard( );
  /* This creates an object called charlie
  of type Blackboard */
```

Slide 12

Blackboard Methods

- How would we tell charlie to perform certain functions?

Slide 13

Answers (Blackboard Methods)

```
charlie.drawSquare( )  
charlie.drawCircle( )  
charlie.drawText("Hello")  
charlie.clear( )
```

Slide 14

Calculator class

- What actions was a calculator asked to do?
- How would you write a special method with exactly the same name as the class constructor?

Slide 15

```
Answer  
Calculator( )  
    /* Method that allows us to create  
    an object of type calculator */  
int add(int a, int b)  
    /* Given two pieces of information,  
    returns the sum of a and b */
```

Slide 16

```
Answer (continued)  
  
int subtract(int a, int b)  
    /* Returns the difference (a  
    minus b) */  
  
int multiply(int a, int b)  
    // Returns the product of a and b
```

Slide 17

```
Define a Calculator Object  
  
Calculator sue = new Calculator( );  
    // This defines sue as a Calculator
```

Slide 18

Calculator Methods

- How would we tell sue to perform certain actions?

Slide 19

Answers (Calculator Methods)

```
sue.add(7, 3);  
sue.subtract(15, 1);  
sue.multiply(6, 4);
```

Appendix K

Lesson Plan for Day 7

Summary of A2 of the ICT Textbook(Quesenberry, 2006)

Slide 1

Lesson 7

- To introduce methods, the use of jar files and expand on the understanding of classes and object.
- To expand on the use of JCreator with the program DrawSquare.

Slide 2

Java program

- Code: what the programmer types in, with words and symbols, to tell the computer what to do.
- Built from objects and classes that a programmer writes or reuses.

Slide 3

Program to solve a problem

- Understand the problem.
- Develop program requirements.
 - e.g. "Write a program to draw a square on a piece of paper with a pencil".
- To determine objects – think "nouns".
 - Pencil, a piece of paper, and a square.
- To determine methods – think "verbs".
 - turnLeft, forward, drawCircle, etc.

Slide 4

JCreator DrawSquare

- Open new project and call it DrawSquare (be sure to include library files folder).
- Make two separate files:
 - File, new, class:
 - DrawSquare – click Public, Generate default constructor.
 - File, new, class:
 - driver – click Public, Generate main method.

Slide 5

Classes

- Each class has its own role.
- Are built from instructions that are used in such a way that they manipulate objects to perform.
- Have behavior and attributes.

Slide 6

Driver

- Outside of class DrawSquare, but will be using that class.
- Purpose is to provide a simple environment to use to test methods of a class.

Slide 7

Jar files

- Add

```
import apcslib.*;
```


to program.
- Import statements tell compiler where it can find classes that your program uses.

Slide 8

Libraries

- Predefined classes.
- Java calls them packages.
- Java comes with many packages.
- Programmers have created packages that can be used.
 - e.g. apcslib.
- To use in program, use import statement.

Slide 9

Attributes/Object declarations

- In DrawSquare:

```
private DrawingTool myPencil;  
private SketchPad myPaper;
```
- Using existing classes to create objects.
- The DrawingTool object is given the name myPencil.
- The SketchPad object is given the name myPaper.

Slide 10

Public DrawSquare()

- This is a constructor.
- It is exactly the same name as class and as the file.
- The main task is to instantiate new objects.
- It is a method but can only be called when an object is first created.

Slide 11

Instantiate

- Type in public DrawSquare():

```
myPaper = new SketchPad(300, 300);  
myPencil = new DrawingTool(myPaper);
```
- Constructs a new SketchPad object named myPaper with dimensions 300 by 300.
- Constructs a new DrawingTool object named myPencil to be constructed using the SketchPad object named myPaper.
- myPaper is an instance of the SketchPad class.
- myPencil is an instance of the DrawingTool class.

Slide 12

Behaviors

- These are some of the behaviors from the DrawingTool class:

```
forward  
turnLeft  
getColor
```

Slide 13

Behaviors (continued)

- An object's behavior is determined by instructions within its methods.
- The instructions execute in the order they appear.

Slide 14

Behaviors (continued)

- Type in program:

```
public void draw( )  
{  
    myPencil.forward(100);  
    myPencil.turnLeft(90);  
    myPencil.forward(100);  
    myPencil.turnLeft(90);  
    myPencil.forward(100);  
    myPencil.turnLeft(90);  
    myPencil.forward(100);  
}
```

Slide 15

Behaviors (continued)

- When we call the method "forward" and pass the distance:
 - The object draws a line of the specified length.
- When we call the method "turnLeft" and pass the degrees:
 - The object turns left that number of degrees.
- A value we pass to an object's method is called an argument.

Slide 16

Behaviors (continued)

- forward and turnLeft carries out a request by the user but does not respond to the sender.
- getColor method returns a value to the sender.
 - e.g., we want to know the current color that is being used for drawing.

Slide 17

Summary of class DrawSquare

- Includes two methods.
 - DrawSquare
 - Used to construct new instances of this class.
 - draw()
 - Where most of the action for this class takes place.
 - Contains only those instructions directly related to the actual drawing of the square.

Slide 18

Driver

- Click on the driver tab.
- Type into program:

```
DrawSquare squareOne = new DrawSquare( );  
squareOne.draw( );
```
- squareOne is created (instantiated).
- Sending a message to the object squareOne to draw.

Slide 19

Compiling & Running a Program

- Compiling – converting a program into bytecode language the Java interpreter understands.
- Compilation errors - usually due to syntax rules.
- Run-time errors – those caught by the compiler that stops the running of the program. (e.g., dividing by zero).
- Logical errors – program runs but results are not what is expected (e.g., numbers should have been added instead of multiplied).

Slide 20

Compile & Run Program

- Click Build File or F7.
- Click Run File or F5.

Appendix L

Lesson Plan for Day 10

Summary of A2 of the ICT Textbook (Quesenberry, 2006)

Slide 1

Lesson 10 Objects & Classes

- Objective: to summarize and understand the similarities and differences between objects and classes.

Slide 2

Similarities

- An object has attributes as defined by its class.
- An object's behavior is restricted by the methods that are included in its class.
- Objects and classes have attributes and methods in common.

Slide 3

A Class

- Purpose of a class is to guide the creation of objects.
- It is a model of an object type.
- It is an abstract idea, a general concept.
- It is written by a programmer.
- It is named by a class name.

Slide 4

A Class (continued)

- The attributes for an object are defined in a class.
- All of the attributes are fixed. They do not change before, during or after execution of a program.
- Methods are defined in the class of the object.

Slide 5

Object

- Is created and eventually destroyed.
- Must be explicitly declared and constructed by the executing program.
- Rather than general concept, objects are specific and real instances of that concept.
- Objects from the same class all share common characteristics.

Slide 6

Object (continued)

- Attributes of an object will change.
- Methods are invoked on objects.
- Does not have the same name as a class name.
- Many objects of a class can be created.

Appendix M

Lesson Plan for Day 11

Summary of Chapter 1 (Bergin, 1997)

Slide 1

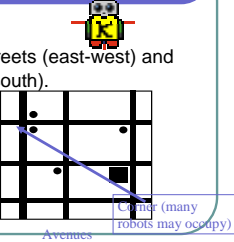
Lesson 11 Karel J. Robot Ch1

- Objective: to understand the Karel J. Robot world, its capabilities, tasks and situations.

Slide 2

Karel J. (the Robot)

- Robot World.
- A flat plane of streets (east-west) and avenues (north-south).
- "A" in avenue points north and "V" points south.



The diagram shows a grid world with a robot at the top center. A blue path starts from the robot and moves down, then left, then down again, ending at a black square. Labels 'Streets' and 'Avenues' are present. A legend indicates 'C' for 'C' (many robots may occupy)'.

Slide 3

Karel's World (continued)

- Contains Beepers & Walls.
- Beepers.
 - Emits a quiet beeping sound.
 - May be picked up, carried and placed again.
 - May place several on a corner and they don't interfere with Robot movement.



Slide 4

Robot Capabilities

- Move forward in the direction its facing.
- Turn.
- Sense surroundings.
 - Hear beepers (on same corner).
 - Determine direction it is facing.
- Pick up, carry, and put down beepers.
- Detect a wall ½ block away.
- Turn itself off.

Slide 5

Karel-Werke


- Factory – Builds the robots.
 - Standard model.
 - Write a spec for a new model.
 - Extension of an existing base model.
- Factory – Delivers the robots.

“factories”
(auf Deutsch)

Slide 6

Tasks & Situations

- **Examples:**
 - Move to a corner (3rd St. & 5th Ave.).
 - Run a race.
 - Escape from a maze.
 - Find a beeper and deliver it to the origin.



Appendix N

Lesson Plan for Day 12

Summary of Chapter 2 (Bergin, 1997)

Slide 1

Lesson 12 Karel J. Robot Ch2

- Objective: to understand the Karel J. Robot language and to write programs that instruct robots to perform simple obstacle avoidance and beeper transportation tasks.

Slide 2

Karel Primitive Instructions

- Basic tools with which all problems are solved (analogies: carpentry, geometry).
 - move()
 - turnLeft()
 - putBeeper()
 - pickBeeper()
 - turnOff()

Slide 3

OOP -ness

- `object.method1();`
- `object.method2();`

- Where a method **acts on** the object.
- Objects are typically nouns (e.g. karel).
- Methods are typically verbs (e.g. move) and represent behavior.

`karel.move();`

Slide 4

move()

- forward only, one block.
- “Error Shutoff” if trying to move into a wall (a duh! Look first!).

Slide 5

turnLeft()

- Stay at same corner.
- Turn 90 degree to the left.
- Cannot cause an error shutoff.

Slide 6

pickBeeper()

- Picks up beeper on current corner and places in beeper bag.
- Attempted on beeper-less corners causes an error shutoff.

Slide 7

putBeeper()

- Take beeper from beeper bag and put down on current corner.
- Attempted with an empty beeper bag causes an error shutoff.

Slide 8

turnOff()

- Robot turns off and is incapable of executing another instruction until restarted.
- The last instruction executed on a robot object.

Slide 9

Description

urRobot class (the "Model-T" Robot)

```
Public class urRobot
{
    void move(){...}
    void turnOff(){...}
    void turnLeft(){ }
    void pickBeeper(){...}
    void putBeeper(){...}
}
```

"primitive"
You can't/don't need to define this class - it is in a library for you to use

Slide 10

Sample Program

```
import kareltherobot.*;

Public class SampleTest implements Directions
{
    public static void main(String args[] )
    {
        urRobot karel = new urRobot(2, 1, East, 0);
        karel.move();
        karel.move();
        karel.turnLeft();
        karel.turnOff();
    }
}
```

Slide 11

Error Classification

- 4-types.
 - Lexical error (compiler catches).
 - Word not in its dictionary.
 - Syntax error (compiler catches).
 - Incorrect grammar, punctuation, incorrect location of a statement.

Slide 12

Error Classification (continued)

- Execution error (run-time environment catches).
 - Can't perform what you ask (at run-time).
- Intent error (logic – guess who catches this one!).
 - Program terminates successfully – junk output, however.

Which is the hardest type of error to correct? Why?

Slide 13

Now You Try a Short Program

- On Paper – 10 minutes:
- Start a robot (HBot) at position (1, 1) (a.k.a. the "origin") facing North and having an infinite number of beepers.
- Have the robot "print" (using beepers) the letter H (with 3-beeper sides and a 1-beeper bar).
- When done, the robot should be facing North back at the origin (leaving the world in its original state of being).

Slide 14

Programming the HBot together

- Let's go into JCreator and write the HBot.
- Start a new project called HBot.
- Name a class HBot.

Slide 15

HBot (continued)

- Before public class, type:

`import kareltherobot.*;`

Slide 16

HBot (continued)

- Add:

implements Directions after public class HBot
- It should now look like this:

`public class HBot implements Directions`

Slide 17

HBot (continued)

- Type into the HBot class:

`public void task()
{
 World.setVisible(true);
 Robot karel = new Robot(1, 1, North, 7)
}`
- Add instructions.

Appendix O

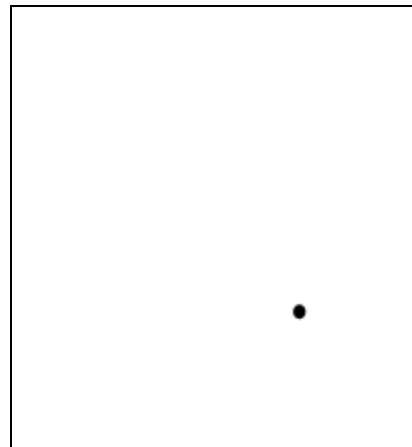
Quiz 1, Unit 1

**INTRO TO COMP SCI
QUIZ #1**

NAME _____

1. Define rachel as an Acrobat.
2. Tell rachel to clap 5 times.
3. Beginning at the point in the box and facing up, draw the figure generated by the following code segment:

```
myPencil.forward(100);  
myPencil.turnLeft(120);  
myPencil.forward(100);  
myPencil.turnLeft(120);  
myPencil.forward(100);
```



Appendix P*Quiz 1, Unit 1 – Answer Key***INTRO TO COMP SCI
QUIZ #1**

NAME _____

ANSWERS

1. Define rachel as an Acrobat.

```
Acrobat rachel = new Acrobat( )
```

3 pts

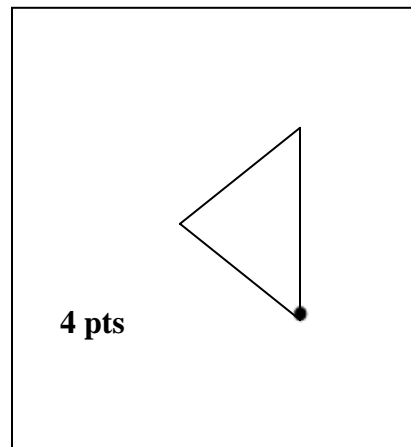
2. Tell rachel to clap 5 times.

```
rachel.clap(5)
```

3 pts

3. Beginning at the point in the box and facing up, draw the figure generated by the following code segment:

```
myPencil.forward(100);  
myPencil.turnLeft(120);  
myPencil.forward(100);  
myPencil.turnLeft(120);  
myPencil.forward(100);
```



Appendix Q*Test, Unit 1***INTRO TO COMP SCI
TEST, UNIT #1**

NAME _____

For #1-6, True or False:

1. A constructor must have the same name as its class.
2. An object may be made up of other objects.
3. Only one object may be created from a particular class.
4. We invoke a method.
5. A class is a type of object.
6. A constructor can only be called when an object is first created.

For #7-8, choose the best response.

7. Object is to class as
 - a. line segment is to triangle
 - b. house is to blueprint
 - c. blueprint is to house
 - d. bicycle is to car
 - e. car is to bicycle
8. Which statement would we use to create an object from a class called Flower:
 - a. Flower pretty;
 - b. Flower pretty = Flower();
 - c. Flower pretty = new Flower;
 - d. Flower pretty = new Flower();

e. `new Flower() = pretty;`

9. Carefully inspect the following program and correct all errors.

```
Public void draw( )
{
    myPencil.forward(100);
    myPencil.TurnLeft(90);
    myPencil.forward(100);
    my.Pencil.turnLeft(90);
    myPenicl.forward(100);
    myPencil.turnLeft(90)
    myPencil.forward(100);
}
```

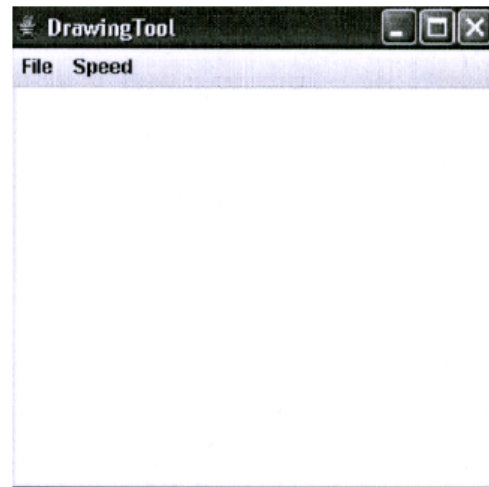
10. What is the difference between an object and a class?

11. What does a constructor do?

12. Assume the following object declarations and initializations. This code will create a DrawingTool object called *crayon* and a SketchPad object called *board*. The *board* will have dimensions of 300 X 300. The *crayon* is constructed to be used with the *board*. The drawing will begin at the center of the board at the point (0,0) and faces up.

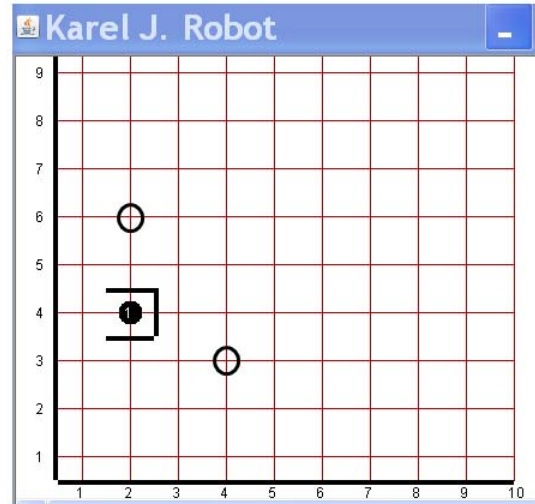
Starting in the center of the box, draw the figure generated by the following code segment:

```
crayon.turnRight(45);  
crayon.forward(40);  
crayon.turnRight(90);  
crayon.forward(40);  
crayon.turnLeft(90);  
crayon.forward(20);  
crayon.setDirection(-90);  
crayon.forward(30);  
crayon.turnRight(135);  
crayon.forward(20);  
crayon.turnLeft(90);  
crayon.forward(40);  
crayon.turnRight(90);  
crayon.forward(40);  
  
crayon.up();  
crayon.move(10,10);  
crayon.down();  
  
crayon.drawCircle(2);
```



13. Write a program which will create a robot at the origin facing north with 1 beeper. The robot should move to pick up the beeper as seen in the drawing below. Then it should place beepers at the two empty circles and end up at the origin facing north and turn off. Try to use the least number of statements possible.

```
public class Test1 implements Directions
{
    public static void main(String[] args)
    {
        // your code goes here....
    }
}
```



Appendix R*Test, Unit 1 – Answer Key***INTRO TO COMP SCI
TEST, UNIT #1**

NAME _____ 40 points _____

ANSWERS**For #1-6, True or False:**

1. A constructor must have the same name as its class. *True (1 pt)*
2. An object may be made up of other objects. *False (1 pt)*
3. Only one object may be created from a particular class. *False (1 pt)*
4. We invoke a method. *True (1 pt)*
5. A class is a type of object. *False (1 pt)*
6. A constructor can only be called when an object is first created. *True (1 pt)*

For #7-8, choose the best response.

7. Object is to class as
 - a. line segment is to triangle
 - b. house is to blueprint *b (2 pts)*
 - c. blueprint is to house
 - d. bicycle is to car
 - e. car is to bicycle
8. Which statement would we use to create an object from a class called Flower:
 - a. Flower pretty;
 - b. Flower pretty = Flower();
 - c. Flower pretty = new Flower;
 - d. Flower pretty = new Flower(); *d (2 pts)*
 - e. new Flower() = pretty;

9. Carefully inspect the following program and correct all errors.

```
Public void draw( )
{
    myPencil.forward(100);
    myPencil.TurnLeft(90);
    myPencil.forward(100;
    my.Pencil.turnLeft(90);
    myPenicl.forward(100);
    myPencil.turnLeft(90);
    myPencil.forward(100)
}
```

*t in Turn should be lowercase
parenthesis missing after 100
no "dot" between my & Pencil
Pencil is misspelled
missing semi-colon
(5 pts)*

10. What is the difference between an object and a class?

*- A class defines attributes and methods; objects' attributes and methods are defined by a class.
- A class cannot be altered during program execution; objects are created and destroyed and have attributes that can change in value and methods that can execute during program execution.
- A class is named by a class name; object is most often referenced using an identifier and is an instance of that class..*

(3 pts)

11. What does a constructor do?

A constructor's main task is to instantiate new objects. (2 pts)

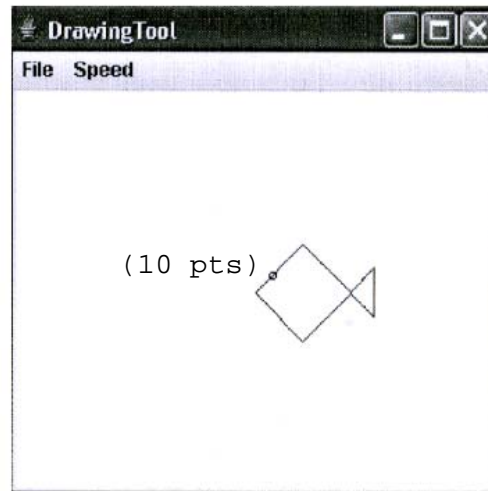
12. Assume the following object declarations and initializations. This code will create a DrawingTool object called *crayon* and a SketchPad object called *board*. The *board* will have dimensions of 300 X 300. The *crayon* is constructed to be used with the *board*. The drawing will begin at the center of the board at the point (0,0) and faces up.

Starting in the center of the box, draw the figure generated by the following code segment:

```
crayon.turnRight(45);
crayon.forward(40);
crayon.turnRight(90);
crayon.forward(40);
crayon.turnLeft(90);
crayon.forward(20);
crayon.setDirection(-90);
crayon.forward(30);
crayon.turnRight(135);
crayon.forward(20);
crayon.turnLeft(90);
crayon.forward(40);
crayon.turnRight(90);
crayon.forward(40);

crayon.up();
crayon.move(10,10);
crayon.down();

crayon.drawCircle(2);
```



13. Write a program which will create a robot at the origin facing north with 1 beeper. The robot should move to pick up the beeper as seen in the drawing below. Then it should place beepers at the two empty circles and end up at the origin facing north and turn off. Try to use the least number of statements possible.

```
public class Test1 implements Directions
```

```
{
```

```
    public static void main(String[] args)
```

```
    {
```

```
        // your code goes here....
```

```
        Robot karel = new Robot(1, 1, North, 1);
```

```
        karel.move();
```

```
        karel.move();
```

```
        karel.move();
```

```
        karel.turnLeft();
```

```
        karel.turnLeft();
```

```
        karel.turnLeft();
```

```
        karel.move();
```

```
        karel.pickBeeper();
```

```
        karel.turnLeft();
```

```
        karel.turnLeft();
```

```
        karel.move();
```

```
        karel.turnLeft();
```

```
        karel.move();
```

```
        karel.turnLeft();
```

```
        karel.move();
```

```
        karel.move();
```

```
        karel.move();
```

```
        karel.putBeeper();
```

```
        karel.turnLeft();
```

```
        karel.move();
```

```
        karel.move();
```

```
        karel.move();
```

```
        karel.turnLeft();
```

```
        karel.move();
```

```
        karel.move();
```

```
        karel.putBeeper();
```

```
        karel.move();
```

```
        karel.turnLeft();
```

```
        karel.move();
```

```
        karel.move();
```

```
        karel.move();
```

```
        karel.move();
```

```
        karel.move();
```

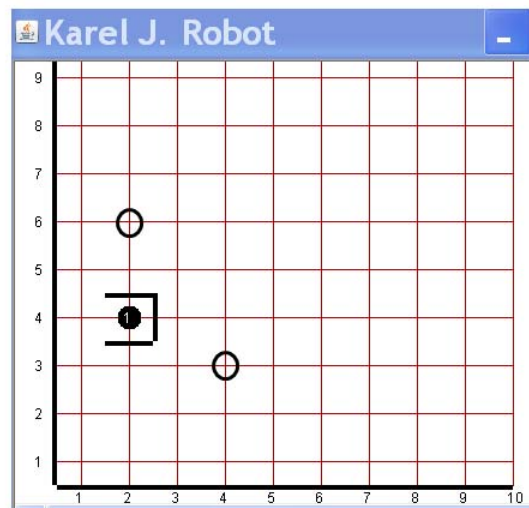
```
        karel.turnLeft();
```

```
        karel.turnLeft();
```

```
        karel.turnOff();
```

```
    }
```

```
}
```



(10 pts)