Java Developer Internship Task

Detailed Answers to Interview Questions

Task 1: Java Console Calculator

What is method overloading?

- Method overloading in Java allows a class to have more than one method with the same name, but different parameters. This means you can create multiple methods with the same name but with different parameter types, order, or number of parameters. This helps increase the readability and reusability of the code. The method to be executed is determined at compile time based on the method signature.

- For example:
- int add(int a, int b)
- double add(double a, double b)

- Both methods perform addition, but one works for integers and the other for doubles. This is a type of compile-time polymorphism in Java.

How do you handle divide-by-zero?

- Divide-by-zero is a common arithmetic error in Java that can cause the program to crash. To handle this, we can use conditional checks before performing the division operation or use a try-catch block to catch the ArithmeticException.

- For instance:
- if (denominator != 0) {
-     result = numerator / denominator;
- } else {
-     System.out.println("Cannot divide by zero");
- }

- This ensures the program continues running smoothly without runtime errors and provides feedback to the user.

Difference between == and .equals()?

- In Java, `==` and `.equals()` are used for comparison, but they serve different purposes.

- `==` compares the reference or memory address of objects. So, if two string objects contain the same characters but are stored in different memory locations, `==` will return false.

- `.equals()` is a method used to compare the contents (values) of two objects. For strings and many other classes, `.equals()` is overridden to check the logical value.

- Example:
- String a = new String("hello");
- String b = new String("hello");
- a == b → false (different objects)
- a.equals(b) → true (same content).

What are the basic data types in Java?

- Java provides 8 primitive data types, each designed to store specific types of values:

- 1. byte - 8-bit integer (-128 to 127)
- 2. short - 16-bit integer
- 3. int - 32-bit integer (commonly used)
- 4. long - 64-bit integer
- 5. float - 32-bit decimal (single precision)
- 6. double - 64-bit decimal (double precision)
- 7. char - 16-bit Unicode character
- 8. boolean - stores true or false

- These types are not objects and are efficient in terms of performance and memory usage.

How is Scanner used for input?

- The Scanner class in Java (from java.util package) is used to read user input from the console. It simplifies input handling by offering different methods to read different types like int, double, string, etc.

- Steps to use Scanner:
- 1. Import it: import java.util.Scanner;
- 2. Create an object: Scanner sc = new Scanner(System.in);
- 3. Use methods like nextInt(), nextLine(), nextDouble(), etc.

- Example:
- Scanner sc = new Scanner(System.in);
- System.out.print("Enter a number: ");
- int num = sc.nextInt();

Explain the role of a loop.

- Loops in Java are used to execute a block of code repeatedly as long as a certain condition is met. They reduce code repetition and automate repetitive tasks.

- Java supports three main types of loops:
- 1. for loop – best when the number of iterations is known.
- 2. while loop – best when the condition is evaluated first.
- 3. do-while loop – executes the block at least once, then checks the condition.

- Loops improve code structure and efficiency, especially when processing collections or performing repeated calculations.

Difference between while and for loop?

- Both while and for loops are used to execute repetitive code, but they have differences:

- for loop:
- - Used when the number of iterations is known.
- - Initialization, condition, and increment/decrement are written in one line.
- Example:
- for (int i = 0; i < 5; i++) { ... }

- while loop:
- - Used when the number of iterations is not known in advance.
- - Only the condition is in the loop declaration.
- Example:
- int i = 0;
- while (i < 5) {
-     i++;
- }

- Use `for` for fixed loops, `while` for conditional or indefinite loops.

What is the JVM?

- JVM (Java Virtual Machine) is the part of Java Runtime Environment (JRE) responsible for running Java bytecode. It provides a platform-independent execution environment by converting bytecode into machine-specific instructions.

- Main roles of JVM:
- - Executes bytecode
- - Manages memory (heap, stack)
- - Handles garbage collection
- - Ensures security and performance

- JVM is what makes Java platform-independent. Each OS has its own version of JVM that executes the same Java bytecode.

How is Java platform-independent?

- Java is platform-independent due to its unique compilation process. When you write Java code, it is compiled into bytecode using the Java compiler. This bytecode is not specific to any operating system or hardware.

- Instead, it runs on the Java Virtual Machine (JVM), which is available for various platforms. This means you can write your code once and run it anywhere that has a JVM installed.

- This concept is known as WORA — Write Once, Run Anywhere — and is a core advantage of Java.

How do you debug a Java program?

- Debugging is the process of identifying and fixing bugs in your code. Java provides multiple tools and practices for debugging:

- 1. Using System.out.println() to trace variable values.
- 2. Using breakpoints in IDEs like IntelliJ or Eclipse to pause and examine program flow.
- 3. Using a debugger to step through code line by line.
- 4. Reading exception stack traces to locate the issue.

- Good debugging helps find logical errors, understand unexpected behavior, and ensure the program works correctly under all conditions.