

Programming I Java: Grades 9, 10, 11, 12

Adopted 2007

Introduction to Programming and Ethics in Programming

1.1 Define terminology

1. Prepare a list of terms with definitions [1.1.1](#)
-

1.2 Explain the difference between system and application software

1. Identify various software as either system or application [1.2.1](#)
-

1.3 Discuss terms related to hardware and software

1. Identify technology as either hardware or software [1.3.1](#)
-

1.4 Discuss various operating systems and their differences (i.e., Windows, Mac Linux)

1. Tell where each operating system is used most frequently [1.4.1](#)
-

1.5 Explain the difference between high-level and low-level languages

1. Classify commonly used programming languages as high-level or low-level [1.5.1](#)
-

1.6 Explain the difference between interpreters and compilers

1. Give an example of how a compiler functions vs. how an interpreter functions [1.6.1](#)
-

1.7 Explain the difference between executable code and bytecode

1. Draw a diagram of how a Java source code program is translated to bytecode and then to executable code [1.7.1](#)
-

Programming Techniques and Characteristics of Good Programs

2.1 Define terminology

1. Prepare a list of terms with definitions [2.1.1](#)
-

2.2 List the steps of the programming process

1. When given an example, be able to identify the correct step [2.2.1](#)
-

2.3 Identify the syntax of a simple program

1. Key, save, compile, and execute "Hello World" program [2.3.1](#)

2.4 Identify the syntax to output String literals to screen

1. Write programs that use System.out.println with String literals [2.4.1](#)
-

2.5 Identify syntax of comments

1. Use appropriate syntax to include comments in programs [2.5.1](#)
-

2.6 Explain the characteristics of user-friendly programs

1. Write programs that have clear instructions [2.6.1](#)
 2. Write programs whose output is easy to read and understand [2.6.2](#)
-

2.7 Explain the importance of program documentation and maintenance

1. Be able to use // and /* */ to write programs that are well-documented [2.7.1](#)
 2. Update an existing program [2.7.2](#)
-

2.8 Explain the importance of algorithm and/or pseudocode in program development

1. Write a pseudocode (algorithm) for a programming problem [2.8.1](#)
-

2.9 Identify different types of errors (syntax, semantic, run-time, compile time)

1. When given an example, identify the error type [2.9.1](#)
-

2.10 Explain the characteristics of readable programs

1. Explain the characteristics of readable programs [2.10.1](#)
 2. Use descriptive identifiers [2.10.2](#)
 3. Document difficult logic to make it easy to follow [2.10.3](#)
-

Data Types and Mathematical Operations

3.1 Define terminology

1. Prepare a list of terms with definitions [3.1.1](#)
-

3.2 List the following four primitive types: int, boolean, double, char.

1. Compare the four data types [3.2.1](#)
 2. Determine whether a particular "number" would be considered numeric [3.2.2](#)
 3. Determine whether a number should be treated as an integer or a floating point (i.e. single, double) [3.2.3](#)
 4. Designate data type using correct syntax [3.2.4](#)
-

3.3 Explain the use of a String object

1. Write programs that declare and utilize String objects [3.3.1](#)
2. Determine whether an identification number (such as Social Security Number) should be treated as a string or number [3.3.2](#)

3.4 Explain the purpose of concatenation

1. Write output lines using + as the concatenation operator [3.4.1](#)

3.5 Explain the purpose of escape sequences (\n, \t, \\)

1. Write programs that use escape sequences to print strings [3.5.1](#)

3.6 Explain the advantages of using integer variables whenever possible (faster computation, require less memory, obtain exact answers)

1. Use integer variables in programs where appropriate [3.6.1](#)

3.7 Explain the advantages and disadvantages of floating-point numbers (round-off errors, more memory, approximate answers, slower computation, size of numbers to be stored, etc.)

1. Use floating point variables in programs where appropriate [3.7.1](#)

3.8 List arithmetic operations and order of operations (*, /, %, +, -)

1. Write formulas using operators and order of operations [3.8.1](#)
2. Write programs that use mathematical operations correctly (integer arithmetic vs floating point arithmetic) [3.8.2](#)

3.9 Explain the difference in promotion and casting

1. Determine the correct answer to math expressions where casting and promotion are involved [3.9.1](#)
2. Write math expressions correctly that use promotion and casting [3.9.2](#)

3.10 Explain rules for choosing variable names

1. Write programs that use descriptive variable names [3.10.1](#)
2. Write programs that use descriptive variable names [3.10.2](#)

3.11 Explain the circumstances and give examples of appropriate occasions to use constants

1. Use constants when appropriate in programs [3.11.1](#)

Using Selected Standard Classes

4.1 Define terminology

1. Prepare a list of terms with definitions [4.1.1](#)

4.2 Explain the purpose of the compiler directive import

1. Write programs that import the selected classes [4.2.1](#)

4.3 Give examples of meaningful prompts

1. Write programs with meaningful prompts [4.3.1](#)

4.4 Describe how to instantiate a Scanner object. (java.util.Scanner)

1. Write a program that instantiates a Scanner object [4.4.1](#)
-

4.5 Describe how to read the following data types from the keyboard using the Scanner object: nextLine(), nextInt(), nextDouble() (java.util.Scanner)

1. Expand the program above so that it reads Strings, ints and doubles from the keyboard [4.5.1](#)
-

4.6 Describe how to close a Scanner object (java.util.Scanner)

1. Revise the program above so that it closes the scanner object [4.6.1](#)
-

4.7 Describe how to set up a variable to print currency or percents using NumberFormat and how to use the format method (java.text.NumberFormat)

1. Write a program that displays the output in currency and percent formats [4.7.1](#)
-

4.8 Describe how to instantiate a DecimalFormat object and how to use # and 0 to print the specified digits (java.text.DecimalFormat)

1. Write a program that instantiates a DecimalFormat object [4.8.1](#)
-

4.9 Describe how to print using a specific number of digits before and/or after the decimal place with the format method from DecimalFormat (java.text.DecimalFormat)

1. Write a program that displays the output with a specified number of decimal places [4.9.1](#)
-

4.10 Describe how to use the Math class to get a square root, power and/or absolute value (sqrt , pow , abs methods)

1. Write a program that uses sqrt, pow, and abs Math functions in calculations [4.10.1](#)
-

4.11 Describe how to obtain a random double and explain the range of possible values (Math class)

1. Write a program that obtains and uses random numbers using Math.Random() [4.11.1](#)
-

4.12 Describe how to obtain an integer random integer in the range of 0...N or 1...N inclusive (Math class)

1. Write a program that obtains and uses random integers in the ranges 1...N and 0...N using (int) ((high - low + 1) * Math.Random() + low) formula [4.12.1](#)
-

Decision Structure

5.1 Define terminology

1. Prepare a list of terms with definitions [5.1.1](#)

5.2 List relational operators

1. Write boolean expressions that use the appropriate relational operator [5.2.1](#)
-

5.3 Describe the process of comparing two strings

1. When given two strings, determine if they are equal, the first is smaller, or the first is larger [5.3.1](#)
-

5.4 Explain how to use the compareTo() and equals() methods from the String class

1. Write boolean expressions that compare strings for equality and order [5.4.1](#)
-

5.5 Explain the syntax and logic of if statements

1. Write programs that use if statements [5.5.1](#)
 2. Write programs that use braces correctly to form block if statements [5.5.2](#)
-

5.6 Explain the syntax and logic of if-else statements

1. Write statements that use if-else to make the correct decision based on the data; use braces where needed to block the statements [5.6.1](#)
-

5.7 Explain the syntax and logic of nested statements

1. Write programs using block if-else for 3 or more alternatives [5.7.1](#)
-

5.8 Explain the use of logical operators and , or , and not (&&, ||, !)

1. Write programs which require the use of &&, ||, and ! [5.8.1](#)
 2. Write a program that requires the use of short-circuit and (&&) and short-circuit or (||) [5.8.2](#)
-

Loops

6.1 Define terminology

1. Prepare a list of terms with definitions [6.1.1](#)
-

6.2 Describe the purpose and syntax of for loops

1. Write programs that use for loops [6.2.1](#)
-

6.3 Explain the procedure to use for loops to count in increments/decrements other than one

1. Write counting for loops with increments other than 1 [6.3.1](#)
-

6.4 Explain the syntax of nested loops

1. Determine the output of a nested loop [6.4.1](#)
 2. Write programs that use nested loops for [6.4.2](#)
-

6.5 Explain the logic of while loops

1. Write programs that use loops while [6.5.1](#)

6.6 Explain the process of using counters with loops

1. Write programs that use counters with loops [6.6.1](#)

6.7 Explain the logic of using accumulators with loops

1. Write programs that use accumulators with loops [6.7.1](#)

6.8 Explain the difference in the effect of a while loop (entrance condition loop) and a do while (exit condition loop) loop

1. Write programs that use do while loops [6.8.1](#)