

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.

Prerequisite:

Introduction to Computer Science as described in the Introductory Course syllabus.

Introductory Course includes a study of hardware and software components of a computer system and the roles of these components within the system.

Reading and exercises: Chapter 1: *An Introduction of Hardware, Software and the Internet*, from *Java Methods A&AB: Object-Oriented Programming and Data Structures, AP Edition*, Andover, Mass., Litvin, Maria, and Gary Litvin, [Skylight Publishing](#), 2006.

Assignment: Research the evolution of the computer industry.

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 A&AB: Object-Oriented Programming and Data Structures, AP Edition*, Andover, Mass.: [Skylight Publishing](#), 2006.

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.

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

Current magazine and Internet articles 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 Disk, Teacher Disk, PowerPoint slides, 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 A&AB, 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: Review of the introductory material (4 weeks)**1. Java syntax and style (Weeks 1-2; duration 2 weeks)**

Import statements and library classes. Comments. Indentation and braces. Primitive data types. Declaring fields and local variables. Arithmetic operators. Boolean expressions, relational and logical operators.

Reading and exercises: Chapters 5-7, Appendix A.

Labs: Exercises and/or free-response questions from the Test Package for Chapters 5-7.

2. Review of algorithms (Week 3; duration 1 week)

Algorithms using iterations and recursion. When not to use recursion (e.g., Fibonacci Numbers).

Reading and exercises: Chapters 4 and 8.

Labs: Exercises and/or free-response questions from the Test Package for Chapters 4 and 8.

3. Review of classes and objects (Week 4; duration 1 week)

Classes and objects. Variables, constructors and methods. An introduction to inheritance. *First Steps* case study review.

Reading and exercises: Chapter 3.

Lab: “Bystander” in *First Steps* (Exercise 3-12).

Unit 2: Classes, class hierarchies, GridWorld (7 weeks)

4. An introduction to GridWorld (Week 5; duration 1 week)

Experimenting with the GridWorld GUI. An overview of the classes and objects involved. Role-play exercise.

Reading and exercises: GridWorld Part 1.

Lab: Set up a GridWorld project and run [BugRunner](#).

Lab: Simple extensions of the [Bug](#) class.

5. Details of defining classes and using objects (Weeks 6-7; duration 2 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. GridWorld continued.

Reading and exercises: Chapter 9; GridWorld Part 2.

Lab: *Snack Bar*.

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

6. Strings (Week 8; duration 1 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).

7. Class hierarchies, abstract classes, and interfaces (Weeks 9-11; duration 3 weeks)

Class hierarchies. Polymorphism. Abstract classes. Invoking superclass’s constructors and calling superclass’s methods. Case Study: *Dance Studio*. Interfaces.

Reading and exercises: Chapter 11, GridWorld Part 3.

Lab: *Dance Studio*.

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

Lab: Creating a subclass of [Actor](#), GridWorld Part 3 group activity (p. 24).

Unit 3: Arrays, the [List](#) interface, [ArrayLists](#), searching and sorting (8 weeks)**8. Arrays (Week 12; duration 1 week)**

One-dimensional arrays. Arrays as objects. Declaring and initializing. Indices. Length. [IndexOutOfBoundsException](#).

Reading and exercises: Sections 12.1-12.3.

Lab: *Fortune Teller* (Section 12.3).

9. [ArrayLists](#); algorithms for arrays and [ArrayLists](#) (Weeks 13-14; duration 2 weeks)

The [List](#) interface. [ArrayList](#)'s constructors and methods. Traversals and the “for each” loop. Finding the largest and the smallest element. Inserting and removing elements. When to use arrays and when — [ArrayLists](#). [ArrayLists](#) in GridWorld.

Reading and exercises: Sections 12.4-12.9, GridWorld Part 4.

Lab: *Creating an Index for a Document* (Section 12.9).

Lab: Past free-response questions on arrays and [ArrayList](#).

Lab: GridWorld Part 4 exercises (p. 32).

10. Two-dimensional arrays (Weeks 15-16; duration 2 weeks)

Declaring 2-D arrays. Indices. Accessing the number of rows and columns. Traversals.

Reading and exercises: Sections 12.10-12.11.

Lab: *Chomp* (Section 12.11).

11. Searching and sorting. Introduction to analysis of algorithms. (Weeks 17-19; duration 3 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 13.

Lab: *Chapter 13 exercises (e.g., 13-4, 13-9)*.

Lab: *Keeping Things in order* (Section 13.4).

Lab: *Benchmarks* (Section 13.9) — compares efficiency of several sorting algorithms.

Unit 4: Enrichment (optional, 5 weeks)**12. Streams and files (Weeks 20-21; duration 2 weeks)**

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

Reading and exercises: Chapter 14.

Lab: *Dictionary for Ramblecs* (Section 14.5).

Lab: Exercises and projects from the Test Package for Chapter 14.

13. Graphics and GUI (Weeks 22-24; duration 3 weeks)

Computer graphics concepts. The Java [Graphics](#) class. GUI components and their events. Layouts. Handling mouse and keyboard events.

Reading and exercises: Chapters 15, 16, 17.

Lab: *Pieces of the Puzzle* (Section 15.7).

Programming project: *Ramblecs* (Section 16.6).

Programming project: *Drawing Editor* (Section 17.4).

Unit 5: Review (6 weeks)**14. GridWorld review (Weeks 25-27; duration 3 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*).

15. Review and practice for the exam (Weeks 28-30; duration 3 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 6: After the AP exam (Duration varies)

Suggested activities: a team project to implement a game (for example, the Game of SET, <http://www.skylit.com/ooop> or Battleship or a project based on GridWorld); a potentially useful project for the school; student presentations on ethical and social issues related to the use of computers (Appendix F).