Abstract:

This project presents a graphical user interface (GUI) calculator developed in Java, utilizing the Swing framework for interactive user experience. The calculator supports basic arithmetic operations, including addition, subtraction, multiplication, division, and modulus, enabling users to perform calculations efficiently.

The application features a clean and intuitive layout, displaying a text field for input and results, along with buttons for numbers (0-9), arithmetic operations, a clear function, and backspace functionality. Each button is implemented with action listeners to capture user input and execute calculations in real-time.

Key functionalities include:

- **Basic Operations**: Users can perform fundamental arithmetic operations and view results instantly.
- **Input Handling**: The calculator can process multiple digits and decimal inputs, with error handling for invalid operations, such as division by zero.
- **Dynamic User Interaction**: The GUI updates seamlessly as users press buttons, enhancing the overall user experience.

This project serves as an educational tool for understanding event-driven programming in Java and demonstrates the principles of GUI design and implementation. It aims to provide a functional and user-friendly calculator application suitable for both educational purposes and everyday use.

Objective

The objective of this project is to design and implement a functional graphical user interface (GUI) calculator using Java and the Swing framework. The calculator aims to provide users with an intuitive and user-friendly platform for performing basic arithmetic operations, including addition, subtraction, multiplication, division, and modulus.

Key objectives include:

1. User Interaction: Create an interactive GUI that allows users to input numbers and operations easily through buttons.
2. Real-Time Calculations: Enable immediate feedback and display results as users perform calculations, enhancing usability.
3. Robust Input Handling: Implement error handling for invalid operations, such as division by zero, to ensure reliability and prevent crashes.

4. Educational Tool: Serve as a practical demonstration of event-driven programming concepts and GUI design in Java, making it suitable for educational purposes.

The ultimate goal is to deliver a reliable and efficient calculator application that can be used for everyday calculations while providing insights into Java programming and GUI development.

Introduction

In the digital age, calculators are essential tools that facilitate mathematical computations in both personal and professional settings. This project focuses on developing a graphical user interface (GUI) calculator using Java's Swing framework, aiming to combine functionality with an intuitive user experience.

The calculator will support fundamental arithmetic operations such as addition, subtraction, multiplication, division, and modulus, allowing users to perform calculations with ease. By employing a simple yet effective design, the application will cater to a diverse audience, including students, professionals, and anyone in need of quick mathematical solutions.

Java is an ideal choice for this project due to its platform independence, extensive libraries, and strong community support. The Swing framework, in particular, provides robust tools for building interactive interfaces, enabling the creation of a visually appealing and user-friendly application.

This project not only serves as a practical utility for users but also as an educational opportunity to explore core programming concepts, such as event handling, layout management, and GUI design principles. Through this endeavor, we aim to enhance our understanding of Java programming while delivering a useful and engaging tool for everyday calculations.

Methodology

The development of the Java GUI calculator will follow a structured approach, encompassing the following key phases:

1. **Requirement Analysis**:
    - Identify and define the functional and non-functional requirements of the calculator. This includes determining the basic arithmetic operations to be supported (addition, subtraction, multiplication, division, modulus) and the desired user interface features.
2. **Design**:
    - **User Interface Design**: Create a layout for the GUI using Java Swing components, such as JFrame, JTextField, and JButton. The

design will prioritize usability, ensuring that buttons are clearly labeled and arranged logically to enhance user experience.
    - **System Architecture**: Establish the overall architecture of the application, defining how different components will interact. This includes planning the event-driven programming model where button clicks trigger specific actions.
3. **Implementation**:
    - **Coding**: Develop the application using Java. This involves:
        - Creating the main frame and text field for displaying input and results.
        - Implementing buttons for numbers, operations, and special functions (clear, backspace).
        - Writing event listeners for buttons to handle user interactions and perform calculations.
    - **Error Handling**: Implement mechanisms to handle invalid inputs (e.g., division by zero, non-numeric input) gracefully, providing user feedback as needed.
4. **Testing**:
    - Conduct thorough testing of the application to ensure all functionalities work as intended. This will include:
        - Unit testing individual components and functions to validate correctness.
        - User acceptance testing to gather feedback on usability and interface design.
    - Identify and fix any bugs or usability issues that arise during testing.
5. **Deployment**:
    - Prepare the application for deployment by packaging it into an executable format. Ensure that the application runs smoothly on different operating systems, leveraging Java's platform independence.
6. **Documentation**:
    - Create user documentation detailing how to use the calculator, including explanations of its features and functionalities.
    - Prepare technical documentation for future developers, covering the code structure, design decisions, and any third-party libraries used.
7. **Maintenance**:
    - Plan for ongoing maintenance and updates to address any issues that arise post-deployment and to potentially add new features based on user feedback.

This structured approach ensures a systematic development process, facilitating efficient project management and the delivery of a functional and user-friendly calculator application.

Code :

```
ame frame;
	private JTextField textField;
	double first_No;
	double second_No;
	String operation;
	Double result;
	String f_Answer;

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					Cal window = new Cal();
					window.frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	/**
	 * Create the application.
	 */
	public Cal() {
		initialize();
	}

	/**
	 * Initialize the contents of the frame.
	 */
	private void initialize() {
		frame = new JFrame();
		frame.getContentPane().setLayout(null);

		textField = new JTextField();
		textField.setBounds(10, 11, 322, 58);
		frame.getContentPane().add(textField);
		textField.setColumns(10);

		JButton btnNewButton = new JButton("\uF0E7");
		btnNewButton.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				String bs = null;
				if(textField.getText().length()>0)
				{
					StringBuilder str=new
StringBuilder(textField.getText());
					str.deleteCharAt(textField.getText().length()-1);
					bs=str.toString();
					textField.setText(bs);
				}

			}
		});
		btnNewButton.setFont(new Font("Windings",Font.BOLD,17));
```

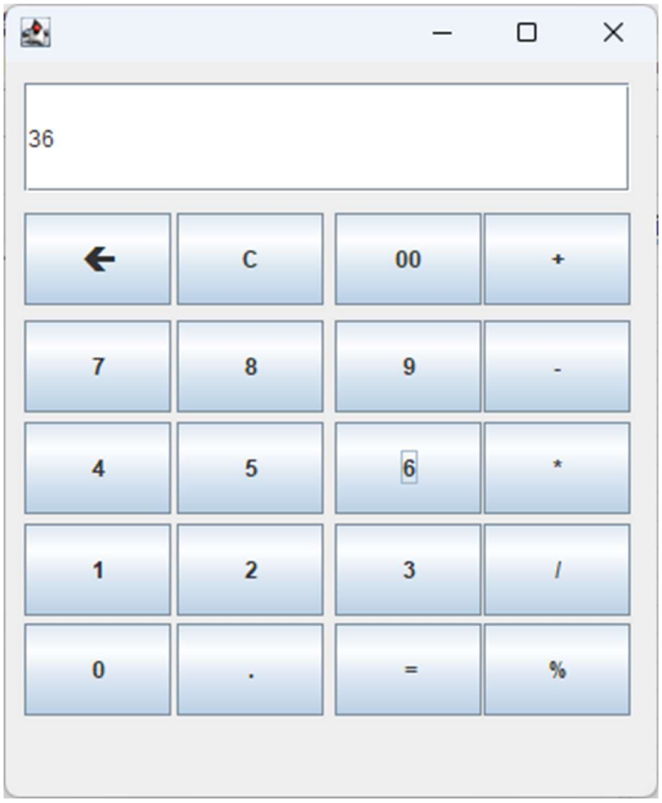
```
		btnNewButton.setBounds(10, 80, 78, 49);
		frame.getContentPane().add(btnNewButton);

		JButton btnNewButton_1 = new JButton("7");
		btnNewButton_1.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				String
num=textField.getText()+btnNewButton_1.getText();
				textField.setText(num);
			}
		});
		btnNewButton_1.setBounds(10, 137, 78, 49);
		frame.getContentPane().add(btnNewButton_1);

		JButton btnNewButton_2 = new JButton("4");
		btnNewButton_2.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				String
num=textField.getText()+btnNewButton_2.getText();
				textField.setText(num);
			}
		});
		btnNewButton_2.setBounds(10, 191, 78, 49);
		frame.getContentPane().add(btnNewButton_2);

		JButton btnNewButton_3 = new JButton("1");
		btnNewButton_3.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				String
num=textField.getText()+btnNewButton_3.getText();
				textField.setText(num);
			}
		});
		btnNewButton_3.setBounds(10, 245, 78, 49);
		frame.getContentPane().add(btnNewButton_3);

		JButton btnNewButton_4 = new JButton("0");
		btnNewButton_4.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				String
num=textField.getText()+btnNewButton_4.getText();
				textField.setText(num);
			}
		});
		btnNewButton_4.setBounds(10, 298, 78, 49);
		frame.getContentPane().add(btnNewButton_4);

		JButton btnC = new JButton("C");
		btnC.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				textField.setText(null);
			}
		});
		btnC.setBounds(91, 80, 78, 49);
		frame.getContentPane().add(btnC);

		JButton btnNewButton_6 = new JButton("8");
		btnNewButton_6.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
```

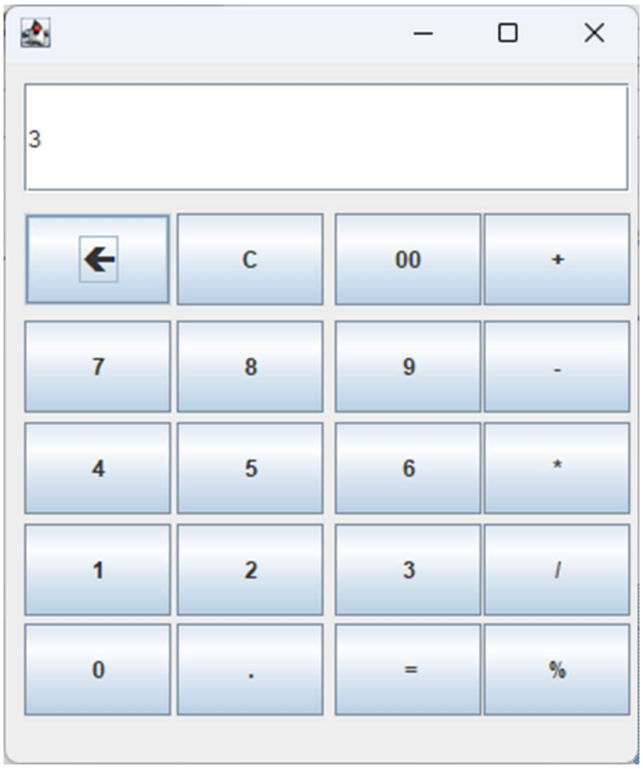
```
                        String
num=textField.getText()+btnNewButton_6.getText();
                        textField.setText(num);
                }
        });
        btnNewButton_6.setBounds(91, 137, 78, 49);
        frame.getContentPane().add(btnNewButton_6);

        JButton btnNewButton_7 = new JButton("5");
        btnNewButton_7.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                        String
num=textField.getText()+btnNewButton_7.getText();
                        textField.setText(num);
                }
        });
        btnNewButton_7.setBounds(91, 191, 78, 49);
        frame.getContentPane().add(btnNewButton_7);

        JButton btnNewButton_8 = new JButton("2");
        btnNewButton_8.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                        String
num=textField.getText()+btnNewButton_8.getText();
                        textField.setText(num);
                }
        });
        btnNewButton_8.setBounds(91, 245, 78, 49);
        frame.getContentPane().add(btnNewButton_8);

        JButton btnNewButton_9 = new JButton(".");
        btnNewButton_9.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                        String
num=textField.getText()+btnNewButton_9.getText();
                        textField.setText(num);
                }
        });
        btnNewButton_9.setBounds(91, 298, 78, 49);
        frame.getContentPane().add(btnNewButton_9);

        JButton btnNewButton_10 = new JButton("00");
        btnNewButton_10.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                        String
num=textField.getText()+btnNewButton_10.getText();
                        textField.setText(num);
                }
        });
        btnNewButton_10.setBounds(175, 80, 78, 49);
        frame.getContentPane().add(btnNewButton_10);

        JButton btnNewButton_11 = new JButton("9");
        btnNewButton_11.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                        String
num=textField.getText()+btnNewButton_11.getText();
                        textField.setText(num);
                }
```

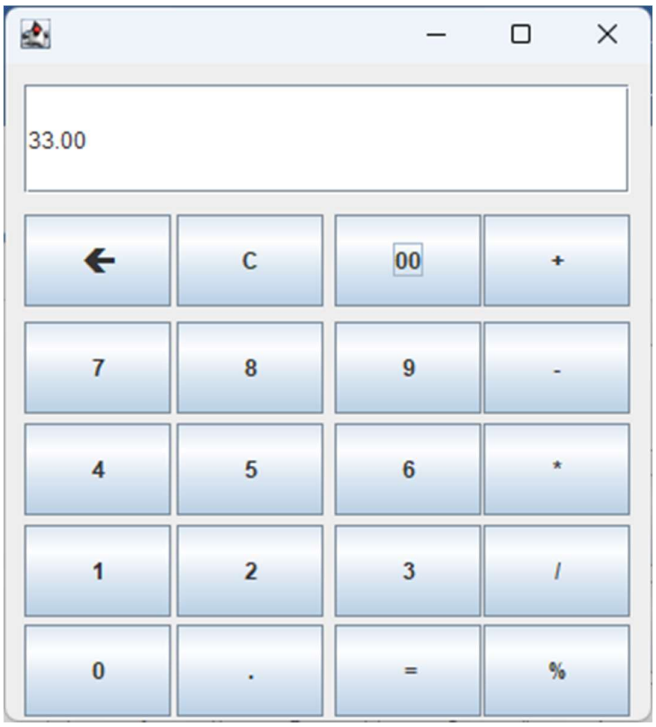
```
		});
		btnNewButton_11.setBounds(175, 137, 78, 49);
		frame.getContentPane().add(btnNewButton_11);

		JButton btnNewButton_12 = new JButton("6");
		btnNewButton_12.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				String
num=textField.getText()+btnNewButton_12.getText();
				textField.setText(num);
			}
		});
		btnNewButton_12.setBounds(175, 191, 78, 49);
		frame.getContentPane().add(btnNewButton_12);

		JButton btnNewButton_13 = new JButton("3");
		btnNewButton_13.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				String
num=textField.getText()+btnNewButton_13.getText();
				textField.setText(num);
			}
		});
		btnNewButton_13.setBounds(175, 245, 78, 49);
		frame.getContentPane().add(btnNewButton_13);

		JButton btnNewButton_14 = new JButton(" =");
		btnNewButton_14.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				second_No=Double.parseDouble(textField.getText());
				if(operation=="+")
				{
					result=first_No+ second_No;
					f_Answer=String.format("%.2f",result);
					textField.setText(f_Answer);
				}
					else if(operation=="-")
					{
						result=first_No - second_No;
						f_Answer=String.format("%.2f",result);
						textField.setText(f_Answer);
					}
					else if(operation=="*")
					{
						result=first_No*second_No;
						f_Answer=String.format("%.2f",result);
						textField.setText(f_Answer);
					}
					else if(operation=="/")
					{
						result=first_No/ second_No;
						f_Answer=String.format("%.2f",result);
						textField.setText(f_Answer);
					}
					else if(operation=="%")
					{
						result=first_No % second_No;
						f_Answer=String.format("%.2f",result);
						textField.setText(f_Answer);
```

```
                }
            }

});
btnNewButton_14.setBounds(175, 298, 78, 49);
frame.getContentPane().add(btnNewButton_14);

JButton btnNewButton_15 = new JButton("+");
btnNewButton_15.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
		first_No=Double.parseDouble(textField.getText());
		textField.setText("");
		operation="+";
	}
});
btnNewButton_15.setBounds(254, 80, 78, 49);
frame.getContentPane().add(btnNewButton_15);

JButton btnNewButton_16 = new JButton("-");
btnNewButton_16.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
		first_No=Double.parseDouble(textField.getText());
		textField.setText("");
		operation="-";
	}
});
btnNewButton_16.setBounds(254, 137, 78, 49);
frame.getContentPane().add(btnNewButton_16);

JButton btnNewButton_17 = new JButton("*");
btnNewButton_17.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
		first_No=Double.parseDouble(textField.getText());
		textField.setText("");
		operation="*";
	}
});
btnNewButton_17.setBounds(254, 191, 78, 49);
frame.getContentPane().add(btnNewButton_17);

JButton btnNewButton_18 = new JButton("/");
btnNewButton_18.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
		first_No=Double.parseDouble(textField.getText());
		textField.setText("");
		operation="/";
	}
});
btnNewButton_18.setBounds(254, 245, 78, 49);
frame.getContentPane().add(btnNewButton_18);

JButton btnNewButton_19 = new JButton("%");
btnNewButton_19.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
		first_No=Double.parseDouble(textField.getText());
		textField.setText("");
		operation="%";
	}
});
```

```
            btnNewButton_19.setBounds(254, 298, 78, 49);
            frame.getContentPane().add(btnNewButton_19);
        }
}
```







Conclusion

The Java GUI calculator project successfully demonstrates the application of fundamental programming concepts and design principles in creating a functional and user-friendly software tool. By utilizing Java Swing for the graphical user interface, we have developed a calculator that supports basic arithmetic operations, including addition, subtraction, multiplication, division, and modulus.

Throughout the development process, we emphasized user experience by ensuring a clear layout and intuitive navigation. The event-driven programming model effectively handles user interactions, allowing for real-time input and immediate feedback. Rigorous testing phases confirmed the application's reliability and functionality, enabling us to address potential issues before deployment.

This project not only reinforces our understanding of Java programming and GUI design but also highlights the importance of a structured methodology in software development. Future enhancements could include advanced features such as memory functions, scientific calculations, and improved error handling to accommodate a wider range of user needs. Overall, the project serves as a solid foundation for further exploration in software development, showcasing the potential for creating practical applications using Java.