

AP Computer Science A

Course Design:

The proposed syllabus is for a two-semester course, assuming 30 weeks are available prior to the AP exam. The course meets for five 45-minute class periods per week. The course includes several individual programming projects assigned for one or two weeks each. The time after the AP CS Exam is devoted to a team project and enrichment activities. The course includes an optional enrichment unit on files, graphics, and GUI, which is not required for the AP exam.

Course Objectives:

- Understand and apply the main principles of object-oriented software design and programming: classes and objects, constructors, methods, instance and static variables, inheritance, class hierarchies, and polymorphism
- Learn to code fluently in Java in a well-structured fashion and in good style; learn to pay attention to code clarity and documentation
- Learn to use Java library packages and classes within the scope of the AP Java subset
- Understand the concept of an algorithm; implement algorithms in Java using conditional and iterative control structures and recursion.
- Learn to select appropriate algorithms and data structures to solve a given problem.
- Compare efficiency of alternative solutions to a given problem.
- Learn common searching and sorting algorithms: Sequential Search and Binary Search; Selection Sort, Insertion Sort, and Mergesort
- Understand one- and two-dimensional arrays, the `List` interface, and the `ArrayList` class, and use them appropriately in programming projects
- Acquire skills in designing object-oriented software solutions to problems from various application areas
- Learn the GridWorld case study and accompanying exercises and questions provided by the College Board
- Discuss ethical and social issues related to the use of computers
- Prepare for the AP exam in computer science.

Texts and Supplementary Materials:

Litvin, Maria, and Gary Litvin. *Java Methods: Object-Oriented Programming and Data Structures, 2nd AP Edition*, Andover, Mass.: [Skylight Publishing](#), 2011.

Litvin, Maria, and Gary Litvin. *Be Prepared for the AP Computer Science Exam in Java, 4th Edition*, Andover, Mass.: [Skylight Publishing](#), 2009.

The College Board's GridWorld [case study](#), Parts 1-4.

codingbat.com: <http://codingbat.com/java>.

Litvin, Maria, and Gary Litvin. *250 Multiple-Choice Computer Science Questions in Java*. Andover, Mass.: [Skylight Publishing](#), 2008.

Current media sources and Internet articles and blogs discussing ethical and social issues related to computer use.

Teacher Materials:

The College Board's Computer Science A Course Description.

The College Board's GridWorld Teacher Manual.

AP Central resources.

Java Methods student files, teacher files, Powerpoints, *Test Package*, additional resources at <http://www.skylit.com/javamethods> and <http://www.skylit.com/oop>.

Course Outline:

Chapter numbers for readings and exercises refer to *Java Methods, 2nd AP Edition*. The labs, case studies, and projects proposed below come from *Java Methods* and serve only as examples of possible assignments; the teacher's favorites may be used instead. GridWorld refers to the College Board's GridWorld case study narrative.

Unit 1: An introduction to computers and software engineering (2 weeks)**1. An Introduction to hardware, software and the Internet (Week 1; duration 1 week)**

Elements of a computer system. How information is represented in computer memory. Binary and hex number systems and ASCII / Unicode. An introduction to the Internet.

Reading and exercises: Chapter 1.

Lab: Find and explore the home pages of some Internet and World Wide Web pioneers.

2. An Introduction to Software Engineering (Week 2; duration 1 week)

Getting familiar with the software development process. Compilers and interpreters. JDK tools (*javac, java, appletviewer, javadoc*). Running a Java program in a command-line environment (optional). Using an IDE. Java classes and source files. A brief introduction to OOP.

Reading and exercises: Chapter 2.

Lab: Compile and run simple programs using command-line JDK tools or an IDE (Section 2.4).

Lab: Compile and run simple GUI applications and an applet (Section 2.6).

Unit 2: Objects, algorithms, and syntax (7 weeks)**3. A first look at objects and classes using the GridWorld context (Weeks 3-5; duration 2.5 weeks)**

Classes and objects. Classes and source files. *Case study:* GridWorld. CRC cards. Library classes and packages. The `import` statement. A first look at fields, constructors, and methods of a class. Inheritance. Extending library classes.

Reading and exercises: Chapter 3; GridWorld Student Manual Parts 1 and 2.

Lab: Set up the first GridWorld project and run `BugRunner`.

Lab: Interacting with “actors” in GridWorld (Section 3.4).

Lab: A new type of actor: `RandomBug` (Section 3.7).

Lab: `Circle` and `Cylinder` classes (Exercise 12, p. 74).

4. Algorithms (Weeks 5-6; duration 1.5 week)

The concept of an algorithm. Pseudocode and flowcharts. Iterations. Recursion. Working with lists. *Case study:* Euclid’s GCF Algorithm. (Most of the exercises for this chapter are pencil-and-paper exercises.)

Reading and exercises: Chapter 4.

Lab: Print stars using iterations and recursion (Exercise 10, p. 102).

5. Java syntax and style (Week 7; duration 1 week)

Syntax and style in a programming language. Comments. Reserved words and programmer-defined names. Statements, braces, blocks, indentation. Syntax errors, run-time errors, logic errors.

Reading and exercises: Chapter 5; Appendix A.

Lab: Correcting syntax errors and a logic error as an “adventure game” (Section 5.6).

Unit 3: Arithmetic, logic, and control statements (6 weeks)**6. Data types, variables, and arithmetic (Weeks 8-9; duration 2 weeks)**

The concepts of a variable and a data type. Declarations of variables. Fields vs. local variables. The primitive data types: `int`, `double` and `char`. Literal and symbolic constants. Initialization of variables. Scope of variables. Arithmetic expressions. Data types in arithmetic expressions. The cast operator. The compound assignment (`+=`, etc.) and increment and decrement operators (`++`, `--`). Converting numbers and objects into strings.

Reading and exercises: Chapter 6.

Lab: Exercises for Chapter 6 (for example, 16, 17, 18, pp. 153-154).

Lab: *Pie Chart* (Section 6.10).

Lab: *Rainbow* (Exercise 19, p. 155).

7. The if-else statement (Weeks 10-11; duration 2 weeks)

The if-else statement, Boolean expressions, the boolean data type, true and false values. Relational and logical operators. De Morgan's laws. Short-circuit evaluation. Nested if-else and if-else-if. *Case Study: Craps*. Elements of object-oriented design in *Craps*. The switch statement. enum data types.

Reading and exercises: Chapter 7.

Lab: Exercises for Chapter 7 (for example, 2, 11, 14-17).

Lab: The CrapsGame class for *Craps*: fill in the blanks and test in isolation (Section 7.9).

Lab: Finishing and testing the *Craps* program (Section 7.12).

Extra: codingbat.com *Logic-1* and *Logic-2*.

8. Iterative statements (Weeks 12-13; duration 2 weeks)

while, for, and do-while loops. break and return in loops.

Reading and exercises: Chapter 8.

Lab: Exercises for Chapter 8 (for example, 1 - 3, p. 212).

Lab: *Perfect Numbers* (Section 8.6).

Unit 4: Classes and class hierarchies (7 weeks)**9. Details of defining classes and using objects (Weeks 14-16; duration 2.5 weeks)**

Public and private fields and methods. Constructors and the new operator. References to objects. Calling methods and accessing fields. Passing parameters to constructors and methods. return statement. Overloaded methods. Static variables and methods.

Reading and exercises: Chapter 9.

Lab: *Snack Bar*.

Lab: GridWorld exercises for Part 2 (page 12).

Lab: *Snack Bar Continued*.

10. Strings (Weeks 16-17; duration 1.5 week)

String objects. Literal strings. Immutability. String methods. Converting strings into numbers and numbers into strings. The Character class and its methods.

Reading and exercises: Chapter 10.

Lab: *Lipograms* (Section 10.8).

Extra: codingbat.com *String-1*, *String-2*, *String-3*.

11. Class hierarchies, abstract classes, and interfaces (Weeks 18-20; duration 3 weeks)

Class hierarchies. Abstract classes. Invoking superclass's constructors and calling superclass's methods. Polymorphism. Interfaces.

Reading and exercises: Chapter 11, GridWorld Part 3.

Lab: *A GridWorld Dance* (Section 11.6).

Lab: Past AP free-response questions on class hierarchies and polymorphism.

Lab: Creating a subclass of `Actor`, GridWorld Part 3 group activity (GridWorld Student Manual p. 24).

Unit 5: Arrays, the List interface, the ArrayList class, searching and sorting (8 weeks)**12. One- and Two-Dimensional Arrays (Weeks 21-22; duration 2 weeks)**

One-dimensional arrays. Arrays as objects. Declaring and initializing. Indices. Length. `IndexOutOfBoundsException`. Two-dimensional arrays. Accessing the number of rows and columns. Traversals and the “for-each” loop. Inserting and removing elements.

Reading and exercises: Chapter 12.

Lab: *Fortune Teller* (Section 12.3).

Lab: Past free-response questions on arrays.

Lab: *Chomp* (Section 12.5).

Extra: codingbat *Arrays-1, Arrays-2, Arrays-3*.

13. ArrayLists and GridWorld Critters (Weeks 23-24; duration 2 weeks)

`ArrayList`'s structure. The `List` interface. `ArrayList`'s constructors and methods. Pitfalls. `ArrayLists` in GridWorld.

Reading and exercises: Chapter 13, GridWorld Student manual Part 4.

Lab: *Creating an Index for a Document* (Section 13.5).

Lab: Past AP free-response questions on `ArrayList`.

Lab: GridWorld Part 4 exercises (GridWorld Student Manual, p. 32).

Lab: GridWorld Critters (Section 13.6).

14. Searching and sorting. Introduction to analysis of algorithms. (Weeks 25-26; duration 2 weeks)

Comparing objects. The `equals` method and the `Comparable` interface. Sequential and Binary Search. Selection Sort, Insertion Sort, and Mergesort. The `java.util.Random` class. The number of comparisons required in Sequential and Binary Search. Comparison of efficiency of “quadratic” sorting algorithms (Selection Sort and Insertion Sort) vs. Mergesort.

Reading and exercises: Chapter 14.

Lab: Chapter 14 exercises (for example, 4, 9 pp. 408-409).

Lab: *Keeping Things in order* (Section 14.4).

Lab: *Benchmarks* (Section 14.9) — compares efficiency of several sorting algorithms.

Unit 6: Enrichment (optional, duration varies)**15. Streams and files**

Text and binary files. Streams vs. random-access files. Java I/O package. The Scanner class. Checked exceptions.

Reading and exercises: Chapter 15.

Lab: *Choosing Words* (Section 15.5).

Lab: Exercises and projects from the Test Package for Chapter 15.

16. Graphics and GUI

Computer graphics concepts. The Java Graphics class. GUI components and their events. Layouts. Handling mouse and keyboard events.

Reading and exercises: Chapters 16, 17, 18.

Lab: *Pieces of the Puzzle* (Section 16.7).

Programming project: *Ramblecs* (Section 17.6).

Programming project: *Drawing Editor* (Section 18.4).

Unit 7: Review (4 weeks)**17. GridWorld review (Weeks 27-28; duration 1.5 weeks)**

Review of the GridWorld classes and interfaces. Modifications and exercises.

Reading and exercises: GridWorld Parts 1-4; *Be Prepared* Chapter 6.

Labs: *GridWorld Enhancements* (from suggested exercises for Part 4, p. 32, and *Be Prepared*).

18. Review and practice for the AP exam (Weeks 28-30; duration 2.5 weeks)

Reading: *Be Prepared* Chapters 1-5; *Be Prepared* Chapter 7 (past free-response questions and solutions), *Be Prepared* practice exams, *250 Multiple-Choice Computer Science Questions*.

Unit 8: After the AP exam (Duration varies)

Student presentations and debates on ethical and social issues related to the use of computers, Internet, based on the readings from the print media and the Internet.

Other suggested activities: a team project to implement a game (for example, the Game of SET, <http://www.skylit.com/ooop> or a project based on GridWorld); a potentially useful project for the school.