Unit 01: Primitive Types

Content Area: **Computer Science**
Course(s):
Time Period: **Marking Period 1**
Length: **2-3 Weeks**
Status: **Published**

Summary

This unit introduces students to the Java programming language and the use of classes, providing students with a firm foundation of concepts that will be leveraged and built upon in all future units. Students will focus on writing the main method and will start to call preexisting methods to produce output. Students will start to learn about three built-in data types and learn how to create variables, store values, and interact with those variables using basic operations. The ability to write expressions is essential to representing the variability of the real world in a program and will be used in all future units.

Revison Date: July 2026

1	Make sense of problems and persevere in solving them
L.SS.11–12.1	Demonstrate command of the system and structure of the English language when writing or speaking.
2	Reason abstractly and quantitatively
8.1.12.CS.1	Describe ways in which integrated systems hide underlying implementation details to simplify user experiences.
8.1.12.CS.2	Model interactions between application software, system software, and hardware.
P.5	Utilize critical thinking to make sense of problems and persevere in solving them.
L.KL.11–12.2	Apply knowledge of language to understand how language functions in different contexts, to make effective choices for meaning or style, and to comprehend more fully when reading or listening.
4	Model with mathematics
8.1.12.CS.4	Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors.
8	Look for and express regularity in repeated reasoning
F.IF.A	Understand the concept of a function and use function notation
F.IF.A.1	Understand that a function from one set (called the domain) to another set (called the range) assigns to each element of the domain exactly one element of the range. If □ is a function and □ is an element of its domain, then □(□) denotes the output of □ corresponding to the input □. The graph of □ is the graph of the equation □ = □(□).
8.1.12.DA.3	Translate between decimal numbers and binary numbers.
8.1.12.DA.4	Explain the relationship between binary numbers and the storage and use of data in a computing device.
8.1.12.AP.1	Design algorithms to solve computational problems using a combination of original and existing algorithms.
8.1.12.AP.2	Create generalized computational solutions using collections instead of repeatedly using simple variables.
8.1.12.AP.5	Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.

Essential Questions & Enduring Understanding

- Can you convert between variable types?
- How are mathematical operations evaluated in Java?
- How do you recognize and correct runtime errors?
- How do you recognize and correct syntax errors?
- How is an Integrated Development Environment (IDE) used to create Java programs?
- What is Java?
- What types of file extensions are used for Java programs?
- What types of variables can be created in Java and what information can they contain?

Objectives

Students Will Know

- how to display information on the console.
- how to create and manipulate variables.
- how to preform basic mathematical operations using the Java language.

Students Will Be Skilled At

- rewriting mathematical formulas in Java.
- asking users for input and utilizing that input to solve a given problem.
- determining syntax error and runtime error causes and how to fix them.

Learning Plan

Lecture on primative types, pair programming exercise on evaluating Java expressions

Demonstration on IDE: creating new projects, new programs, running programs, debugging and reading error messages.

Pair programming exercises on error analysis.

Pair programming exercises on casting between primitive types.

Individual lab assignment demonstrating mastery of input and output, storage of variables & evaluating expressions using mathematical operations.

Assessment

Assessments

- Formative: Daily assessments using examples from class notes and CodeHS.com, AP Classroom/Albert Checks for Understanding
- Summative: Teacher-created assessments/projects and CodeHS Computer Science Projects, AP Classroom/Albert Unit Assessments
- Benchmark: Check for understanding benchmark assessments on CodeHS, AP Classroom/Albert/Khan Academy Diagnostics
- Alternative Assessments: Student-centered activities such as a doorbell coding project, game design projects, and other activities involving real world applications

complete performance tasks:

- Students will be able to design programs using appropriate code.
- Students will be able to write programs using appropriate code.

complete quizzes/tests:

- Design of a program
- Structure of a program
- Algorithms

complete sample AP multiple choice questions.

Materials

Core instructional materials: Computing Concepts with Java

Supplemental instructional materials:

Java Concepts for AP Computer Science Study Guide
CollegeBoard AP Classroom Website
CollegeBoard AP Computer Science A Website

Integrated Accomodations

Possible accommodations/modification for AP Computer Science A