

Boca Raton Community High School

AP Computer Science A - Syllabus 2009/10

Instructor: Ronald C. Persin

Course Resources

Java Software Solutions for AP® Computer Science, A. J. Lewis, W. Loftus, and C. Cocking, 2nd Edition, 2007, Prentice Hall.

Java Software Solutions: Foundations of Program Design, Second Edition, J. Lewis, W. Loftus, 2000, Addison-Wesley (Chapter Eight)

AP GridWorld Case Study

Website: Alice: <http://www.alice.org/>

Institute for Mathematics and Computer Science Online Courses (eIMACS): <http://www.eimacs.com>

Note: A chart is available below to show a correlation between the “Computer Science A” column of the Topic Outline in the *AP Computer Science Course Description* and each unit of this syllabus.

Nine-week evaluation: Tests 50%
Homework 10%
Computer labwork 30%
Notebook check 10%

End of 1st semester: 1st nine-week grade 40%
2nd nine-week grade 40%
Semester Exam 20%

End of 2nd semester: 3rd nine-week grade 40%
4th nine-week grade 40%
Semester Exam 20%

Units of Study:

Unit 1: Introduction to the Computer and Ready-Made Programming Environments

We begin with an introduction to using the computer and its components. Students experiment with the Alice program, which leads them to explore two ready-made programs and provides an exercise that allows them to create a program from scratch. Students also work through the first chapter of the GridWorld Case Study. They learn about the components of a computer system, networks, and the Internet.

Sample Student Activity for Unit 1: Submit a research paper (Inside the Computer)

Resources:

- Alice
- GridWorld Case Study
- Lewis, Loftus, Cocking: Chapter One

Unit 2: Introduction to the Programming Environment

This unit begins teaching students how to plan their work by learning about proper pseudocode and flowchart structures. Students are led through their first program (Hello World) to explore the format of a proper Java

program. They continue with learning how to use simple input/output, primitive data types, the string class, arithmetic expressions, and random number generation.

Sample Student Activities for Unit 2: Practice pseudocode and flowchart, Hello World, Strings, Expressions, Random/IO, Marvin's Game—Parts One and Two.

Resource:

- Lewis, Loftus, Cocking: Chapter 2

Unit 3: Conditional and Repetition

Students focus on how to use conditionals (if, if-else, switch) and repetition (for, while, do while) structures. Students learn to design and test programs that solve assigned problems.

Sample Student Activities for Unit 3: Conditionals, Loops, Code Segments, Marvin's Game—Part Three

Resources:

- Lewis, Loftus, Cocking: Chapter 3
- Search for Prime Numbers

Unit 4: Methods and Classes

Students learn how to create their own classes by defining objects. Proper method and class structure is emphasized. They are also introduced to interfaces (Comparable), inheritance, and polymorphism.

Sample Student Activities for Unit 4: Exploring Applets, Methods Assignment, Simple Animation Assignment, Marvin's Game—Part Four

Resources:

- Lewis, Loftus, Cocking: Chapters 4, 5 and 7

Unit 5: Advanced Programming Structures [C4, C5, C6]

This unit focuses on the creation of advanced data structures and algorithms. Topics include 1-D and 2-D arrays, ArrayLists, sequential and binary searching, sorting algorithms (Insertion Sort, Selection Sort, and Merge Sort) and recursion. Students write pre- and post-conditions for their methods. Students learn when to prefer iteration and when to prefer recursion. They learn the differences between arrays and ArrayLists. They compare sequential searching to binary searching and compare algorithms for sorting with one another.

Sample Student Activities for Unit 5: Arrays and ArrayList, Marvin's Game—Parts Five and Six

Resources:

- Lewis, Loftus, Cocking: Chapters 6 and 8
- Sorting Out Sorting
- Secret Messages Using the Caesarian Cipher

Unit 6: AP GridWorld Case Study, continued

Students work through Chapters Two, Three, and Four of the AP GridWorld Case Study.

Resource:

- AP GridWorld Case Study

Unit 7: The Computer and Society

Students explore how the computer has affected society. Topics include the evolution of computer languages, ergonomics, ethics, and computer careers. Students will submit a series of research papers and do a class presentation.

Resources:

- Assignment handouts

1. Evolution of Computer Languages—research paper
2. Ergonomics—research paper and class presentation
3. Computer Careers and Ethics—research paper

Unit 8: Input/Output

This unit teaches students how to use input and output streams. Topics include try and catch, exceptions, standard I/O, and reading from and writing to text files.

Resource:

- Lewis, Loftus: Chapter 8

Unit 9: Getting Ready for the AP Exam

Students work through sample multiple-choice questions and free-response questions. They will write a mock exam to simulate the timing and type of questions to expect on the actual exam. Students review Java theory and the AP GridWorld Case Study using eIMACS.

Resources:

- AP GridWorld Case Study
- eIMACS
- Supplemental handouts—multiple-choice and free-response questions

Unit 10: Summative

Students work through a series of summative assignments that evaluate their ability to work through the development of a program. Tasks include writing a program proposal, planning the program using pseudocodes/flowcharts, creating a user’s manual, writing program code, and creating a limitations document.

Additional Information:

Students work through a cumulative programming assignment (Marvin’s Game) that is part of Units 2 through 5. This program contains six parts. Each part evaluates the student’s understanding of the structures taught in each unit.

Correlation to AP Topic Outline

I. Object-Oriented Program Design

The overall goal for designing a piece of software (a computer program) is to correctly solve the given problem. At the same time, this goal should encompass specifying and designing a program that is understandable, can be adapted to changing circumstances, and has the potential to be reused in whole or in part. The design process needs to be based on a thorough understanding of the problem to be solved.

A. Program design

1. Read and understand a problem description, purpose, and goals. Units 1 and 3
2. Apply data abstraction and encapsulation. Units 2 and 4
3. Read and understand class specifications and relationships among the classes (“is-a,” “has-a” relationships). Unit 4
4. Understand and implement a given class hierarchy. Unit 4
5. Identify reusable components from existing code using classes and class libraries. Unit 4

B. Class design

1. Design and implement a class. Unit 4
2. Choose appropriate data representation and algorithms. Units 3 and 4
3. Apply functional decomposition. Unit 4
4. Extend a given class using inheritance. Unit 4

II. Program Implementation

The overall goals of program implementation parallel those of program design. Classes that fill common needs should be built so that they can be reused easily in other programs. Object-oriented design is an important part of program implementation.

- A. Implementation techniques
 - 1. Methodology
 - a. Object-oriented development Unit 2
 - b. Top-down development Unit 3
 - c. Encapsulation and information hiding Unit 4
 - d. Procedural abstraction Unit 4
- B. Programming constructs
 - 1. Primitive types vs. objects Unit 2
 - 2. Declaration
 - a. Constant declarations Unit 2
 - b. Variable declarations Unit 2
 - c. Class declarations Unit 4
 - d. Interface declarations Unit 4
 - e. Method declarations Unit 4
 - f. Parameter declarations Unit 4
 - 3. Console output (System.out.print/println) Unit 2
 - 4. Control
 - a. Methods Unit 4
 - b. Sequential Unit 3
 - c. Conditional Unit 3
 - d. Iteration Unit 3
 - e. Recursion Unit 5
- C. Java library classes (included in the A-level AP Java Subset) Units 2 and 4

III. Program Analysis

The analysis of programs includes examining and testing programs to determine whether they correctly meet their specifications. It also includes the analysis of programs or algorithms in order to understand their time and space requirements when applied to different data sets.

- A. Testing
 - 1. Test classes and libraries in isolation. Unit 4
 - 2. Identify boundary cases and generate appropriate test data. Unit 4
 - 3. Perform integration testing. Unit 4
- B. Debugging
 - 1. Categorize errors: compile-time, run-time, logic. Units 1, 2, and 8
 - 2. Identify and correct errors. Units 1, 2, 5, and 8
 - 3. Employ techniques such as using a debugger, adding extra output statements, or hand-tracing code. Units 1, 2, 5, and 8
- C. Understand and modify existing code Units 1, 2, 3, 4, 5, 6, and 8
- D. Extend existing code using inheritance Unit 4
- E. Understand error handling
 - 1. Understand runtime exceptions. Unit 4
- F. Reason about programs
 - 1. Pre- and post-conditions Unit 4
 - 2. Assertions Unit 4
- G. Analysis of algorithms
 - 1. Informal comparisons of running times Unit 5

- 2. Exact calculation of statement execution counts Unit 5
- H. Numerical representations and limits
 - 1. Representations of numbers in different bases Unit 1
 - 2. Limitations of finite representations (e.g., integer bounds, imprecision of floating-point representations, and round-off error) Units 2 and 3

IV. Standard Data Structures

Data structures are used to represent information within a program. Abstraction is an important theme in the development and application of data structures.

- A. Simple data types (int, boolean, double) Unit 2
- B. Classes Units 2 and 4
- C. One-dimensional arrays Unit 5

V. Standard Algorithms

Standard algorithms serve as examples of good solutions to standard problems. Many are intertwined with standard data structures. These algorithms provide examples for analysis of program efficiency.

- A. Operations on A-level data structures previously listed
 - 1. Traversals Unit 5
 - 2. Insertions Unit 5
 - 3. Deletions Unit 5
- B. Searching
 - 1. Sequential Unit 5
 - 2. Binary Unit 5
- C. Sorting
 - 1. Selection Unit 5
 - 2. Insertion Unit 5
 - 3. Mergesort Unit 5

VI. Computing in Context

A working knowledge of the major hardware and software components of computer systems is necessary for the study of computer science, as is the awareness of the ethical and social implications of computing systems. These topics need not be covered in detail but should be considered throughout the course.

- A. Major hardware components
 - 1. Primary and secondary memory Unit 1
 - 2. Processors Unit 1
 - 3. Peripherals Unit 1
- B. System software
 - 1. Language translators/compiler Unit 1
 - 2. Virtual machines Unit 1
 - 3. Operating systems Unit 1
- C. Types of systems
 - 1. Single-user systems Unit 1
 - 2. Networks Unit 1
- D. Responsible use of computer systems
 - 1. System reliability Unit 1
 - 2. Privacy Unit 7
 - 3. Legal issues and intellectual property Unit 7
 - 4. Social and ethical ramifications of computer use Unit 7