**Self-Review Exercises**

**2.1 Fill in the blanks**

a) A(n) **left brace {** begins the body of every method, and a(n) **right brace }** ends the body of every method.  
b) You can use the **if** statement to make decisions.  
c) **//** begins an end-of-line comment.  
d) **Space, tab, newline** are called white space.  
e) **Keywords** are reserved for use by Java.  
f) Java applications begin execution at method **main**.  
g) Methods **System.out.print**, **System.out.println**, and **System.out.printf** display information in a command window.

**2.2 True or False**

a) **False** – Comments are ignored by the compiler and do not affect program execution.  
b) **True** – All variables must have a type when declared.  
c) **False** – Java is case-sensitive, so number and NuMbEr are different variables.  
d) **False** – The remainder operator % can be used with both integer and floating-point operands.  
e) **False** – \*, /, and % have higher precedence than + and -.

**2.3 Write Statements**

a) int c, thisIsAVariable, q76354, number;  
b) System.out.print("Enter an integer: ");  
c) int value = input.nextInt();  
d) System.out.println("This is a Java program");  
e) System.out.printf("%s%n%s%n", "This is a Java", "program");  
f) if (number != 7) System.out.println("The variable number is not equal to 7");

**2.4 Identify and Correct Errors**

a) **Incorrect:** if (c < 7); System.out.println("c is less than 7");  
**Corrected:** if (c < 7) System.out.println("c is less than 7"); (Remove the semicolon after if)

b) **Incorrect:** if (c => 7) System.out.println("c is equal to or greater than 7");  
**Corrected:** if (c >= 7) System.out.println("c is equal to or greater than 7"); (=> should be >=)

**2.5 Write Declarations & Statements**

a) // This program calculates the product of three integers  
b) Scanner input = new Scanner(System.in);  
c) int x, y, z, result;  
d) System.out.print("Enter the first integer: ");  
e) x = input.nextInt();  
f) System.out.print("Enter the second integer: ");  
g) y = input.nextInt();  
h) System.out.print("Enter the third integer: ");  
i) z = input.nextInt();  
j) result = x \* y \* z;  
k) System.out.printf("Product is %d%n", result);

**2.7 Fill in the blanks**

a) **Comments** are used to document a program and improve its readability.  
b) A decision can be made in a Java program with a(n) **if statement**.  
c) Calculations are normally performed by **assignment** statements.  
d) The arithmetic operators with the same precedence as multiplication are **division /** and **modulus %**.  
e) When parentheses in an arithmetic expression are nested, the **innermost** set of parentheses is evaluated first.  
f) A location in the computer’s memory that may contain different values at various times throughout the execution of a program is called a(n) **variable**.

**2.8 Write Statements**

a) System.out.print("Enter an integer: ");  
b) a = b \* c;  
c) // This program performs a sample payroll calculation

**2.9 True or False**

a) **False** – Operators follow a precedence hierarchy, not just left-to-right evaluation.  
b) **True** – All variable names listed are valid.  
c) **False** – Operator precedence affects evaluation order.  
d) **True** – Variable names cannot start with a digit.

**2.10 Output Prediction**

a) x = 2  
b) Value of 2 + 2 is 4  
c) x =  
d) 5 = 5

**2.11 Modifying Statements**

a) **Modifies p**  
b) **Does not modify variables**  
c) **Does not modify variables**  
d) **Modifies value**

**2.12 Correct Java Statements for Equation**

a) ✅ Correct  
b) ❌ Incorrect  
c) ❌ Incorrect  
d) ✅ Correct  
e) ✅ Correct  
f) ❌ Incorrect

**2.13 Order of Evaluation**

a) x = 7 + (3 \* 6 / 2) - 1 = 7 + 9 - 1 = 15  
b) x = (2 % 2) + (2 \* 2) - (2 / 2) = 0 + 4 - 1 = 3  
c) x = 3 \* 9 \* (3 + (9 \* 3 / 3)) = 3 \* 9 \* (3 + 9) = 3 \* 9 \* 12 = 324

**2.19 Output**

System.out.printf("\*%n\*\*%n\*\*\*%n\*\*\*\*%n\*\*\*\*\*%n");

**Output:**

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

**Explanation:**

* The %n is a newline character, so each \* pattern prints on a new line.

**2.20 Output**

System.out.println("\*");

System.out.println("\*\*\*");

System.out.println("\*\*\*\*\*");

System.out.println("\*\*\*\*");

System.out.println("\*\*");

**Output:**

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*

\*\*

**Explanation:**

* Each System.out.println() prints the string and moves to the next line.

**2.21 Output**

System.out.print("\*");

System.out.print("\*\*\*");

System.out.print("\*\*\*\*\*");

System.out.print("\*\*\*\*");

System.out.println("\*\*");

**Output:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Explanation:**

* System.out.print() prints everything on the same line.
* System.out.println("\*\*"); ends the line after printing "\*\*".

**2.22 Output**

java

Copy code

System.out.print("\*");

System.out.println("\*\*\*");

System.out.println("\*\*\*\*\*");

System.out.print("\*\*\*\*");

System.out.println("\*\*");

**Output:**

\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*

**Explanation:**

* System.out.print("\*"); prints \* without a newline.
* System.out.println("\*\*\*"); prints \*\*\* after \* on the same line.
* System.out.println("\*\*\*\*\*"); moves to the next line.
* System.out.print("\*\*\*\*"); prints \*\*\*\* on the same line.
* System.out.println("\*\*"); appends \*\* to \*\*\*\*, then moves to the next line.

**2.23 Output**

System.out.printf("%s%n%s%n%s%n", "\*", "\*\*\*", "\*\*\*\*\*");

**Output:**

\*

\*\*\*

\*\*\*\*\*

**Explanation:**

* %s is replaced by the corresponding string ("\*", "\*\*\*", "\*\*\*\*\*").
* %n adds a newline after each string.