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 a number of individual programming projects assigned for one week each. The time after the AP CS Exam is devoted to a team project and enrichment activities.

The course is based on numerous problem solving exercises, labs, and case studies, which require students to design and implement Java classes. [CR1] The course requires 40-50 hours of hands-on work in a computer lab.[CR6]

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
- Discuss ethical and social issues related to the use of computers
- Prepare for the AP Computer Science A exam; meet all of the curricular requirements defined by the College Board for this course.

.

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*, *6th Edition*, Andover, Mass.: Skylight Publishing, 2014.

The College Board's *Magpie*, *Picture*, and *Elevens* Labs Student Guides.

CodingBat: 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 *Magpie*, *Picture*, and *Elevens* Labs Teacher Guides.

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/projects/.

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.

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 (Hello World, Greetings) 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 (Weeks 3-5; duration 2.5 weeks)

Classes and objects. Classes and source files. CRC cards. Library classes and packages. The `import` statement. A first look at fields, constructors, and methods of a class. Inheritance.

Reading and exercises: Chapter 3 and *Elevens* Lab Student Guide, Activity 1
Lab: Design and implement `Book` class (Exercise 11, p. 74). [CR1, CR6]
Lab: Design and implement `Circle` and `Cylinder` classes (Exercise 12, p. 74). [CR1, CR6]
Lab: Set up a project for the *Elevens* Lab and run the program [CR6]
Lab: *Elevens*, Activity 1, Design and create a `Card` class. [CR1, CR6]

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). [CR3, CR6]

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). [CR6]

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. [CR5 (toString)]

Reading and exercises: Chapter 6.
Lab: Exercises for Chapter 6 (for example, 16, 17, 18, pp. 153-154). [CR1, CR6]
Lab: *Pie Chart* (Section 6.10). [CR1, CR6]
Lab: *Rainbow* (Exercise 19, p. 155). [CR1, CR6]

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 `Die` [CR5 (random)] and `CrapsGame` classes for *Craps*: fill in the blanks and test in isolation (Section 7.9). [CR1, CR5, CR6]
Lab: Finishing and testing the *Craps* program (Section 7.12). [CR1, CR6]
Extra: `codingbat.com` *Logic-1* and *Logic-2*. [CR6]

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). [CR6]
Lab: *Perfect Numbers* (Section 8.6). [CR1, CR6]

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. Math methods. [CR5]

Reading and exercises: Chapter 9.
Lab: *Snack Bar* (Section 9.9). [CR1, CR6]
Lab: *Snack Bar Continued* (Section 9.12). [CR1, CR6]

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

`String` objects. Literal strings. Immutability. `String` methods. [CR5] Converting strings into numbers and numbers into strings. The `Character` class and its methods.

Reading and exercises: Chapter 10.
Lab: *Magpie*, Activity 1. [CR1, CR6]
Lab: *Lipograms* (Section 10.8). [CR1, CR6]
Extra: `codingbat.com` *String-1*, *String-2*, *String-3*. [CR6]

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.
Lab: *Baker's Dozen* (*Be Prepared*, Practice Exam 1, Question 2). [CR1, CR4, CR6]
Lab: *ChatBots* (*Be Prepared*, Practice Exam 4, Question 3). [CR1, CR4, CR6]
Lab: Past AP free-response questions on class hierarchies and polymorphism. [CR1, CR4]

Unit 5: Arrays, the `List` interface, the `ArrayList` class, searching and sorting, recursion (9 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). [CR1, CR6]
Lab: Past free-response questions on arrays. [CR1, CR6]
Lab: *Chomp* (Section 12.5).
Lab: *Picture Lab*, Activity 9, Collage, or Activity E2, Chromakey. [CR1, CR2b, CR6]
Extra: `codingbat.com`: *Arrays-1*, *Arrays-2*, *Arrays-3*. [CR2b]

13. `ArrayList` (Weeks 23-24; duration 2 weeks)

`ArrayList` structure. The `List` interface. `ArrayList`'s constructors and methods. Pitfalls. `ArrayList` vs. built-in arrays. [CR3]

Reading and exercises: Chapter 13.
Be Prepared, Section 2.5.
Lab: *Creating an Index for a Document* (Section 13.5). [CR2b, CR4]
Lab: Past AP free-response questions on `ArrayList`.
Lab: ECG Analysis (*Be Prepared*, Practice Exam 3, Question 1). [CR1, CR2b, CR6]

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 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). [CR1, CR2a, CR6]
Lab: *Benchmarks* (Section 14.9) — compares efficiency of several sorting algorithms. [CR2a]

15. Recursion revisited. (Week 27; duration 1 week)

More examples of recursion. When not to use recursion. [CR3] Understanding and debugging recursive methods.

Reading and exercises: Chapter 23.
Lab: Chapter 23 exercises (for example, 1-7, 9 pp. 571-573).
Lab: The *Tower Of Hanoi* (Section 23.5). [CR1, CR6]

Unit 6: Enrichment (optional, duration varies)

16. 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.

17. 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 (3 weeks)

18. Review and practice for the AP exam (Weeks 28-30; duration 3 weeks)

Quick reference (library classes and methods). [CR5] Past multiple-choice and free-response questions.

Reading: Be Prepared Chapters 1-5; *Be Prepared* Chapter 6 (past free-response questions and solutions), *Be Prepared* practice exams 1-5, *250 Multiple-Choice Computer Science Questions*.

Unit 8: After the AP Exam (Duration varies)

Student papers, presentations, and debates on ethical and social issues related to the use of computers, Internet, based on the readings from the current print media and the Internet. [CR7]

Reading: Java Methods Chapter 28 (`http://www.skylit.com/javamethods`): Computing in Context: Creative, Responsible, and Ethical Computer Use.

Other suggested activities: a team project to implement a game (for example, the Game of SET, `http://www.skylit.com/projects/` or the *Elevens* lab); or a potentially useful project for the school. [CR1, CR4, CR6]