AP® Computer Science A
Syllabus

Overview of AP® Computer Science A

Computer Facilities

Our classroom is also our computer lab. We have our computers around the outside of the room, with the center set up in a traditional classroom fashion, with daily availability of overhead computer projection and screen, smart board capabilities, a white board and chalk board. Our lab and the labs around the building are managed and maintained by a full-time technical support staff. They save us countless hours and ensure that we are up and running 100 percent of the time. This course is on a tight schedule; any downtime during lab is detrimental to student learning. Classes are 43 minutes long 4 times per week and 65 minutes long once per week The computer Lab is available after school, one 3 hour Saturday each term and 3 hour AP classes are required for all students during state mandatory testing. Each student has a computer workstation. Students have accounts on a networked server and store all of their class files in their accounts. All students have access to all instructional material filed electronically.

Course Overview

The content of Computer Science A is a subset of the content of Computer Science AB. Computer Science A is an introductory course emphasizing object-oriented programming methodology with a concentration on problem solving and algorithm development and is meant to be the equivalent of a first-semester college level course in Computer Science. It includes the study of data-structures, design and abstraction. The emphasis is on design issues that make programs understandable, adaptable and when appropriate reusable. At the same time, the development of useful computer programs and classes is used as content for introducing important concepts in computer science. In addition, an understanding of basic hardware and software components of computer systems and the responsible use of these systems is are integral parts of this course. Current offerings of the AP CS exam require the use of Java. Math and computer pre-requisites are required for registration. The novice will start not by designing a whole program but rather by studying programs already developed, then writing or modifying parts of a program to add or change its functionality, continuing the study of objects primarily as an instance of a class. A complete list of topics can be found on p. 7-11 at AP Central College Board's website.
http://apcentral.collegeboard.com/apc/public/repository/52435apcompscilocked_4315.pdf

Assessment

A variety of instructional techniques and strategies are employed with an emphasis on cooperative class work assignments including investigative and exploratory activities.
A variety of assessment tools are used that emphasize an integrated approach to learning. Students are expected to clearly demonstrate their methods, testing and analyses.
Quizzes, tests and homework and class work assignments include a combination of multiple choice, short-answer, open response type questions, and projects in line with the concepts being learned. Cumulative reviews follow each chapter and include released AP free response questions. Self and peer-evaluations and comparisons using scoring rubrics will be used.

Individual and group exercises and projects are designed to learn, practice and review the material learned to date and will be structured so that students integrate the key components of the programming process according to the waterfall model of design. Throughout the year, students will practice and gain proficiency in communicating verbally and in writing, journaling, using flow charts, other structural design, writing and testing code.

Texts

***Primary Textbook:**

- Litvin, Maria and Litvin, Gary. *Java Methods A & AB: Object Oriented Programming and Data Structures.* Skylight Publishing, 2006. **http://www.skylight.com**

***Supplementary Textbook:**

- Lambert/Osborne. *Fundamentals of Java: AP Computer Science Essentials for the A and AB Exams.* Thomson Course Technology 2007.

http://www.course.com/catalog/product.cfm?category=Computer%20Science&subcategory=AP%20Computer%20Science&isbn=978-0-619-26723-0

Additional Resources:

• The College Board's GridWorld Case Study

• AP Central®: Computer Science A Quick Reference Guide

- Bergin, J., et al. *Karel J. Robot: A Gentle Introduction to the Art of Object-Oriented Programming Using Java.* Copyright Joseph Bergin. http://csis.pace.edu/~bergin/KarelJava2ed/Karel++JavaEdition.html
- Bloch, J. and Gafter, N. Java Puzzlers by. Addison-Wesley, 2005.
- Bolker, E. and Campbell; Java Outside In
- Burnette, E. Eclipse IDE Pocket Guide. O'Reilly 2005
- Eckel, B Thinking in Java fourth edition; Prentice Hall 2006.
- Eubanks, B Wicked Cool Java. No Starch Press Inc. CA. 2005
- Guzdial, M. and Ericson, B. **Introduction to Computing and Programming with Java: A Multimedia Approach**
- Horstmann, Cay. *Big Java*. New York: Wiley, 2002.
  Power Point presentation from Fundamentals and Cay Horstmann's student companion website:
  http://bcs.wiley.com/he-bcs/Books?action=index&bcsId=2215&itemId=0471697044
- Barron's AP review
- Litvin's AP Review
- Raposa, R. Java 60 Minutes a Day. Wiley Publishing.2003
- Touretzky, D. *Common Lisp*, The Benjamin Cummings Publishing Company Inc. 1990; Ch 8 p. 231 (243-283 of PDF) http://www.cs.cmu.edu/~dst/LispBook/book.pdf
- Weiss, Mark Allen. *Data Structures and Problem Solving Using Java*, 2nd ed. New York: Addison Wesley, 2002.

AP Computer Science

General Outline & Resources *continued*

Instructional Activities	**Assessment**
Direct Instruction	Quizzes
Class Exercises / Activities	Chapter Tests
Cooperative Learning	Class work
Homework Exercises	Homework
Study Guides / Chapter Reviews	Computer Lab exercises and projects
Spiral Activities	Self Assessment & Peer Reviews
Technology	Free Response Rubric
Integrated Skills Projects	
Released AP Free Response	
Independent Reading	

Syllabus at a Glance

Unit	General Topic	Case study Code/ Primary Projects* (Methods)	3 Week Sections	Chapters (Fundamentals)
1	Background CS	GridWorld Part 1	0–2	Ch 1
	First Java Programs	Karel J. Robot Conversion examples	3-5	Ch 2- 3
2	Syntax, Errors and Debugging	‘First Steps- Methods ch 3, ‘Dance Studio’ Methods ch 11 Euclid’s GCF algorithm- Methods ch 4 Bank Code JOI Histograms Open response 2000 ‘Rolling Dice’ Methods ch 7	6-8	Ch 3- 4
	Control Statements		9–11	Ch 4- 5
3	Defining Classes	GridWorld Part 2 ‘Pie Chart’ Methods ch 6 Towers of Hanoi; Common Lisp ch 8	12–14	Ch 5, Ch 10
	Control Statements Continued Recursion		15-17	Ch 6, 7
4	Arrays and ArrayList	GridWorld Part 3	18-20	Ch 9, Ch 11
	Quadratic Sorts and Linear/Binary Search	GridWorld Part 4	21-23	Ch 12
	Mergesort	Comparing Sort Algorithms Fundamentals	24-26	Ch 12-13
	Review (from Fundamentals)	12 Units	27-29	Ch 1-12

* Other selected exercises also included in weekly plan- Appendix 1

1

Course Outline [C2]

C2—The course includes all of the topics listed in the "Computer Science A" column of the Topic Outline in the AP Computer Science Course Description
C4—The course teaches students to use and implement commonly used algorithms and data structures.
C5—The course teaches students to develop and select appropriate algorithms and data structures to solve problems.
C9—The course teaches students to recognize the ethical and social implications of computer use.

Unit 1: Weeks 0

Summer reading list- one book of choice

Summer independent study with Alice, Karel and/or Robocode

Unit 1: Weeks 1-5

Introduction to the principal concepts in computer science using Grid World Case Study Part 1 and Karel J Robot.

Objectives [C4] [C5]

- Become familiar with the computer lab, accounts, and IDE 's
- Understand object-oriented programming and top-down design/refinement of individual tasks
- Basic class structure including instance variables, local variables, parameter passing, scope, public/private visibility, use of super
- Sequence, selection, and iteration
- Error categorization/correction
- Part 1 of GridWorld Case Study
- Creating projects and running the GridWorld Case Study
- Black-box testing
- Computer ethics and social implications
- Arithmetic operators

Teaching Strategies

To teach computer science concepts so that students have immediate visual feedback—at least in the beginning, while learning stepwise refinement. They can understand what they have done right and wrong because they can see it. Students should not lose sight of computer science as they examine the details of the computer language. This undertaking is not too difficult since algorithms that solve a variety of robot tasks, and Gridworld are both plentiful and provocative, as are the topics of study associated with them. I emphasize for creativity and imagination so that students can experience the results of computer science at a first-hand level, while keeping in mind the basic concepts.

A good place to begin talking about computer ethics is when we begin the case study. The students will immediately notice that each source file contains a statement referring to GNU licensing. From there I introduce them to both the ACM and IEEE and their published Codes of Ethics. Dr. Jody Paul has an excellent site listing many links that will help to facilitate thought and discussion among teachers and students. **[C9]**

References/Readings

Karel J. Robot and many other related ideas at the author's site.
Java Fundamentals A & AB, selected readings from Chapters 1-3.
Sample daily schedule includes PowerPoint presentations, labs, homework un-plugged, and review exercises; Introduction of API and Sun's tutorials.

Assignments/Labs

- See the weekly schedule, which includes homework assignments, labs, review exercises, PowerPoint presentations, and worksheets, quizzes and tests.

  Programs and activities: Binary card activity; Hello World, Hello World cup of java, Hello World with date; Gridworld Part 1; Karel J Robot; Temperature Conversion and modifications (kilometers to nautical miles, minutes in a year, momentum output); Income tax calculator; Drawing Shapes.

C3—The course teaches students to design and implement computer-based solutions to problems in a variety of application areas.
C7—The course teaches students to read and understand a large program consisting of several classes and interacting objects, and enables students to read and understand the current *AP Computer Science Case Study* posted on AP Central®.

2
Unit 2: Weeks 6–11

Conversion examples; Selected Case Studies

Objectives [C3] [C7]

- Source, bytecode, compilers, interpreters, Java virtual machine, Platform independence
- Computer software and hardware components, operating systems
- Basic logic gates and computer numbering systems
- Assignment statement, primitive data types
- Arithmetic operators cont., ArithmeticException, precedence, Casting/promotion
- Interfaces
- java.lang.Math (abs, pow, sqrt, random), static methods

Teaching Strategies

Classroom discussions on topics of processors, peripherals, and system software are ongoing throughout the course. Students discuss and identify major components and how they interact, while gaining familiarity with the operations of the hardware and software available in our school and ability to distinguish between a single-user system and a network. It is expected that all students will adhere to the Acceptable Users' Policy given by our district and signed as a freshman contract.
Interfaces are introduced by providing one for students and having them write a couple of classes that implement the interface. In this manner, I am giving their lab/class its basic structure, providing a lab specification, especially if it contains Javadoc. It's also a way to automate testing their labs, guaranteeing that the students' classes all have the same method signatures, enabling to easily test all of their methods.
I require that the labs be fault-tolerant, that is, handle incorrect data entered by the user, so I give them additional practice with selection, iteration, and string and primitive comparisons and conversions.

References/Readings

Jamtester, JUnit, and unit testing **www.jamtester.com**
Java Fundamentals A & AB, selected readings from Chapters 3-5

Assignments/Labs

- *Java Methods A and AB*, selected exercises and labs from chapters 1, 3, 4, 7 and 11
- *Fundamentals* selected exercises and labs from chapters 1 -5

Programs and activities: First Steps and Dance Studio (Methods); Employee total weekly pay calculator; Mulle-Lyer illusion; Folly of Gambling; Euclid's GCF algorithm; Bank code JOI; Population study; bases program; Rolling dice; Mode and Histogram Lab.

3

Unit 3: Weeks 12-17

Grid World Part 2, Towers of Hanoi, Selected Case Studies

Objectives [C6] [C8]

- Intercommunicating objects
- Data structure design and selection
- Feeling comfortable with the Case Study
- Parameter passing terminology and concepts
- String class, object references, aliasing
- Selection in more detail
- Object is the superclass of all superclasses, overriding toString()
- Recursion
- Inheritance and polymorphism, overriding methods
- java.lang.Math.random(), RandNumGenerator
- Analyze, design, code, and test software

C6 - The course teaches students to code fluently in an object-oriented paradigm using the programming language Java. The course teaches students to use standard Java library classes from the AP Java subset delineated in Appendices A and B of the *AP Computer Science Course Description*. (Note: Students who study a language other than Java in AP Computer Science must also be taught to use Java, as specified in the AP Java subset.)

C8—The course teaches students to identify the major hardware and software components of a computer system, their relationship to one another, and the roles of these components within the system.

Teaching Strategies

The GridWorld Case Study can be sliced into byte-sized pieces by incorporating some of the classes as early as possible in the course. I have the students comfortable with the case study making changes and using it in investigations of concepts as students gain comfort using libraries and objects and writing and designing object-oriented code. As they are mastering these tasks, they are also mastering important AP concepts. Recursion is introduced at a conceptual then a working level.

References/Readings

Java Fundamentals A & AB, selected readings from Chapters 5-8, 10.
Dr. Jody Paul **www.jodypaul.com/SWE/ethics.html ;** Ch 8 Common Lisp

Assignments/Labs

- Towers of Hanoi Activities
- Exercise sets in Part 2 of the GridWorld Case Study
- *Java Methods A and AB*, selected exercises and labs from chapter 6.
- Mode and Histogram lab (based on 2000 AP exam question)
- *Fundamentals* selected exercises and labs from chapters 5-8

Programs and Activities: GridWorld Part 2; Student Class and Shape extensions; Weekly pay extension; Truth tables; Tax tables; Pie Chart (Methods); Fibonacci numbers; Circle animations; Common Lisp reading; Perfect numbers (Methods ch 8); Towers of Hanoi; Thermometer Class extension.

4
Unit 4 Section 1: Weeks 18-20

Arrays and Array Lists; Grid World Part 3 and 4

Objectives [C3] [C4] [C5] [C6] [C7]

- Declaring, constructing, initializing, and indexing arrays/ArrayList
- Storing primitives and objects in arrays/ArrayList
- Traversing, inserting, deleting array/ArrayList elements
- Passing arrays/ArrayList to methods
- Wrapper classes—Double, Integer
- Casting, ClassCastException, ArrayIndexOutOfBoundsException
- Java 5.0's Generics
- Java 5.0's enhanced for loop
- Inheritance and polymorphism
- Data structure/algorithm selection and design
- Interfaces (Comparable, Locatable) and Abstract classes

Teaching Strategies

Students get an informal look at arrays working with parts 2 and 3 of the GridWorld Case Study. Now we go into it in depth. The first few labs in this section are small and focused, used for practicing simple array traversals, insertions, and deletions. I keep it simple at this point and not embed array concepts within too many object-oriented concepts. Afterward, I then introduce them to some object-oriented GUI labs to give them even more practice with arrays and ArrayLists.

By this point in the year, students have a working knowledge of the Java language and object-oriented principles and can complete the last chapter of the Gridworld case study and have fun having it will fresh in their mind while taking the AP Exam. This is a great time to give students more practice with selecting and designing data structures and algorithms on their own. Within the context of the GridWorld Case Study there are several lab ideas that will help students further hone in their data structure and algorithm design skills. The main idea is to have them working within the many classes and to become extremely comfortable with where things are and how they work. Role playing in order to see the big picture is part of 'CS-unplugged' activities. Seeing and acting out the object responsibilities will help students internalize the complex intercommunication. I like to be creative and let everyone have fun with it. Professor Levine shows how to use role-plays.

References/Readings

GridWorld Case Study Parts 3 and 4
Fundamentals Ch 9, Ch 11, Ch 12
Java Methods A & AB, Chapter 12

Assignments/Labs

- Exercise sets in Parts 3 and 4 of the GridWorld Case Study
- *Fundamentals* selected exercises and labs from chapters 9-12
- *Java Methods A & AB*, Chapter 12 exercises and labs

Programs and Activities: GridWorld Part 3 and 4; Student Class extension; Field Trip to commercial robotics lab; Building a deck of cards

C3—The course teaches students to design and implement computer-based solutions to problems in a variety of application areas.
C4—The course teaches students to use and implement commonly used algorithms and data structures.
C5—The course teaches students to develop and select appropriate algorithms and data structures to solve problems.
C6 - The course teaches students to code fluently in an object-oriented paradigm using the programming language Java. The course teaches students to use standard Java library classes from the AP Java subset delineated in Appendices A and B of the *AP Computer Science Course Description*. (Note: Students who study a language other than Java in AP Computer Science must also be taught to use Java, as specified in the AP Java subset.)
C7—The course teaches students to read and understand a large program consisting of several classes and interacting objects, and enables students to read and understand the current *AP Computer Science Case Study* posted on AP Central®.

C3—The course teaches students to design and implement computer-based solutions to problems in a variety of application areas.
C4—The course teaches students to use and implement commonly used algorithms and data structures.
C5—The course teaches students to develop and select appropriate algorithms and data structures to solve problems.

6

Unit 4 Section 2: Weeks 21–23

Quadratic Sorts and Linear/Binary Searching

Objectives [C3] [C4] [C5]
- Insertion and selection sorts
- Sequential versus binary searching
- Introduction to some friendly Big-Oh ideas
- Recursion revisited

Teaching Strategies
While working with the traditional sorts and searches, I introduce some simple Big-Oh concepts and counting. Big-Oh is not part of the AP CS A Exam, but the counting of statements being executed is a part. I have the students count comparisons done while sorting and then graph the results. We discover why comparisons/operations relevant to the dataset size are used as a benchmark as opposed to execution speed. I also use the algorithms that they have studied up to now (e.g., reading data, common array algorithms) into their respective Big-Oh family.
This is a good place to work recursion back into the course, since I we can explore further how the linear and binary searches can be written both iteratively and recursively.

References/Readings
Java Methods A & AB, Chapters 4 and 13
Big-Oh handout
The xSortLab Applet **http://math.hws.edu/TMCM/java/xSortLab**

Assignments/Labs
- Worksheets and sample source code—sorting, searching, recursion, counting iterations, analysis
- *Fundamentals* selected exercises and labs from chapters 12-13
- *Java Methods A & AB*, Chapters 4 and 13

Programs and Activities: GridWorld Part 4; java.util.ArrayList; Towers of Hanoi extension; Many Queens problem; Comparing Sort Algorithms.

7
Unit 4 Section 3: Weeks 24–26

Mergesort

Objectives [C3] [C4] [C5] [C6]

- Mergesort
- Recursion
- (optional) java.util.Arrays and java.util.Collections

C6 - The course teaches students to code fluently in an object-oriented paradigm using the programming language Java. The course teaches students to use standard Java library classes from the AP Java subset delineated in Appendices A and B of the *AP Computer Science Course Description*. (Note: Students who study a language other than Java in AP Computer Science must also be taught to use Java, as specified in the AP Java subset.)

Teaching Strategies

Students will gain additional practice with arrays as they explore the nontrivial task of merging two sorted lists. In addition, students will once again see a comparison between a recursive and nonrecursive solution to an algorithm. Now that the students have had a chance to play with all of the sorts and searches in the AP curriculum, I like to introduce them to two more powerful and fun classes, java.util.Arrays and java.util.Collections. By this time in the course the students are quite adept at reading an API; this gives them a bit more practice.

References/Readings

Java Methods A & AB, Chapter 13
Fundamentals chapters 12-13

C3—The course teaches students to design and implement computer-based solutions to problems in a variety of application areas
C4—The course teaches students to use and implement commonly used algorithms and data structures.
C5—The course teaches students to develop and select appropriate algorithms and data structures to solve problems.

Assignments/Labs

- *Fundamentals* selected exercises and labs from chapters 12-13
- *Java Methods A & AB*, Chapter 13

Programs and Activities: Abstract Art- generating recursive patterns, what are fractals?; Search and Sort team presentations.

8
Weeks 27–29

Review

Objectives

- Ensure students know what is coming on the AP Exam
- Earn a 5 on the AP Exam

Appendix 1

WEEKLY CALENDAR
Personal binder to have dated code, weekly worksheets, tests and quizzes corrected- separated into 4 units

Week	**Unit**	**Section**	**Weekly reading Fundamentals**	**Primary Case Studies- Announced exercises due at end of unit or section**	**Main topic**	**Weekly Labs- all weekly exercises and projects due on Fridays.**	**Test or Quiz / Worksheets due Wednesdays HW: Additional 1-2 day short written assignments TBA**
0	1	1	Ch 1	GridWorld part 1	History and ethics/ IDE's		
1	1	1	Ch 1	GridWorld part 1	Hardware and software/ Binary representation	Binary cards HelloWorld (2) and Hello world with current date Gridworld exercises part 1	Worksheet 1
2	1	1	Ch 1	GridWorld part 1	Development and Design process	Gridworld exercises part 1 GUI Window	Quiz 1
3	1	2	Ch 2	Karel J. Robot Temperature conversion and modifications ch 2.6 Fundamentals	Why Java?; JVM; Edit compile and execute details; IO	Convert Projects 2-3, 2-4, 2-5 in teams; Project 2.6 each student	Worksheet 2
4	1	2	Ch 3	Karel J. Robot- ch9	Language Elements	Income tax calculator p.79 Count Angles p.88	Quiz 2
5	1	2	Ch 3	Karel J. Robot- ch 10	Syntax and semantics	Drawing Shapes- Graphics Class	Worksheet 3
6	2	1	Ch 3	First Steps and Dance Studio- Methods ch 3 and 11	Programming errors and debugging	I/O Projects 3.4 and modification 3.5 weekly pay- modify to not accept invalid data. Project 3.6- illusion	Quiz 3
7	2	1	Ch 4	First Steps and Dance Studio- Methods ch 3 and 11	Additional Operators	Folly of Gambling p. 130	Worksheet 4
8	2	1	Ch 4	Euclid's GCF algorithm- Methods ch 4 (P6.5 Fundamentals) Bank Code JOI	Standard Classes and Methods	Project 4.6 a population study	Quiz 4
9	2	2	Ch 4	Rolling Dice	If/ Else / While and for	Project 4.7-4.9 base 2	Worksheet 5
10	2	2	Ch 4	Rolling Dice modifications Histograms	Nest Control and Break statement Design testing and debugging	Project 4-13 induced contrast	Quiz 5
11	2	2	Ch 5	Histograms with presentations	Internal structure of Classes and Objects	Case Study- Student Class and test scores and project 5.1 modification adding constructors and testing Bank Account project 5.5	Worksheet 6
12	3	1	Ch 5	GridWorld Part 2	Structure and behavior of Methods; Scope and lifetime of variables		Quiz 6
13	3	1	Ch6	GridWorld Part 2	Interfaces- the client perspective, Implementation perspective	Case Study: Weekly Pay p.206 Exercise 6.3 p. 218 Truth table	Worksheet 7
14	3	1	Ch 6	Pie Chart – Methods ch 6	Code re-use through inheritance; Abstract classes	Exercise 6.4 Tax table p. 222 Case study: Fibonacci numbers p. 226	Quiz 7 Ex 6.5 homework
15	3	2	Ch 10	Towers of Hanoi activities	Logical operators; Logical errors in nested if statements; Testing and verification of loops	6.8 Animations Ch 10 modifications and further study of Student Class and Shapes	Worksheet 8
16	3	2	Ch 7	Common Lisp reading	Input/ Output	'Perfect numbers' Methods	Quiz 8

						ch 8	
17	3	2	Ch7	Common Lisp reading Towers of Hanoi code	Handling number format exceptions during input	Thrmometer Class- Fundamentals ch 7- with menu	Worksheet 9
18	4	1	Ch 9	GridWorld Part 3	Array manipulation; Looping through arrays; declaring arrays; Two-dim arrays; Enhanced for-loops; Arrays, methods and objects; Adding and removing elements from arrays	Student Class revisited Project 9.6 together. Students in pairs work together on 9.1-9.4. Choosing team projects for short presentation.	Quiz 9
19	4	1	Ch 9	GridWorld Part 3			Worksheet 10 Field Trip to commercial Robotics lab
20	4	1	Ch 11	GridWorld Part 4	Advanced operations on strings; Searching; Sorting	Case study- building a deck of cards	Quiz 10
21	4	2	Ch 11	GridWorld Part 4	Insertions and removals; Working with arrays of objects; Class java.util.ArrayList	Projects 11-1 count words, avg word length, sentence length. and 11-2 10 int sort	Open Response Q 2007
22	4	2	Ch 12	Comparing Sort Algorithms Fundamentals	Recursive patterns; Complexity analysis;	Towers of Hanoi revisited. P12.6Case Study: Many Queens problem. P12.7	Multiple choice practice quiz 1
23	4	2	Ch 12	Comparing Sort Algorithms Fundamentals	Binary search; Quicksort	Assign presentations Case Study: Comparing Sort Algorithms all	Open Response Q 2006
24	4	3	Ch 12	Comparing Sort Algorithms Fundamentals	Merge sort	Abstract Art- generating recursive patterns, what are fractals? P12.8 all	Multiple choice practice quiz 2
25	4	3	Ch 13	Search and Sort Presentations	Overview of Analysis and Design	Case Study: Comparing Sort Algorithms cont. al Search and Sort Presentations	Open Response Q 2005
26	4	3	Ch 13	Search and Sort Presentations		Search and Sort Presentations	Multiple choice practice quiz 3
27	4	4-review	Ch 1-4	SATURDAY SESSION* 2007 Open response A	Add exercises with solutions from ch 1-12 folders	Review	Open Response Q 2004
28	4	4-review	Ch 5-8			Review	Multiple choice practice quiz 4
29	4	4-review	Ch 9-12			Review	Practice test
Additional weeks after AP test	5	Project	Ch 8	Robotics Robocode L2Bot and more		Appliance Applet	Project based assessment Field trip to University Robotics Lab

***One of the last Saturdays of each term, dates to be announced the computer lab will be available.**

Daily Plan

Mondays	Receive worksheet or quiz due on Wednesday Introduce topic and weekly agenda Power Point presentation from Fundamentals and Cay Horstmann's student companion website: http://bcs.wiley.com/he-bcs/Books?action=index&bcsId=2215&itemId=0471697044
Tuesdays	Lecture/ Computer Lab
Wednesday	Computer Lab after worksheet or Quiz turned in
Thursday	CS unplugged 5-20 min activity
Friday	Computer Lab

Note: Power point projection of all labs in process is available for group discussion. Labs will be combinations of short exercises, project based exercises, group work and individual work. Collaboration is generally encouraged.

Fundamentals Chapter Objectives

Chapter 1—Background

Objectives

- Give a brief history of computers.
- Describe how hardware and software make up computer architecture.
- Understand the binary representation of data and programs in computers.
- Discuss the evolution of programming languages.
- Describe the software development process.
- Discuss the fundamental concepts of object-oriented programming.

Lecture Notes

Understand the concept of object-oriented programming (as well as how it differs from top-down design) analogous to working in teams of experts Business Model: manager, data entry individuals, and accounting specialists. As related to the concepts of classes, methods, encapsulation, information hiding, polymorphism, and inheritance.

Chapter 2—First Java Programs

Objectives

- Why is Java is an important programming language?
- Understand the Java virtual machine and byte code.
- Choosing a user interface style.
- Know the structure of a simple Java program.
- Write a simple program.
- Edit, compile, and run a program using a Java development environment.
- Format a program to give a pleasing, consistent appearance.
- Understand compile-time errors.
- Write a simple graphics program.

Lecture Notes

The short, simple programs in this chapter are easy to understand and learn. The best way to teach them is to take students through each step of the process — edit, compile, and run. First, run them off the shelf to show the results, and then edit one to insert a syntax error to demonstrate what happens during syntax checking. Fix the error, compile, and run again.
Program code consists of data and operations. Attention to these elements should be paid in each program. The programs use strings and numbers as data and perform input, arithmetic, and output as operations. Using short, simple programs that produce immediate results dispels any mystery and enables students to create their own programs. Why Java? Secure, robust, and portable.

Chapter 3—Syntax, Errors, and Debugging

Objectives

- Construct and use numeric and string literals.
- Name and use variables and constants.

- Create arithmetic expressions.
- Understand the precedence of different arithmetic operators.
- Concatenate two strings or a number and a string.
- Know how and when to use comments in a program.
- Tell the difference between syntax errors, run-time errors, and logic errors.
- Insert output statements to debug a program.
- Understand the difference between Cartesian coordinates and screen coordinates.
- Work with color and text properties.

Lecture Notes

This chapter examines in depth the syntax and semantics of basic program elements such as variables, primitive data types, arithmetic expressions, string expressions, and input/output. The chapter also lays the foundation for good programming habits. These include the use of informative variable names, appropriate program comments, and an organized program structure. The chapter discusses the differences between syntax errors, semantic errors, and logic errors and teaches techniques for debugging simple programs with these errors. Finally, two case studies introduce the ideas of analysis and design before coding a program.

Chapter 4—Introduction to Control Statements

Objectives

- Use the increment and decrement operators.
- Use standard math methods.
- Use if and if-else statements to make choices.
- Use while and for loops to repeat a process.
- Construct appropriate conditions for control statements using relational operators.
- Detect and correct common errors involving loops.

Lecture Notes

This chapter introduces students to the ideas of selection and repetition. Until now, their programs functioned like pocket calculators — accepting inputs, performing calculations, and displaying results. They now learn how to write programs that make decisions based on the inputs they receive and that repeat the same set of tasks. This chapter focuses on the simplest logic of control using if, if-else, while, and for statements. More complex control logic is examined in Chapter 6.

Chapter 5—Introduction to Defining Classes

Objectives

- Design and implement a simple class from user requirements.
- Organize a program in terms of a view class and a model class.
- Use visibility modifiers to make methods visible to clients and restrict access to data within a class.
- Write appropriate mutator methods, accessor methods, and constructors for a class.
- Understand how parameters transmit data to methods.
- Use instance variables, local variables, and parameters appropriately.
- Organize a complex task in terms of helper methods.

Lecture Notes

This chapter introduces students to class definitions. The view of classes in this chapter is simple: they are repositories of data and the methods for operating on those data. Nonetheless, many concepts are covered. The more important of these is data encapsulation. Students should work in small groups to thoroughly discuss the request, analysis, design, and implementation of the Case Study. When going over the Case Study, students should be able to point out methods, objects, global variables, local variables, instance variables, and visibility modifiers. More advanced concepts related to classes, such as inheritance and polymorphism, are deferred until Chapter 10.

Chapter 10—Classes Continued

Objectives

- Know when it is appropriate to include class (static) variables and methods in a class.
- Understand the role of Java interfaces in a software system and define an interface for a set of implementing classes.
- Understand the use of inheritance by extending a class.
- Understand the use of polymorphism and know how to override methods in a superclass.
- Place the common features (variables and methods) of a set of classes in an abstract class.
- Understand the implications of reference types for equality, copying, and mixed-mode operations.

- Know how to define and use methods that have preconditions, postconditions, and throw exceptions

Lecture Notes

This chapter shows students the full power of object-oriented programming. Primary topics include the use of interfaces to specify the behavior of a set of implementing classes and the use of polymorphism and inheritance to organize and simplify a design and its coding. Along the way, students will learn to use these tools to decompose software systems into cooperating classes with well-defined roles and responsibilities that include documentation and error handling.

Chapter 6—Control Statements Continued

Objectives

- Construct complex Boolean expressions using the logical operators `&&` (AND), `||` (OR), and `!` (NOT).
- Construct truth tables for Boolean expressions.
- Understand the logic of nested `if` statements and extended `if` statements.
- Test `if` statements in a comprehensive manner.
- Construct nested loops.
- Create appropriate test cases for `if` statements and loops.
- Understand the purpose of assertions, invariants, and loop verification.

Lecture Notes

This chapter continues the discussion of control structures by examining logical operators, compound Boolean expressions, nested selection statements, and nested loops. The focus is on designing correct control statements. Students learn strategies for testing programs that contain control statements. There are two Case Studies designed to help students understand these concepts.

Chapter 7—Improving the User Interface

Objectives

- Construct a query-driven terminal interface.
- Construct a menu-driven terminal interface.
- Construct a graphical user interface.
- Format text, including numbers, for output.
- Handle number format exceptions during input.

Lecture Notes

This chapter focuses on aspects of user interface programming. Examples of complex terminal I/O interactions, such as taking repeated sets of inputs and allowing the user to select options from a menu, are discussed. Students learn how to format data for structured output and also how to handle format errors in input data. They become aware that the human user is the most important part of a computer system.

Chapter 9—Introduction to Arrays

Objectives

- Write programs that handle collections of similar items.
- Declare array variables and instantiate array objects.
- Manipulate arrays with loops, including the enhanced for loop.
- Write methods to manipulate arrays.
- Create parallel arrays and two-dimensional arrays.

Lecture Notes

This chapter introduces the array as a data structure containing elements that are ordered by position. To get students thinking about these concepts, we consider how to change the Student class to maintain 20 scores rather than just the three as in Chapter 5. An array would be an effective way of doing this.

Conceptual Overview

This section provides an overview of the logical structure of an array. Important concepts are the elements contained in an array, its index positions, and its length. Figure 9-1, illustrates these ideas nicely. Focus on the syntax of the subscript operator and note that its values range from 0 to the length of the array minus 1. Stress the difference between an array element and its position or index. To keep students from getting confused about arrays, they will draw pictures like these and label the cells with elements and positions.

Chapter 11—Arrays Continued - Sorting and searching

Objectives

- Use string methods appropriately.
- Write a method for searching an array.
- Understand why a sorted array can be searched more efficiently than an unsorted array.
- Write a method to sort an array.
- Write methods to perform insertions and removals at given positions in an array.
- Understand the issues involved when working with arrays of objects.
- Perform simple operations with Java’s ArrayList class.

Lecture Notes

This chapter gives students a deeper understanding of arrays and operations on arrays. The chapter also reveals the limitations of using arrays that lead programmers to develop and use other data structures, such as array lists. Be sure to spend some time on the Marine Biology Case Study, which not only illustrates these issues, but also reviews basic principles of software design with classes.

Chapter 12—Recursion, Complexity, and Searching and Sorting

Objectives

- Design and implement a recursive method to solve a problem.
- Understand the similarities and differences between recursive and iterative solutions of a problem.
- Check and test a recursive method for correctness.
- Understand how a computer executes a recursive method.
- Perform a simple complexity analysis of an algorithm using big-O notation.
- Recognize some typical orders of complexity.
- Understand the behavior of a complex sort algorithm such as the quicksort.

Lecture Notes

This chapter introduces recursive problem solving and the use of big-O notation for measuring the complexity of algorithms. Recursive algorithms and methods provide natural solutions to many problems. Students also need to become acquainted with the cost (memory) of running algorithms on large data sets. Recursive strategies can be applied to develop efficient and elegant searching and sorting algorithms, and standard formal techniques apply.