Course: AP Computer Science A (Cortado) - New for Fall 2025! | Module: Using Objects and Methods

Lesson 1.1: Introduction to Algorithms, Programming, and Compilers

https://codehs.com/course/22971/lesson/1.1

Description	In this introductory Java lesson, students will explore the concept of algorithms, understand the structure of basic Java programs, and write their first lines of Java code using `System.out.println()`. They will also begin to recognize common syntax errors and learn the processes of compiling, executing, and debugging Java programs.
Objective	Students will be able to: • Define and construct step-by-step algorithms for real-world and computing tasks. • Identify the structure of a basic Java program and use `System.out.println()` to display output. • Describe and demonstrate the compilation and execution process of a Java program. • Identify and correct syntax errors through debugging exercises.
Activities	1.1.1 Video: Introduction to Algorithms 1.1.2 Free Response: Real World Algorithms 1.1.3 Video: Basic Java Program Structure 1.1.4 Video: Live Coding: The CodeHS IDE 1.1.5 Quiz: Basic Program Structure and Print Statements 1.1.6 Exercise: Exploration: Hello World 1.1.7 Exercise: Welcome Program 1.1.8 Exercise: ASCII Art 1.1.9 Video: Compilation, Execution, and Errors 1.1.10 Exercise: Debugging: Quotes 1.1.11 Example: Compiling and Running Java 1.1.12 Example: Explore the Exception Error 1.1.13 Free Response: Compilation and Execution: In Your Words 1.1.14 Quiz: Algorithms, Programming, and Compilers Quiz
Prior Knowledge	• Experience following step-by-step instructions (e.g., in a recipe or lab procedure). • Some basic comfort expressing a sequence of steps in writing.

Planning Notes

- Students may have varying ideas of what an algorithm is—prompt them to share real-life examples to make the concept relatable.
- Java syntax can feel overwhelming; reassure students that making mistakes is expected and part of learning.
- Emphasize that programming is about logical thinking, not memorization or typing speed.
- Use the compilation process to demystify how code becomes executable—encourage students to "think like the computer."
- Consider creating a classroom "bug wall" to celebrate common errors and their solutions as a shared learning experience.
- This is a longer lesson and will likely take more than one class period. Consider setting a goal for students to achieve for each class period.

Exploration Activities

- This lesson, and most lessons in this course, includes a new "Exploration" activity designed using the PRIMM model (Predict, Run, Investigate, Modify, Make). These activities help students develop their conceptual understanding of coding concepts before asking them to create programs on their own.
- With the surge of generative AI, the importance of being able to read code and modify it to your needs is more important than ever. These Exploration activities give students many opportunities to practice these vital skills throughout the course.
- We provide an **Investigate.txt** file in each Exploration activity where students can record their thoughts and observations. This file **will not be autograded**, but it is an important tool for developing a strong foundational knowledge of coding concepts and encouraging thoughtful experimentation.
- Consider modeling one cycle of PRIMM early on, especially the transition from prediction to investigation to modification, to help students understand the value of each phase.
- As students move through the course, you can decide whether or not students complete these activities on their own, in pairs, or as a class.

Standards Addressed

Teaching and Learning Strategies

Lesson Opener

- Have students brainstorm and write down answers to the discussion questions listed below. Students can work individually or in groups/pairs. Have them share their responses. [5 mins]

Activities

- Watch the instructional video *Introduction to Algorithms*. [5–7 mins]
  - Pause to define what an algorithm is and how detailed instructions improve clarity.

- Complete the *Real World Algorithms* free response activity. [10–12 mins]
  - Compare examples of well-written vs. poorly-written algorithms to highlight the importance of precision.
- Watch the video *Basic Java Program Structure*. [5 mins]
- Take the *Basic Program Structure and Print Statements* quiz. [5 mins]
- Complete the *Exploration: Hello World* activity. [10–12 mins]
  - Predict output, test variations, and document findings in the **Investigate.txt** file.
- Complete the *Welcome Program* exercise. [5–7 mins]
- Complete the *ASCII Art* exercise. [10–15 mins]
  - Optional: encourage students to create their own designs if time allows.
- Watch the video *Compilation, Execution, and Errors*. [5–7 mins]
- Complete the *Debugging: Quotes* exercise. [10–15 mins]
  - Students reflect on their debugging process in **Investigate.txt**.
- Analyze the *Compiling and Running Java* simulation activity. [10–12 mins]
- Analyze the *Explore the Exception Error* simulation activity. [10–12 mins]
- Complete the *Compilation and Execution: In Your Words* free response activity. [10 mins]
  - Students choose between writing an analogy or a compiler's point-of-view narrative.
- Take the *Algorithms, Programming, and Compilers* quiz. [5–7 mins]

Lesson Closer

- Have students brainstorm and write down answers to the discussion questions listed below. Students can work individually or in groups/pairs. Have them share their responses. [5 mins]

Discussion Questions

Beginning of Class

- What are some tasks you do every day that involve steps?
  - *Making lunch, brushing your teeth, tying your shoes…*
- What would happen if someone skipped a step or got them out of order?
  - *The task might not work—your sandwich could be missing something, or your shoes might come untied.*
- What do you think a computer needs in order to complete a task?
  - *It probably needs a list of exact steps to follow.*

End of Class

- What command is used to print in Java?
  - *System.out.println("text");*
- What happens if you forget the quotation marks or semicolon?

	◦ *A syntax error will occur because Java expects specific formatting.* • Why is it important to understand how a program is compiled and executed? ◦ *It helps you understand how your code is translated into actions and makes debugging easier.*
Resources/Handouts	

Vocabulary

Term	Definition
Debugging	Debugging is fixing a problem in your code.
Syntax error	An error in the sequence of words or rules in a program that prevents the program from running.
Algorithm	An algorithm is a set of steps or rules to follow to solve a particular problem.
System.out.println	Java method that lets us print out a line of output to the user
System.out.print	Java method that lets us print output to the user, without ending the line printed.
String	String is a Java type that represents a string of characters (text)
Syntax	The rules for writing code in a specific programming language

Modification: Advanced	Modification: Students Needing Additional Support	Modification: English Language Learners
• Challenge students to create two real-world algorithms—one with a flowchart. • Extend the ASCII Art task (1.1.7) by having students animate characters using multiple frames. • Have students use `System.out.print()` and `System.out.println()`	• Use printed pseudocode examples for algorithm writing. • Allow students to describe their algorithms orally or draw them instead of typing. • Work in pairs on debugging tasks and print statement exercises.	• Provide a vocabulary list with definitions: algorithm, print statement, output, compile, error. • Use translated captions on videos where available. • Provide sentence starters: ◦ *“An algorithm is…”* ◦ *“To print a message in Java, you use…”*

together to explore output formatting.		