AP Computer Science A Syllabus
Based on the Course and Exam Description Effective Fall 2019

Course Design

The proposed syllabus is for a two-semester course, assuming 30-32 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. The course requires 40-50 hours of hands-on work in a computer lab.

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 and the `ArrayList` class, and use them appropriately in programming projects
- 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

Maria Litvin and Gary Litvin. *Java Methods: Object-Oriented Programming and Data Structures, 3rd AP Edition*, Skylight Publishing, 2015.

Maria Litvin and Gary Litvin. *Be Prepared for the AP Computer Science Exam in Java*, *7th Edition*, Skylight Publishing, 2019.

The College Board's Exemplar Labs Student Guides.

CodingBat, codingbat.com/java
Practice It!, practiceit.cs.washington.edu/.

Maria Litvin and Gary Litvin. *250 Multiple-Choice Computer Science Questions in Java*, Skylight Publishing, 2008.

Maria Litvin and Gary Litvin. Annotated Solutions to Free-Response Questions from Past Exams 2004-Present, www.skylit.com/beprepared/fr.html.

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

Teacher Materials

The College Board's 2019 Computer Science A Course and Exam Description (CED).

AP Classroom (Topic Questions, Progress Checks) and other *AP Central* resources.

The College Board's Exemplar Labs Teacher Guides.

Maria Litvin and Gary Litvin. *Java Methods Test Package*.

Java Methods student files, teacher files, powerpoints, additional resources at www.skylit.com/javamethods and www.skylit.com/projects/.

Course Outline

Chapter numbers for readings and exercises refer to *Java Methods, 3rd AP Edition*. Numbers in brackets, such as [1.1], refer to CED topics.

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.

Module 1: An introduction to computers and software engineering (2 weeks)

1. 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.
Lab: *Picture Lab*, Activities 1 and 2.

2. An introduction to software engineering (Week 2; duration 1 week)

Getting familiar with the software development process. Compilers and interpreters. [1.1] JDK tools (*javac*, *java*, *javadoc*). Using an IDE. Java classes and source files. A brief introduction to OOP. Software engineer's Code of Ethics. [5.10]

Reading and exercises: Chapter 2 and Section 28.3
Lab: Compile and run simple programs (Hello World, Greetings) using an IDE (Section 2.4). [1.1]
Lab: Compile and run simple GUI applications (Section 2.6, optional).

Module 2: Syntax and an introduction to objects (3 weeks)

3. Java syntax and style (Week 3; 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 3; Appendix A.
Lab: Correcting syntax errors and a logic error as an "adventure game" (Section 3.7). [5.3]

4. A first look at objects and classes (Weeks 4-5; duration 2 weeks)

Classes and objects. [2.1] Classes and source files. Library classes and packages. The `import` statement. A first look at fields, constructors, and methods of a class. [2.2 - 2.5]

Reading and exercises: Chapter 4 (except Section 4.5).
Lab: Design and implement `Circle` and `Cylinder` classes (Exercise 8, p. 92). [2.2, 2.5]
Case study: BalloonDraw (Section 4.2). [2.2]
Lab: *TestBalloon* class (Section 4.4, Page 80). [2.1, 2.4]
Lab: *Elevens* Lab, Activity 1

Module 3: Arithmetic, logic, and control statements (7 weeks)

5. Data types, variables, and arithmetic (Weeks 6-7; duration 2 weeks)

The concepts of a variable and a data type. [1.2] Declarations of variables. Fields vs. local variables. The primitive data types: `int`, `double` [1.2] and `char`. Literal and symbolic constants. [2.6] Initialization of variables. Scope of variables. Arithmetic expressions. [1.3] Data types in arithmetic expressions. The cast operator. [1.5] The compound assignment (+=, etc.) and increment and decrement operators (++, --). [1.4] Converting numbers and objects into strings. [2.8] `Math` class methods (`abs`, `sqrt`, `pow`, `random`). [2.9]

Reading and exercises: Chapter 5.
Lab: Exercises for Chapter 5. [1.2-1.5]
Lab: *Pie Chart* (Section 5.11). [1.2-1.5]
Lab: *Rainbow* (Exercise 27 p. 130). [1.2-1.5]
Lab: Practice-It!, Chapter 2.

6. The `if-else` statement (Weeks 8-9; duration 2 weeks)

The `if` and `if-else` statement. [3.2-3.3] Boolean expressions, the `boolean` data type, `true` and `false` values. [3.1] Relational and logical operators. De Morgan's laws. [3.6] Short-circuit evaluation. [3.5] Nested `if-else` and `if-else-if`. [3.4] *Case Study: Craps*. Elements of object-oriented design in *Craps*. The `switch` statement.

Reading and exercises: Chapter 6 (Section 6.11 is optional).
Lab: Exercises for Chapter 6 (for example, 2-5, 10-12).
Lab: The `Die` [2.1-2.2, 2.9 (random)] and `CrapsGame` classes for *Craps*: fill in the blanks and test in isolation (Section 6.9).
Lab: Finishing and testing the *Craps* program (Section 6.12). [2.1-2.2]
Lab: `codingbat.com` *Logic-1* and *Logic-2*.

7. Algorithms and iterations (Weeks 10-12; duration 3 weeks)

The concept of an algorithm. Properties of algorithms. [4.5] Iterations. `while` and `for` loops. [4.1-4.2] `break` and `return` in loops. Nested loops. [4.4]

Reading and exercises: Chapter 7.
Lab: Exercises for Chapter 7.
Case study and lab: Euclid's GCF algorithm (Section 7.7 and Exercise 26 on p. 206). [4.1]
Lab: *Perfect Numbers* (Section 7.8). [4.5]

Interlude: Ethical and social implications of computer use (Week 13)

Student papers, presentations, and debates on ethical, social, and privacy issues related to the use of computers and the Internet. [5.10, 7.7]

Reading: Sections 28.3 - 28.5; current news and commentary in the online media.

Module 4: Strings and arrays (5 weeks)

8. Strings (Week 14-15; duration 1.5 week)

`String` objects. [2.6] Literal strings. [2.6] Immutability. `String` methods. [2.7] Converting strings into numbers and numbers into strings. [2.8] The `Character` class and its methods.

Reading and exercises: Chapter 8.
Lab: *Magpie*, Activities 1 and 2. [4.3]
Lab: *Consumer Review* [4.3]
Lab: *Lipograms* (Section 8.8). [4.3]
Lab: `codingbat.com` *String-1*, *String-2*, *String-3*.

9. One-dimensional arrays (Weeks 15-17; duration 2 weeks)

One-dimensional arrays. [6.1] Arrays as objects. Declaring and initializing. [6.1] Indices. Length. `ArrayIndexOutOfBoundsException`. Traversals and the "for-each" loop. [6.2, 6.3] Inserting and removing elements. [6.4]

Reading and exercises: Chapter 9.
Lab: *Fortune Teller* (Section 9.3).
Lab: *Magpie*, Activity 5.
Lab: Past free-response questions on arrays.
Case study and lab: The Sieve or Eratosthenes (Section 9.8). [4.2, 6.2, 6.4]
Lab: `codingbat.com`, *Arrays-1*, *Arrays-2*.

10. Two-dimensional arrays (Weeks 17-18; duration 1.5 weeks)

Declaring and initializing two-dimensional arrays. [8.1] Accessing the number of rows and columns. Traversals and nested "for-each" loops. [8.2]

Reading and exercises: Chapter 9.
Lab: Past free-response questions on 2D arrays. [8.1-8.2]
Lab: *Chomp* (Section 9.5).

Module 5: Classes and class hierarchies (6 weeks)

11. Details of defining classes and using objects (Weeks 19-20; duration 2 weeks)

Public and private fields and methods. [5.8] Constructors and the `new` operator. [5.2] References to objects. [2.2] `this` keyword. [5.9] Calling methods and accessing fields. [5.4-5.5] Passing parameters to constructors and methods. [2.4, 2.5] `return` statement. Overloaded methods. Static variables and methods. [5.7]

Reading and exercises: Chapter 10.
Case study: the `Fraction` class (Sections 10.1 - 10.8). [5.1]
Case study and lab: Snack Bar (Section 10.9). [5.6]
Lab: *Snack Bar Continued* (Section 10.12). [5.1, 5.7]

12. `ArrayList` (Weeks 21-22; duration 2 weeks)

The `ArrayList` class. [7.1] The `List` interface (optional). `ArrayList`'s constructors and methods. [7.2] Pitfalls. [7.4] `ArrayList` vs. built-in arrays.

Reading and exercises: Chapter 11.
Be Prepared, Section 2.6.
Lab: *Shuffler* (Section 11.4). [7.3-7.4]
Lab: Creating an index for a document — using `ArrayList` and writing a subclass of `ArrayList` (Section 11.6). [7.3-7.4]
Lab: Past AP free-response questions on `ArrayList`. [7.3-7.4]
Lab: *ECG Analysis* (*Be Prepared*, Practice Exam 4, Question 3). [7.2-7.4]
Lab: Practice-It!, Chapter 10

13. Class hierarchies, abstract classes, and interfaces (Weeks 23-24; duration 2 weeks)

Class hierarchies. [9.1] The IS-A relationship between objects. [9.5] The `Object` class. [9.7] Abstract classes (optional). Invoking superclass's constructors and calling superclass's methods. [9.2, 9.4] Polymorphism. [9.6] Interfaces (optional).

Reading and exercises: Section 4.5, Chapter 12.
Lab: Inflatable Balloon [9.1-9.6]
Lab: *Apple Purchase with Pie* (*Be Prepared*, Practice Exam 5, Question 2 lab) — writing a subclass and overriding methods. [9.3]
Lab: Past AP free-response questions on class hierarchies and polymorphism. [9.1-9.6]
Case study and lab: Balloons of All Kinds (extend the `Balloon` class, coding constructors and overriding methods (Section 4.6). [9.1-9.6]

Module 6: Recursion, searching and sorting (4 weeks)

14. Recursion (Week 25; duration 1 week)

Recursive methods. Base case. Understanding and debugging recursive methods. When not to use recursion. [10.1]

Reading and exercises: Chapter 13 and Sections 23.3 - 23.5.
Lab: Chapter 13 exercises. [10.1]
Lab: The *Tower of Hanoi* (Section 23.5). [10.1]

15. Searching and sorting. Introduction to analysis of algorithms. (Weeks 26-28; duration 3 weeks)

Comparing objects. [3.7] The `equals` method. The `Comparable` interface (optional). Sequential and Binary Search. [7.5, 10.2] The number of comparisons required in Sequential and Binary Search. [4.5] Selection Sort, Insertion Sort, and Mergesort. [7.6, 10.2] Comparison of efficiency of "quadratic" sorting algorithms (Selection Sort and Insertion Sort) vs. Mergesort. [10.2]

Reading and exercises: Chapter 14.
Lab: Chapter 14 exercises.
Lab: *Keeping Things in Order* (Section 14.4).
Lab: *Benchmarks* (Section 14.9) — compares efficiency of several sorting algorithms. [7.6, 10.2]

Module 7: Review (3 weeks)

16. Review and practice for the AP exam (Weeks 29-31; duration 3 weeks)

Quick reference (library classes and methods). 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 in Java*.

Module 8: After the exam, enrichment (optional, duration varies)

17. 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 exercises and the Test Package for Chapter 15.

18. Graphics and GUI

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

Reading and exercises: Chapters 16, 17, 18.
Lab: Pieces of the Puzzle (Section 16.7).
Programming project: Ramblecs (Section 17.6).
Lab: Slide Show (Section 18.7).

19. Projects that demonstrate creative computer use.

Reading and exercises: Java Methods Chapter 28, "Computing in Context: Creative, Responsible, and Ethical Computer Use", Section 28.2.
Other suggested activities: a team project to implement a game (for example, the Game of SET, www.skylit.com/projects/ or the *Elevens* lab); or a potentially useful project for the school or community.