Java Calculator with Exception Handling

### **Project Overview**

This Java-based calculator is a **console application** that performs basic arithmetic operations—**Addition, Subtraction, Multiplication, Division, and Modulus**—while implementing **robust exception handling** to manage invalid user input and runtime issues.

### **Objective**

* Perform arithmetic operations based on user input.
* Handle all types of runtime exceptions gracefully.
* Demonstrate best practices in user input validation and exception management.

### **Concepts Covered**

* Java I/O (Scanner)
* Switch Expressions
* Exception Hierarchy:  
  + InputMismatchException
  + ArithmeticException
  + IllegalArgumentException
  + General Exception
* Use of finally block for resource management

### **Technology Used**

* Language: Java 17+
* IDE: IntelliJ IDEA / Eclipse / VS Code
* Execution: Console-based application

## **Code Explanation**

### **📁 Package Declaration**

package calculator;

* Declares that this class belongs to the calculator package (logical grouping of related classes).

### **📦 Import Statements**

import java.util.InputMismatchException;

import java.util.Scanner;

* Scanner: Used to take user input from the console.
* InputMismatchException: Specific exception for mismatched input types (e.g., user types text instead of a number).

### **🧠 Main Class**

public class Calculator {

* Defines the Calculator class, which contains the main() method. Execution starts here.

### **🚪 Entry Point (main method)**

public static void main(String[] args) {

* The starting point of the program.
* String[] args: Used to take command-line arguments (not used here).

### **🔧 Scanner Initialization**

Scanner input = new Scanner(System.in);

* Creates a Scanner object named input to read input from the console.

### **🧪 Try Block – User Input & Operation**

try {

* This block attempts to run code that **may throw exceptions**.

### **🔢 Accept First Number**

System.out.print("Enter 1st number : ");

float n1 = input.nextFloat();

* Prompts user to input the first number.
* Reads a float number using nextFloat().

### **🔢 Accept Second Number**

System.out.print("Enter 2nd number: ");

float n2 = input.nextFloat();

* Prompts user to input the second number.

### **➕ Accept Operator**

System.out.print("Enter operator (+, -, \*, /, %): ");

char operator = input.next().charAt(0);

* Prompts user to enter an arithmetic operator.
* next() reads a string token; charAt(0) extracts the first character.

### **🔄 Switch Statement – Perform Operation**

switch (operator) {

* Matches the operator character and executes the respective arithmetic block.

The **arrow (->) syntax** in the switch statement is part of the **enhanced switch expression**, which was introduced in **Java 14**.

#### **✅ Case '+'**

case '+' -> System.out.println("Result: " + (n1 + n2));

* Adds two numbers and displays the result.

#### **➖ Case '-'**

case '-' -> System.out.println("Result: " + (n1 - n2));

* Subtracts the second number from the first.

#### **✖️ Case '\*'**

case '\*' -> System.out.println("Result: " + (n1 \* n2));

* Multiplies the two numbers.

#### **➗ Case '/'**

case '/' -> {

if (n2 == 0) {

throw new ArithmeticException("Cannot divide by zero.");

}

System.out.println("Result: " + (n1 / n2));

}

* Checks for division by zero.
* If n2 is not zero, performs division.

#### **➗ Case '%'**

case '%' -> {

if (n2 == 0) {

throw new ArithmeticException("Cannot perform modulus by zero.");

}

System.out.println("Result: " + (n1 % n2));

}

* Handles modulus and also checks for division by zero.

#### **❌ Default Case – Invalid Operator**

default -> throw new IllegalArgumentException("Invalid operator. Use +, -, \*, /, or %.");

* If operator is not one of the five expected, throws a custom error.

### **🚫 Exception Handlers**

#### **1. InputMismatchException**

} catch (InputMismatchException ime) {

System.out.println("Error: Please enter valid numeric values.");

}

* Catches wrong data type input (e.g., string instead of number).

#### **2. ArithmeticException**

catch (ArithmeticException ae) {

System.out.println("Error: " + ae.getMessage());

}

* Catches math errors like division/modulus by zero.

#### **3. IllegalArgumentException**

catch (IllegalArgumentException iae) {

System.out.println("Error: " + iae.getMessage());

}

* Catches explicitly thrown exception for wrong operators.

#### **4. General Exception**

catch (Exception e) {

System.out.println("Unexpected error occurred: " + e);

}

* Catch-all for any other unexpected exceptions.

### **✅ Finally Block – Resource Cleanup**

finally {

input.close();

System.out.println("Calculator operation completed.");

}

* Always executes (even if exception occurs).
* Closes the Scanner object to free resources.

### **🔎 Test Scenarios and Expected Outputs**

| **Scenario** | **Input** | **Output** |
| --- | --- | --- |
| Valid Addition | 10, 5, + | Result: 15.0 |
| Division by zero | 10, 0, / | Error: Cannot divide by zero. |
| Invalid operator | 10, 2, x | Error: Invalid operator. Use +, -, \*, /, or %. |
| Invalid input (text instead of no) | ten, 2, + | Error: Please enter valid numeric values. |
| Modulus by zero | 10, 0, % | Error: Cannot perform modulus by zero. |

**Conclusion**

This PoC demonstrates a simple yet robust Java calculator that ensures:

* **Safe user interactions**
* **Meaningful error messages**
* **Smooth control flow** It is ideal for learning basic **Java exception handling**, **user input management**, and **clean console output formatting**.