JAVA DEVELOPER INTERNSHIP

TASK 2: Variables, Data Types & Console Input Application

Objective

The objective of this task is to gain a strong foundation in **Java variables, data types, console input handling, and basic program logic**. This task focuses on understanding memory usage, type casting, variable scope, and writing clean, well-structured Java code using best practices.

Tools Used

- **Primary Tools:** Eclipse
- **Alternative Tools:** Text Editor

Task Activities Performed

1. Primitive Data Types Declaration

- Declared variables of all Java primitive data types:
    - byte, short, int, long
    - float, double
    - char, boolean
- Explained their **memory usage** using inline comments in the source code.

2. Console Input Using Scanner

- Used the Scanner class to accept multiple types of user input such as:
    - Integers
    - Floating-point numbers
    - Characters
- Ensured proper input handling to avoid runtime exceptions.

3. Arithmetic Operations

- Implemented basic arithmetic operations:
    - Addition
    - Subtraction
    - Multiplication
    - Division
- Operations were performed using **user-provided input**.

4. Type Casting Demonstration

- Demonstrated:
    - **Implicit (Widening) Casting**

    - **Explicit (Narrowing) Casting**
- Explained the difference between compatible and incompatible data type conversions.

5. Invalid Input Handling

- Used conditional checks to validate user input.
- Prevented invalid operations such as division by zero.

6. Formatted Output

- Displayed results using:

System.out.printf()

- Ensured clean and readable console output.

7. Variable Scope Demonstration

- Demonstrated the difference between:
    - **Local Variables**
    - **Instance Variables**
    - **Static Variables**
- Explained variable scope and lifetime through comments and examples.

8. Code Documentation

- Added meaningful inline comments explaining:
    - Why specific data types were chosen
    - Logic behind calculations
    - Purpose of variables and methods

📘 Interview Questions & Answers

🔹 Difference between Primitive and Non-Primitive Data Types

- **Primitive:** Stores simple values directly (int, double, char, etc.).
- **Non-Primitive:** Stores references to objects (String, Arrays, Classes).

🔹 What is Type Casting?

Type casting is the process of converting one data type into another.

🔹 Why is Scanner Preferred Over BufferedReader?

Scanner provides built-in methods for reading different data types easily, whereas BufferedReader requires manual parsing.

🔹 What is the Default Value of Variables?

- Instance variables have default values.
- Local variables do **not** have default values and must be initialized.

🔹 **Explain Scope of Variables**

Scope defines where a variable is accessible:

- Local → inside a method
- Instance → per object
- Static → shared across all objects

📁 **Deliverables**

- ✓ Interactive console-based calculator
- ✓ Clean, well-commented Java source code
- ✓ README.md documentation

- ✅ **Final Outcome**
- By completing this task, I developed a clear understanding of Java fundamentals including data types, input handling, type casting, variable scope, and formatted output. This task strengthened my ability to write clean, readable, and error-resistant Java programs.

Outcomes

```java
import java.util.Scanner;

/*
 * TASK 2: Variables, Data Types & Console Input Application
 * This program demonstrates:
 * - Primitive data types
 * - Scanner input
 * - Arithmetic operations
 * - Type casting
 * - Variable scope
 * - Formatted output
 */

public class JavaBasicsDemo {

    // Instance variable (belongs to object)
    int instanceVar = 100;

    // Static variable (shared across all objects)
    static int staticVar = 200;

    public static void main(String[] args) {

        // Local variables (exist only inside this method)
        byte b = 10;          // 1 byte
        short s = 100;        // 2 bytes
        int i = 1000;         // 4 bytes
        long l = 100000L;     // 8 bytes

        float f = 10.5f;      // 4 bytes
        double d = 99.99;     // 8 bytes
```

```java
        char c = 'A';         // 2 bytes
        boolean flag = true;  // 1 bit (logical)

        // Display primitive values
        System.out.println("---- Primitive Data Types ----");
        System.out.println("byte: " + b);
        System.out.println("short: " + s);
        System.out.println("int: " + i);
        System.out.println("long: " + l);
        System.out.println("float: " + f);
        System.out.println("double: " + d);
        System.out.println("char: " + c);
        System.out.println("boolean: " + flag);

        Scanner sc = new Scanner(System.in);

        // Console input
        System.out.println("\n---- Console Input ----");
        System.out.print("Enter first number: ");
        int num1 = sc.nextInt();

        System.out.print("Enter second number: ");
        int num2 = sc.nextInt();

        // Arithmetic operations
        int sum = num1 + num2;
        int diff = num1 - num2;
        int product = num1 * num2;

        System.out.println("\n---- Arithmetic Operations ----");
        System.out.printf("Addition: %d + %d = %d%n", num1, num2, sum);
```

```java
        System.out.printf("Subtraction: %d - %d = %d%n", num1, num2, diff);
        System.out.printf("Multiplication: %d * %d = %d%n", num1, num2, product);

        // Division with validation
        if (num2 != 0) {
            double division = (double) num1 / num2; // explicit casting
            System.out.printf("Division: %.2f%n", division);
        } else {
            System.out.println("Division not possible (division by zero).");
        }

        // Type Casting
        System.out.println("\n---- Type Casting ----");

        // Implicit (Widening) Casting
        int x = 50;
        double y = x; // int to double
        System.out.println("Implicit Casting (int to double): " + y);

        // Explicit (Narrowing) Casting
        double p = 75.45;
        int q = (int) p; // double to int
        System.out.println("Explicit Casting (double to int): " + q);

        // Variable scope demonstration
        JavaBasicsDemo obj = new JavaBasicsDemo();
        System.out.println("\n---- Variable Scope ----");
        System.out.println("Local variable num1: " + num1);
        System.out.println("Instance variable: " + obj.instanceVar);
        System.out.println("Static variable: " + staticVar);
```