AP Computer Science A Syllabus
Based on the Course and Exam Description Effective Fall 2025

Curricular Requirements

CR1	*Students and teachers have access to a college-level computer science textbook or resource in print or electronic format.*
CR2	*The course provides opportunities to develop student understanding of the required content outlined in each unit described in the AP Course and Exam Description (CED).*
CR3	*The course provides opportunities for students to develop the skills related to Computational Thinking Practice 1: Design Code, as outlined in the AP Course and Exam Description (CED.*
CR4	*The course provides opportunities for students to develop the skills related to Computational Thinking Practice 2: Develop Code, as outlined in the AP Course and Exam Description (CED).*
CR5	*The course provides opportunities for students to develop the skills related to Computational Thinking Practice 3: Analyze Code, as outlined in the AP Course and Exam Description (CED).*
CR6	*The course provides opportunities for students to develop the skills related to Computational Thinking Practice 4: Document code and computing systems, outlined in the AP Course and Exam Description (CED).*
CR7	*The course provides opportunities for students to develop the skills related to Computational Thinking Practice 5: Use computers responsibly, as outlined in the AP Course and Exam Description (CED).*
CR8	*The course provides students with hands-on lab experience to practice programming through designing and implementing computer-based solutions to problems.*

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. 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 code. The course requires that students spend over 40 hours of in-class time on labs. [CR8]

Course Objectives

- Understand and apply the main principles of software design and programming: classes and objects, constructors, methods, instance and static variables
- 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; understand procedural and data abstraction
- Learn to select an appropriate algorithm and data structures to solve a given problem
- Informally compare efficiency of alternative solutions to a given problem
- Understand one- and two-dimensional arrays and the `ArrayList` class, and use them appropriately in programming projects
- Learn common searching and sorting algorithms: Sequential Search and Binary Search; Selection Sort, Insertion Sort, and Mergesort
- Discuss ethical and social issues related to the responsible use of computers
- Discuss the role of AI in society and acceptable generative AI use in the AP Computer Science A course
- 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, 4th AP Edition*, Skylight Publishing, 2022. [CR1]

Maria Litvin and Gary Litvin. *Be Prepared for the AP Computer Science Exam in Java*, *9th Edition*, Skylight Publishing, 2026. [CR2]

AP Classroom (videos, Topic Questions, Progress Checks), Bluebook app, and other *AP* resources.

CodingBat, codingbat.com/java

CodeStepByStep, https://codestepbystep.com/

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 responsible computer use and AI.

Teacher Materials

The College Board's 2025 Computer Science A Course and Exam Description (CED). [CR2]

The College Board's sample labs in AP Classroom, with teacher solutions.

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

References for readings and exercises, such as JM Ch.1 or JM 21.1 refer to *Java Methods, 4th AP Edition* chapter and section numbers. Numbers in brackets, such as [CR2, Skill 1.A], refer to Curricular Requirements and Computational Thinking Skills.

The labs, case studies, and projects proposed below mostly 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)

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. [CR7]

Reading and exercises: JM Ch.1.
Project: Find and explore the home pages of several Internet and World Wide Web pioneers. [CR7, Skill 5.A]

2. An introduction to software engineering (Week 2, *CED Topic 1.1*)

Getting familiar with the software development process. Compilers and interpreters. JDK tools (*javac*, *java*, *javadoc*). Using an IDE. Java classes and source files. Software engineer's Code of Ethics. [CR7, Skill 5.A]

Reading and exercises: JM Ch.2 and 21.3. [CR7, Skill 5.A]
Lab #**1**: Compile and run simple programs (Hello World, Greetings) using an IDE (JM 2.4).
Lab #**2**: Compile and run simple GUI applications (JM 2.6, optional).

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

3. Java syntax and style (Week 3, *CED Topic 1.8*)

Syntax and style in a programming language. Comments. [CR6, Skill 4.A] Reserved words and programmer-defined names. Statements, braces, blocks, indentation. Syntax errors, logic errors, run-time errors. [Skill 3.D]

Reading and exercises: JM Ch.3; Appendix A.
Lab #**3**: Correcting syntax errors and a logic error as an "adventure game" (JM 3.7). [CR5, Skill 3.D]

4. A first look at objects and classes (Weeks 4-5, *CED Topic 1.7 and Unit 3 all topics*)

Classes and objects. Classes and source files. Library classes and packages. The `import` statement. A first look at instance variables, constructors, and methods of a class.

Reading and exercises: JM Ch.4 (except 4.5).
Lab #**4**: Design and implement `Circle` and `Cylinder` classes (JM Exercise 8). [Skills 2.B, 2.C]
Case study: BalloonDraw (JM 4.2). [CR4, CR5, Skills 2.B, 2.C]
Lab #**5**: *TestBalloon* class (JM 4.4). [CR4, CR5, Skill 2.C]

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

5. Data types, variables, and arithmetic (Weeks 6-7, *CED Topics 1.2-1.6, 1.11*)

The concepts of a variable and a data type. Declarations of variables. Instance variables vs. local variables. The primitive data types: `int`, `double`, and `char` (optional). 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. `Math` class methods (`abs`, `sqrt`, `pow`, `random`).

Reading and exercises: JM Ch.5.
Lab #**6**: Exercises for JM Ch.5. [Skills 2.A, 2.B, 2.C]
Lab #**7**: *Pie Chart* (JM 5.11). [Skills 1.A, 2.A, 2.B, 2.C]
Lab #**8**: *Rainbow* (Exercise 27). [Skills 3.B, 3.C, 3.D]

6. The `if-else` statement (Weeks 8-9, *CED Topics 2.1-2.6*)

The `if` and `if-else` statement. Boolean expressions, the `boolean` data type, `true` and `false` values. Relational and logical operators. Truth tables. De Morgan's Laws. Short-circuit evaluation. Nested `if-else` and `if-else-if`. *Case Study: Craps*. Elements of software design in *Craps*. [CR8, CR1, CR2, CR3, Skills 1.A, 2.A, 2.B, 2.C] The `switch` statement.

Reading and exercises: JM Ch.6 (6.11 is optional).
Lab #**9**: Exercises for JM Ch.6 (for example, 2-5, 10-12).
Lab #**10**: The `Die` and `CrapsGame` classes for *Craps*: fill in the blanks and test in isolation (JM 6.9). [CR1, CR2, CR3, Skills 1.A, 2.A, 2.B, 2.C]
Lab #**11**: Finishing and testing the *Craps* program (JM 6.12). [CR5, Skills 3.A, 3.B, 3.C]
Lab #**12**: `codingbat.com` *Logic-1* and *Logic-2*.

7. Algorithms and iterations (Weeks 10-12, *CED Topics 2.7-2.12*)

The concept of an algorithm. Properties of algorithms. Iterations. `while` and `for` loops. `break` and `return` in loops. Nested loops.

Reading and exercises: JM Ch.7.
Lab #**13**: Exercises for JM Ch.7.
Case study and Lab #**14**: Euclid's GCF algorithm (JM 7.7 and Exercise 26). [Skill 3.C]
Lab #**15**: *Perfect Numbers* (JM 7.8). [Skills 1.A, 1.B]

Interlude: Ethical and social implications of computer use and AI [CR7, Skill 5.A] (Week 13, *CED Topic 4.1*)

Student papers, presentations, and debates on AI and ethical, social, and privacy issues related to the responsible use of computers, the Internet, and generative AI.

Reading: JM 21.3-21.5; current news and commentary in the online media. [CR7, Skill 5.A]

Module 4: Strings and arrays (5 weeks)

8. Strings (Weeks 14-15, *CED Topic 1.15*)

`String` objects. Literal strings. Immutability. `String` methods. Converting strings into numbers and numbers into strings. The `Character` class and its methods (optional).

Reading and exercises: JM Ch.8.
Lab #**16**: *Magpie 2.0* (available in AP Classroom), Activities 1 and 2. **[CR8, Skills 2.A, 2B]**
Lab #**17**: *Lipograms* (JM 8.8). **[CR8, Skills 1.A, 1.B]**
Lab #**18**: `codingbat.com` *String-1*, *String-2*.

9. One-dimensional arrays (Weeks 15-17, *CED Topics 4.2-4.5*)

One-dimensional arrays. Arrays as objects. Declaring and initializing. Indices. Length. `ArrayIndexOutOfBoundsException`. Traversals and the for-each loop. Inserting and removing elements and other standard algorithms.

Reading and exercises: JM Ch.9.
Lab #**19**: *Fortune Teller* (JM 9.3). **[CR8]**
Lab #**20**: *Magpie 2.0*, Activity 5.
Lab #**21**: Past free-response questions on arrays.
Case study and Lab #**22**: *The Sieve or Eratosthenes* (JM 9.8).**[CR8]**
Lab #**23**: `codingbat.com`, *Arrays-1*, *Arrays-2*.

10. Two-dimensional arrays (Week 18, *CED Topics 4.11-4.13*)

Declaring and initializing two-dimensional arrays. Accessing the number of rows and columns. Traversals and nested for-each loops. Standard algorithms for 2D arrays.

Reading and exercises: JM Ch.9.
Lab #**24**: Past free-response questions on 2D arrays. **[CR8, Skills 1A, 2A, 2B, 2C]**
Lab #**25**: *Chomp* (JM 9.5). **[CR8, Skills 1A, 2A, 2B, 2C]**

Module 5: Classes (5 weeks)

11. Details of defining classes and using objects (Weeks 19-20, *CED Topics 1.9-1.10, 1.12-1.14, Unit 3 all topics*)

Public and private instance variables and methods. Constructors and the `new` operator. References to objects. `this` keyword. Calling methods and accessing instance variables. Passing parameters to constructors and methods. `return` statement. Overloaded methods. Static variables and methods. [Skill 5.7]

Reading and exercises: JM Ch.10.
Case study: the `Fraction` class (JM 10.1 - 10.8). [CR5, CR4, Skills 3.A, 3.B, 3.C, 2.A, 2.B, 2.C]
Lab #**26**: *Time* (JM Ch. 10, Exercise 13). Design and implement the class `Time`. Include a constructor, `Time(int h, int m)`, a `private int` method `toMins` that returns the time in minutes since the beginning of the day for this `Time` object, as well as a `public boolean` method `lessThan(Time t)`. Add a `public int` method `elapsedTime(Time t)` that returns the number of minutes from `t` to this time. In this method, assume that `t` ≤ *this time* < `t + 24h`. Describe the purpose of each method and its return value in *javadoc* comments; include preconditions that describe the intitial conditions for each parameter to the constructor and each method, if any, in order for the method to work correctly. [CR8, CR6, Skills 4.A, 4.B]
Case study and Lab #**27**: *Snack Bar* (JM 10.9). [CR8, CR4, Skills 3.A, 3.B, 3.C, 2.A, 2.B, 2.C]
Lab #**28**: *Snack Bar Continued* (JM 10.12). [CR8, CR4, Skills 3.A, 3.B, 3.C, 2.A, 2.B, 2.C]

12. `ArrayList` (Weeks 21-22, *CED Topics 4.7-4.10*)

Wrapper classes. The `ArrayList` class. The `List` interface (optional). `ArrayList`'s constructors and methods. Pitfalls. `ArrayList` vs. built-in arrays.

Reading and exercises: JM Ch.11.
Be Prepared, JM 2.6.
Lab #**29**: *Shuffler* (JM 11.4). [CR8]
Lab #**30**: Creating an index for a document using `ArrayList` (JM 11.6). [CR6, CR8, Skills 4.A, 4.B]
Lab #**31**: Past AP free-response questions on `ArrayList`. [CR8, Skills 1A, 2A, 2B, 2C]

13. Reading text files (Week 23, *CED Topic 4.6*)

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

Reading and exercises: JM Ch.15.
Lab #**32**: *Choosing Words* (JM 15.5). [CR8, Skills 1A, 2A, 2B, 2C]
Lab #**33**: Exercises and projects from exercises and the Test Package for JM Ch.15.

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

14. Recursion (Week 24, *CED Topic 4.16*)

Recursive methods. Base case. Understanding and debugging recursive methods. When not to use recursion. [CR5, Skill 1.A]

Reading and exercises: JM Ch.13 and JM 19.3 - 19.4.
Lab #**34**: JM Ch.13 exercises.
Lab #**35**: The *Tower of Hanoi* (JM 19.5). [CR5, CR8, Skill 1.A]

15. Searching and sorting. Introduction to analysis of algorithms. (Weeks 25-27, *CED Topics 4.14, 4.15, 4.17*)

Comparing objects. The `equals` method. Sequential and Binary Search [CR5, Skills 3.A, 3.B, 3.C] The number of comparisons required in Sequential and Binary Search, Selection Sort, Insertion Sort, and Mergesort. Comparison of efficiency of "quadratic" sorting algorithms (Selection Sort and Insertion Sort) vs. Mergesort.

Reading and exercises: JM Ch.14.
Lab #**36**: JM Ch.14 exercises.
Lab #**37**: *Keeping Things in Order* (JM 14.4). [CR5, CR8, Skills 3.A, 3.B, 3.C]
Lab #**38**: *Benchmarks* (JM 14.9) — compares efficiency of several sorting algorithms. [CR5, Skills 1.A, 1.B, 3.A, 3.B, 3.C]

Module 7: Review (3 weeks) [CR2]

16. Review and practice for the AP exam (Weeks 28-30)

Java Quick Reference (library classes and methods tested on the AP exam). Past multiple-choice and free-response questions.

Reading: Be Prepared Chapters 1-4; *Be Prepared* Chapter 5 (past free-response questions and solutions), *Be Prepared* practice exams 1-4, *250 Multiple-Choice Computer Science Questions in Java.*
Practice taking the digital exam in Bluebook.

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

17. 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: JM Ch.16, 17, 18.
Lab #**39**: *Pieces of the Puzzle* (JM 16.7).
Lab #**40**: *Ramblecs* (JM 17.6).
Lab #**41**: *Slide Show* (JM 18.7).

18. Projects that demonstrate creative computer use.

Reading and exercises: Java Methods JM Ch.21, “Computing in Context: Creative, Responsible, and Ethical Computer Use”, JM 21.2.

Other suggested activities: a team project to implement a game (for example, the Game of SET, www.skylit.com/projects/); or a potentially useful project for the school or community.