Assignment-7

Title: Program to Demonstrate Exception Handling for Division by Zero and Invalid Input in Java

Objective: To implement a Java program that demonstrates exception handling for division by zero and invalid input using try, catch, and finally blocks.

Theory: Exception handling in Java allows programmers to handle runtime errors gracefully without terminating the program abruptly. Java provides five key keywords for exception handling:

- try: Defines a block of code to test for exceptions.
- catch: Catches and handles exceptions if they occur.
- finally: A block that executes after the try and catch blocks, regardless of an exception.
- throw: Used to explicitly throw an exception.
- throws: Declares exceptions that a method might throw.

In this lab, we will demonstrate exception handling for two scenarios:

1. Division by zero
2. Invalid input (non-integer values)

Algorithm:

1. Start
2. Prompt the user to enter two integers: numerator and denominator.
3. Use a try block to:
    - Read user input.
    - Perform division.
4. Use catch blocks to handle exceptions:
    - ArithmeticException for division by zero.
    - InputMismatchException for invalid input.
5. Display appropriate error messages.
6. Use the finally block to display a completion message.
7. End

Program Code:

```
import java.util.InputMismatchException;
import java.util.Scanner;

public class ExceptionHandlingDemo {
   public static void main(String[] args) {
```

```
        Scanner scanner = new Scanner(System.in);

        try {
            System.out.print("Enter numerator: ");
            int numerator = scanner.nextInt();

            System.out.print("Enter denominator: ");
            int denominator = scanner.nextInt();

            int result = numerator / denominator;
            System.out.println("Result: " + result);
        } catch (ArithmeticException e) {
            System.out.println("Error: Division by zero is not allowed.");
        } catch (InputMismatchException e) {
            System.out.println("Error: Invalid input. Please enter integers only.");
        } finally {
            System.out.println("Execution completed.");
            scanner.close();
        }
    }
}
```

Expected Output (Example 1 - Valid Input):

```
Enter numerator: 10
Enter denominator: 2
Result: 5
Execution completed.
```

Expected Output (Example 2 - Division by Zero):

```
Enter numerator: 10
Enter denominator: 0
Error: Division by zero is not allowed.
Execution completed.
```

Expected Output (Example 3 - Invalid Input):

```
Enter numerator: ten
Error: Invalid input. Please enter integers only.
Execution completed.
```

Explanation:

1. The program uses a try block to read input and perform division.
2. If the denominator is zero, an ArithmeticException is caught and handled.
3. If the user enters non-integer values, an InputMismatchException is caught and handled.
4. The finally block ensures the program concludes gracefully, closing the scanner.

Conclusion: This lab demonstrated how to handle exceptions in Java effectively. By using try, catch, and finally blocks, we ensured the program could handle errors like division by zero and invalid input without crashing.