**PROJECT TITLE:**

**CA LCULATOR APP**

**PRESENTED BY:**

**BHAVANI.A**

**INTERN ID:**

**VN-JD-4W213**

****

**INTRODUCTION**

The **Java Arithmetic Calculator Application** is a user-friendly tool designed to perform basic arithmetic operations such as addition, subtraction, multiplication, and division. Developed using the Java programming language, this project provides a simple and efficient graphical user interface (GUI) built with the Swing framework.

The primary objective of this application is to demonstrate core Java programming concepts, including object-oriented design, event handling, and exception management, while providing an interactive experience for users. The calculator accepts two numeric inputs, processes the chosen operation, and displays the result instantly.

Key features of the project include robust input validation to handle erroneous or invalid user inputs and dynamic feedback for division by zero and other exceptions. The use of modular programming ensures that the application logic is well-structured and maintainable.

This project serves as an excellent foundation for understanding the integration of Java's core and GUI libraries, making it suitable for beginners and intermediate developers looking to explore Java application development.

**ABSTRACT:**

The **Java Arithmetic Calculator Application** is a lightweight and efficient desktop-based utility designed to perform fundamental arithmetic operations, including addition, subtraction, multiplication, and division. Built using the Java programming language and the Swing framework, the application offers an intuitive graphical user interface (GUI) for easy interaction.

The primary goal of this project is to provide a practical implementation of Java concepts, such as modular programming, event-driven design, and exception handling. The application processes user inputs in real-time, validates the data to prevent errors, and delivers accurate results. Robust error handling mechanisms are integrated to manage common issues such as invalid input formats and division by zero.

Designed with simplicity and usability in mind, the project highlights the versatility of Java in creating interactive desktop applications. The modular structure ensures that the codebase is maintainable and extensible, allowing for future enhancements like advanced mathematical operations or data persistence. This project is a practical demonstration of Java's capabilities and serves as a foundational step for aspiring developers in application development.

**SYSTEM ARCHITECTURE**

The **System Architecture** of the Java Arithmetic Calculator Application is designed with a modular and layered approach to ensure efficiency, maintainability, and user-friendliness. At the core is the **User Interface Layer**, developed using the Swing framework, which provides an interactive graphical interface for users. This layer facilitates input collection through text fields, allows users to select operations via buttons, and displays results or error messages in a designated output area. Beneath the user interface lies the **Logic Layer**, which encapsulates the core functionality of the calculator, including methods for addition, subtraction, multiplication, and division. This layer processes the inputs received from the user and returns the computed results. Robust error handling is embedded within the logic to validate inputs and manage exceptions, such as division by zero or invalid data. The architecture ensures clear separation between the user interface and the business logic, enhancing the application's maintainability and scalability, while providing a seamless user experience.

**IMPLEMENTATION:**

The implementation of the **Java Arithmetic Calculator Application** involves developing a modular and interactive system using Java and the Swing framework. The project is structured into the following components:

1. **Graphical User Interface (GUI)**:  
   The user interface is created using Java's Swing framework, which provides a simple and interactive platform for user input and output. The GUI includes:
   * **Text fields** for users to enter two numerical values.
   * **Buttons** for each arithmetic operation: Add, Subtract, Multiply, and Divide.
   * **Labels** to display the result and guide the user through the input process. The layout is organized using the GridLayout manager, ensuring a clean and responsive design.
2. **Core Logic**:  
   The application's core logic is implemented through modular methods for each arithmetic operation. These methods (add, subtract, multiply, and divide) are reusable and encapsulated to promote clarity and scalability. The logic validates user inputs, processes the calculations, and handles specific scenarios like division by zero using exception handling.
3. **Event Handling**:  
   The application uses ActionListeners to handle button clicks, linking user interactions with the appropriate logic. When a button is pressed, the corresponding method is called to process the input values and compute the result. The result is then displayed in the output area of the GUI.
4. **Error Handling and Validation**:  
   Input validation ensures that users enter valid numeric data. The application manages common errors like invalid input formats (e.g., letters instead of numbers) and division by zero gracefully, displaying appropriate error messages to guide the user.
5. **Execution and Display**:  
   When the program is executed, a JFrame window appears, displaying the calculator's interface. Users can input values, select an operation, and view the results dynamically.

**SOURCE CODE:**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class CalculatorApp {

    // Core calculator logic

    public static double add(double a, double b) {

        return a + b;

    }

    public static double subtract(double a, double b) {

        return a - b;

    }

    public static double multiply(double a, double b) {

        return a \* b;

    }

    public static double divide(double a, double b) throws ArithmeticException {

        if (b == 0) {

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

        }

        return a / b;

    }

    public static void main(String[] args) {

        // Create JFrame

        JFrame frame = new JFrame("Arithmetic Operations Application");

        frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        frame.setSize(400, 300);

        frame.setLayout(new GridLayout(6, 2, 5, 5));

        // Input fields and labels

        JLabel label1 = new JLabel("Enter first number:");

        JTextField field1 = new JTextField();

        JLabel label2 = new JLabel("Enter second number:");

        JTextField field2 = new JTextField();

        JLabel resultLabel = new JLabel("Result:");

        JLabel resultValue = new JLabel();

        // Buttons for operations

        JButton addButton = new JButton("Add");

        JButton subtractButton = new JButton("Subtract");

        JButton multiplyButton = new JButton("Multiply");

        JButton divideButton = new JButton("Divide");

        // Adjust button sizes

        Dimension buttonSize = new Dimension(100, 30);

        addButton.setPreferredSize(buttonSize);

        subtractButton.setPreferredSize(buttonSize);

        multiplyButton.setPreferredSize(buttonSize);

        divideButton.setPreferredSize(buttonSize);

        // ActionListener for buttons

        ActionListener listener = new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                try {

                    double num1 = Double.parseDouble(field1.getText());

                    double num2 = Double.parseDouble(field2.getText());

                    double result = 0;

                    if (e.getSource() == addButton) {

                        result = add(num1, num2);

                    } else if (e.getSource() == subtractButton) {

                        result = subtract(num1, num2);

                    } else if (e.getSource() == multiplyButton) {

                        result = multiply(num1, num2);

                    } else if (e.getSource() == divideButton) {

                        result = divide(num1, num2);

                    }

                    resultValue.setText(String.valueOf(result));

                } catch (NumberFormatException ex) {

                    resultValue.setText("Invalid input!");

                } catch (ArithmeticException ex) {

                    resultValue.setText(ex.getMessage());

                }

            }

        };

        addButton.addActionListener(listener);

        subtractButton.addActionListener(listener);

        multiplyButton.addActionListener(listener);

        divideButton.addActionListener(listener);

        // Add components to frame

        frame.add(label1);

        frame.add(field1);

        frame.add(label2);

        frame.add(field2);

        frame.add(addButton);

        frame.add(subtractButton);

        frame.add(multiplyButton);

        frame.add(divideButton);

        frame.add(resultLabel);

        frame.add(resultValue);

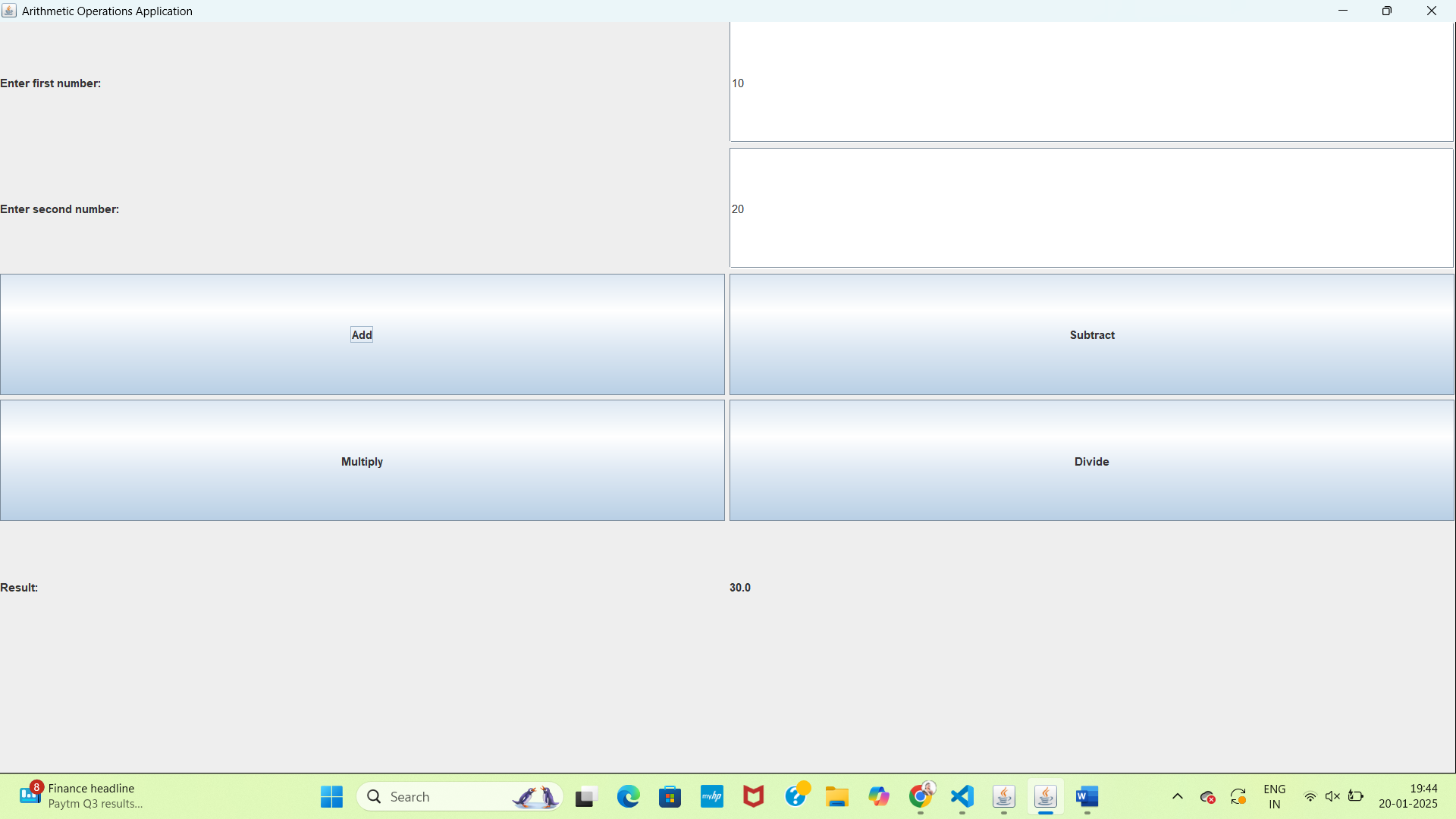
        // Display the frame

        frame.setVisible(true);

    }

}

**RESULT**



**CONCLUSION:**

The **Java Arithmetic Calculator Application** successfully demonstrates the implementation of a simple yet effective desktop-based calculator using the Java programming language and the Swing framework. The project highlights the application of core Java concepts, such as modular programming, event-driven design, and exception handling, to create a user-friendly and functional tool.

Through its intuitive graphical user interface, the application allows users to perform basic arithmetic operations seamlessly. The integration of robust error handling and input validation ensures the reliability of the application, providing accurate results even in cases of invalid inputs or exceptional scenarios such as division by zero. The modular design of the application enhances maintainability and facilitates future scalability, making it adaptable for further enhancements like advanced operations or data persistence.

Overall, this project serves as a practical example of how Java can be used to design and develop interactive and efficient desktop applications. It is a valuable learning tool for aspiring developers and a stepping stone toward mastering more complex Java-based applications.

**FUTURE ENHANCEMENTS**

The **Java Arithmetic Calculator Application** can be extended and improved in several ways to enhance its functionality, usability, and scalability. Some potential future enhancements include:

1. **Advanced Mathematical Operations**:  
   Add support for advanced functions like square roots, exponentiation, logarithms, trigonometric functions (sin, cos, tan), and factorials to make the calculator suitable for scientific calculations.
2. **History Tracking**:  
   Implement a feature to record the history of calculations performed by the user. This could be displayed within the application or saved to a file for future reference.
3. **Data Persistence**:  
   Integrate a database (e.g., SQLite) to store user preferences, calculation history, or application settings, enabling persistent storage and retrieval of data.
4. **Exporting Results**:  
   Provide functionality to export calculation results or history to external formats such as CSV, Excel (using Apache POI), or JSON files for documentation or sharing purposes.
5. **Improved User Interface**:  
   Enhance the GUI by adopting JavaFX for more modern and visually appealing designs, including theming, animations, and responsive layouts for different screen sizes.
6. **Multi-Language Support**:  
   Incorporate localization features to allow the application to support multiple languages, improving accessibility for users from diverse linguistic backgrounds.
7. **Error Visualization**:  
   Highlight input errors more effectively, such as color-coded text fields or pop-up dialogs, to improve user experience when invalid inputs are entered.
8. **Mobile Compatibility**:  
   Develop a mobile-friendly version of the calculator using frameworks like Android Studio, allowing the application to reach a broader audience.
9. **Voice Input and Output**:  
   Enable voice recognition to accept inputs and perform operations through voice commands, as well as text-to-speech functionality to read out results.
10. **Customization Options**:  
    Allow users to customize the application, such as selecting themes, configuring button layouts, or enabling/disabling specific operations.