**Chapter 2**

**Self-Review Exercises**

**2.1 Fill in the blanks in each of the following statements:**

a) A(n) opening brace begins the body of every method, and a(n) closing 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) Spaces, tabs, and newlines 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 State whether each of the following is true or false. If false, explain why.**

a) Comments cause the computer to print the text after the // on the screen when the program executes.

Answer : False. Comments do not cause the computer to print the text after the `//` on the screen when the program executes. Comments are ignored by the compiler and are meant for human readers to provide explanations or notes in the code.

b) All variables must be given a type when they’re declared.

Answer : True. All variables must be given a type when they’re declared in Java. Java is a statically typed language, meaning that the type of a variable must be specified at compile time.

c) Java considers the variables number and NuMbEr to be identical.

Answer : False. Java considers the variables `number` and `NuMbEr` to be different. Java is case-sensitive, so variable names that differ only in case are treated as distinct identifiers.

d) The remainder operator (%) can be used only with integer operands.

Answer : False. The remainder operator `%` can be used with both integer and floating-point operands in Java. For example, `5.5 % 2.0` is valid and will yield a floating-point result.

e) The arithmetic operators \*, /, %, + and - all have the same level of precedence.

Answer : False. The arithmetic operators `\*`, `/`, `%`, `+`, and `-` do not all have the same level of precedence. In Java, multiplication (`\*`), division (`/`), and remainder (`%`) have higher precedence than addition (`+`) and subtraction (`-`). This means that multiplication, division, and remainder operations are performed before addition and subtraction in expressions unless parentheses are used to change the order of operations.

**2.4 Identify and correct the errors in each of the following statements:**

a) if (c < 7); System.out.println("c is less than 7");

Error: The semicolon (;) after the if condition terminates the if statement, making the subsequent System.out.println statement execute unconditionally.

Ans : if (c < 7) System.out.println("c is less than 7");

b) if (c => 7) System.out.println("c is equal to or greater than 7");

Error: The operator => is incorrect. The correct operator for "greater than or equal to" in Java is >=

Ans :if (c >= 7) System.out.println("c is equal to or greater than 7");

**Exercises**

**2.7 Fill in the blanks in each of the following statements:**

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.

**2.9 State whether each of the following is true or false. If false, explain why.**

a) Java operators are evaluated from left to right.

False.

Explanation: Java operators are evaluated based on operator precedence and associativity. While many operators are left-associative, not all expressions are simply evaluated left to right. For example, multiplication and division have higher precedence than addition or subtraction.

b) The following are all valid variable names: under\_bar, m928134, t5, j7, her\_sales$, his\_$account\_total, a, b$, c, z and z2.

True.

Explanation: All these names follow Java’s variable naming rules:

Can contain letters, digits, underscores \_, and dollar signs $

Cannot begin with a digit

Cannot be a reserved keyword

c) A valid Java arithmetic expression with no parentheses is evaluated from left to right.

False.

Explanation: Operator precedence still applies. For example, in the expression 2 + 3 \* 4, multiplication is done before addition, giving 2 + 12 = 14, not (2 + 3) \* 4 = 20.

d) The following are all invalid variable names: 3g, 87, 67h2, h22 and 2h.

➤ False.

Explanation: Only 3g, 87, 67h2, and 2h are invalid because variable names cannot start with a digit.

However, h22 is valid, because it starts with a letter and follows naming rules.

**2.10 Assuming that x=2 and y=3, what does each of the following statements display?**

1. System.out.printf("x = %d%n", x);

1. System.out.printf("Value of %d + %d is %d%n", x, x, (x + x));

1. System.out.printf("x =");

1. System.out.printf("%d = %d%n", (x + y), (y + x));

Output : 5 = 5

**2.11 Which of the following Java statements contain variables whose values are modified?**

a) p = i + j + k + 7;

Yes – p is assigned a new value based on the sum of i, j, k, and 7.

b) System.out.println("variables whose values are modified");

No – Just prints a message. No variable is modified.

c) System.out.println("a = 5");

No – Just prints a string. The value of a is not being changed.

d) value = input.nextInt();

Yes – The variable value is assigned a new value from user input.

**2.12 Given that y = ax3 + 7, which of the following are correct Java statements for this equation?**

a) y = a \* x \* x \* x + 7;

Correct – This matches the original equation exactly.

b) y = a \* x \* x \* (x + 7);

Incorrect – This changes the expression to ax²(x + 7), not ax³ + 7.

c) y = (a \* x) \* x \* (x + 7);

Incorrect – This evaluates to ax²(x + 7), also incorrect.

d) y = (a \* x) \* x \* x + 7;

Correct – This simplifies to a \* x \* x \* x + 7, which matches.

e) y = a \* (x \* x \* x) + 7;

Correct – Same as a \* x³ + 7, just grouped differently.

f) y = a \* x \* (x \* x + 7);

Incorrect – This expands to a \* x \* (x² + 7), not the same expression.

**2.13 State the order of evaluation of the operators in each of the following Java statements, and show the value of x after each statement is performed:**

 a) x = 7 + 3 \* 6 / 2 - 1;

b) x = 2 % 2 + 2 \* 2 - 2 / 2;

c) x=(3 \* 9 \* (3 + (9 \* 3 / (3))));

Answer :

public class Evaluation {

    public static void main(String[] args) {

        int x;

        // a) x = 7 + 3 \* 6 / 2 - 1

        x = 7 + 3 \* 6 / 2 - 1;

        System.out.println("a) x = " + x); // Expected: 15

        // b) x = 2 % 2 + 2 \* 2 - 2 / 2

        x = 2 % 2 + 2 \* 2 - 2 / 2;

        System.out.println("b) x = " + x); // Expected: 3

        // c) x = (3 \* 9 \* (3 + (9 \* 3 / (3))))

        x = (3 \* 9 \* (3 + (9 \* 3 / (3))));

        System.out.println("c) x = " + x); // Expected: 324

    }

}

2.22 What does the following code print?

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

 System.out.println("\*\*\*");

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

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

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

**Answer :**

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

prints \* (no newline)

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

prints \*\*\* immediately after \*, making the line \*\*\*\*, and then moves to the next line

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

 prints \*\*\*\*\* on a new line

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

prints \*\*\*\* (no newline)

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

prints \*\* right after \*\*\*\*, making \*\*\*\*\*\*, and moves to a new line