**Project 1 - Calculator Built in Java**

**Task : Create a Native Calculator using Java and Java SWING GUI**

1. ABSTRACT :-

The Java Swing Calculator project focuses on the development of a graphical user interface (GUI)-based calculator using Java and the Swing framework. This project aims to provide users with an interactive, efficient, and visually appealing application capable of performing essential arithmetic operations, including addition, subtraction, multiplication, and division. Unlike traditional command-line calculators that rely on text-based input, this Swing-based implementation allows users to interact with graphical elements such as buttons and input fields, making mathematical computations more intuitive and accessible. The incorporation of a well-structured GUI ensures seamless navigation, enhancing user experience by providing an interactive and error-free approach to calculations.

One of the key aspects of this project is the demonstration of event-driven programming. Each user interaction with the calculator, such as clicking numeric buttons or arithmetic operators, triggers corresponding events that result in real-time calculations. This ensures that all computations are processed dynamically, improving efficiency and responsiveness. Java Swing components, including JButton, JTextField, and JFrame, play a crucial role in building this calculator by facilitating interactive elements that maintain a structured and lightweight design. The project prioritizes usability, ensuring that the layout is intuitive, responsive, and capable of handling real-time arithmetic operations with minimal delay.

Error handling is another fundamental feature incorporated into this calculator to improve its reliability. Invalid computations, such as division by zero or multiple consecutive operators, are prevented through rigorous validation techniques. This ensures that users receive accurate and meaningful feedback rather than encountering abrupt errors that might disrupt calculations. To evaluate mathematical expressions dynamically, the application leverages JavaScript’s ScriptEngine, which allows flexible parsing and execution of arithmetic operations. This approach simplifies the handling of calculations while maintaining precision in numerical outputs. Additionally, formatting mechanisms, such as the use of DecimalFormat, ensure that results remain readable by restricting excessive decimal places and unnecessary rounding errors.

This project is built with structured modular coding principles, allowing for easy maintenance and scalability. Cross-platform compatibility is ensured by leveraging Java’s platform-independent nature, making the calculator accessible across different operating systems without modifications to its core functionality. The modular approach also enables future enhancements, allowing developers to extend its capabilities by incorporating more complex operations. Beyond basic arithmetic functionality, potential improvements include integrating scientific features such as trigonometric functions, logarithmic calculations, and factorial operations. These additions would further enhance the calculator’s capabilities, making it a versatile computational tool for various users.

A significant advantage of developing this calculator within the Java Swing framework is the ability to create a visually appealing interface that goes beyond mere functionality. The graphical elements are carefully structured to provide an engaging experience, ensuring that users can easily navigate through the buttons and display field. The implementation of the GridLayout and FlowLayout frameworks ensures uniform button arrangement, improving accessibility and usability. The calculator’s display field dynamically updates values based on user interactions, allowing seamless input recognition and expression evaluation.

The integration of JavaScript’s ScriptEngine for dynamic evaluation offers an effective way of processing mathematical expressions within the Java environment. This engine interprets user-provided input as mathematical expressions and computes the results accordingly. Unlike traditional manual arithmetic processing, where each operator and operand must be individually handled, the ScriptEngine automates the evaluation process, reducing code complexity while maintaining computational integrity. The ability to execute mathematical expressions dynamically improves the flexibility of operations, enabling users to enter longer expressions that can be parsed and solved instantly.

Future improvements to this calculator could introduce a range of additional functionalities to enhance usability and efficiency. Scientific calculations, such as sine, cosine, tangent, logarithms, and exponentials, could be integrated into the application, expanding its usefulness to a broader audience, including students, engineers, and researchers. Memory storage features could also be implemented to allow users to save previous calculations and retrieve stored values for further computations. These improvements would significantly enhance the calculator’s ability to handle complex mathematical operations beyond basic arithmetic.

Another potential upgrade includes transitioning from Java Swing to JavaFX to further improve graphical design. JavaFX provides enhanced graphical capabilities, allowing for smoother animations, scalable UI elements, and modern interface aesthetics. This transition would not only refine the visual appeal of the calculator but also make it more adaptable to mobile applications, where touch-based interactions could be supported. By integrating JavaFX into future iterations of the project, the calculator could become more versatile and appealing to users who require a modern computational tool.

The structured documentation of this project ensures clarity in development processes, making it easier for future contributors to understand and modify the application. The report explores best practices in GUI-based development within Java, focusing on usability, efficiency, and maintainability. Methodology, objectives, implementation details, and scalability considerations are thoroughly examined to provide an in-depth understanding of the project’s execution. The calculator serves as an excellent learning tool for developers who are keen on mastering event-driven programming, GUI-based application design, and mathematical computation handling within Java.

The Java Swing Calculator project is an excellent example of how GUI-based applications improve user interaction and computational efficiency. By eliminating the need for text-based input and replacing it with visually structured elements, the calculator provides an intuitive approach to arithmetic processing. The well-defined layout, responsive elements, and error-handling mechanisms contribute to a reliable and easy-to-use application. Additionally, the integration of dynamic expression evaluation ensures that calculations are processed efficiently, providing users with accurate results in real-time.

As technology continues to evolve, graphical calculators remain essential tools for both professional and educational purposes. This Java Swing implementation demonstrates the capabilities of GUI-based applications in the field of computational mathematics while serving as a stepping stone for future enhancements. The calculator’s modular structure allows ongoing improvements, ensuring its adaptability to newer features that meet the growing demands of users. Whether used for simple arithmetic or expanded into a scientific computational system, this project showcases the power of structured programming, graphical design, and interactive application development within Java.

In conclusion, the Java Swing Calculator project successfully implements an intuitive and interactive computational tool that enhances arithmetic processing through graphical elements and dynamic evaluation mechanisms. Its modular coding structure ensures easy maintenance and scalability, making it suitable for future improvements. By focusing on usability, accuracy, and event-driven programming, this project stands as a valuable educational resource for those exploring GUI development in Java. The potential expansions, including scientific calculations, memory storage, and graphical enhancements, further highlight its adaptability as a versatile computational tool. The insights gained from this project reinforce the importance of structured programming and efficient software design, offering a practical application of Java-based graphical computation.

1. OBJECTIVE :-

The primary objective of the Java Swing Calculator project is to develop an interactive and efficient computational tool that allows users to perform fundamental arithmetic operations seamlessly. By leveraging Java Swing components, this calculator provides a structured graphical user interface (GUI) that enhances usability and responsiveness. The project aims to create a system that simplifies numerical input processing, ensuring error-free execution while maintaining a visually appealing layout. Through structured programming principles and event-driven mechanisms, the calculator delivers a well-integrated approach to handling real-time calculations.

A significant goal of this project is to demonstrate the implementation of graphical user interface elements within Java, enabling users to interact with an intuitive digital calculator rather than relying on traditional command-line computation. This objective aligns with the broader aim of making mathematical operations more accessible to users who may not be familiar with programming or manual input processing. The GUI-based approach ensures that users can engage with the system dynamically, entering numbers, performing operations, and receiving instant feedback through a structured display interface.

Another critical aspect of this project is ensuring accurate computational processing. The calculator must handle basic arithmetic operations—addition, subtraction, multiplication, and division—without errors. Utilizing JavaScript’s ScriptEngine, the system interprets expressions dynamically, allowing users to input complete mathematical statements that are parsed and computed instantaneously. This technique simplifies the complexity associated with handling individual arithmetic operations manually, enabling efficient and streamlined computation. By automating the parsing and execution of expressions, the system improves accuracy and enhances user interaction.

Error handling plays a fundamental role in achieving the objective of creating a reliable calculator. Ensuring that invalid entries do not disrupt operations is key to maintaining usability. The project incorporates validation mechanisms to prevent issues such as multiple consecutive operators, division by zero, and other faulty expressions. These safeguards ensure that the calculator responds appropriately to incorrect inputs by displaying meaningful error messages rather than executing unpredictable computations. The ability to prevent invalid sequences further reinforces the reliability of the calculator, making it an efficient computational tool for various users.

One of the core objectives of this project is to develop a lightweight yet scalable application. The calculator must operate with minimal system resource consumption while remaining responsive across different platforms. Java's inherent cross-platform compatibility makes it an ideal choice for ensuring smooth execution on various operating systems, including Windows, Linux, and macOS. By optimizing memory usage and streamlining computational execution, the system achieves a balance between efficiency and performance. The goal is to maintain a fast and reliable calculator that provides seamless operation without excessive system overhead.

Another essential objective is the usability and accessibility of the application. The graphical interface must be designed with clarity in mind, ensuring that users can navigate through buttons effortlessly. The arrangement of buttons follows a structured layout, enabling intuitive interaction while minimizing confusion. The GridLayout approach ensures proper spacing and uniform organization, making mathematical operations straightforward. Display fields must dynamically update values, providing real-time feedback to users as they enter numbers and perform calculations. The calculator’s simplicity in navigation contributes to its effectiveness as a digital computation tool.

Beyond basic arithmetic, the calculator is structured to support future expansion. The project aims to provide a scalable framework where additional functionalities can be incorporated. Possible enhancements include scientific operations, logarithmic calculations, trigonometric functions, and statistical analysis. These expansions can be integrated seamlessly within the existing modular code architecture, making the calculator adaptable to varying computational needs. By designing the system with scalability in mind, future modifications can be implemented efficiently without disrupting existing functionality.

Another goal of the project is to explore different graphical enhancements, potentially transitioning from Java Swing to JavaFX for better visual aesthetics. JavaFX provides advanced graphical capabilities, enabling smoother animations and improved UI responsiveness. By implementing JavaFX, the calculator could be made more visually sophisticated, incorporating modern design elements that enhance the user experience. Furthermore, transitioning to JavaFX would allow mobile compatibility, enabling touch-based interactions for smartphone and tablet applications.

In addition to expanding functionality, integrating memory storage features is an essential future objective. Allowing users to store and retrieve previous calculations improves usability, particularly for applications requiring multiple-step computations. Features such as memory recall, history tracking, and persistent storage would significantly increase the practicality of the calculator, making it a versatile computational tool for academic and professional use.

The structured modular approach taken in this project aligns with best practices in software development, ensuring maintainability and long-term usability. The codebase is designed for clarity, allowing future contributors to modify and expand the system efficiently. By adopting structured programming principles, the calculator remains organized, reducing complexity in future developments. The emphasis on modularity ensures that individual components, such as button interaction logic, display field management, and error-handling mechanisms, can be adjusted or extended independently.

The Java Swing Calculator serves as an excellent educational project for developers looking to understand GUI-based programming in Java. It highlights the importance of event-driven programming, illustrating how interactive elements can be used to trigger mathematical operations dynamically. Through the implementation of structured graphical components, the project showcases the practical application of Java’s capabilities in building user-friendly computational tools. The system’s integration of JavaScript’s ScriptEngine demonstrates how mathematical expressions can be processed efficiently within a Java environment, reinforcing the practical value of combining multiple programming techniques.

Usability testing and optimization form another critical objective of this project. The calculator must undergo thorough testing to verify the correctness of arithmetic operations and ensure error-free execution. Performance optimizations focus on reducing unnecessary computations, maintaining quick response times, and preventing lag in button interactions. By rigorously testing different input scenarios and identifying potential bottlenecks, the project aims to deliver a highly responsive and reliable mathematical tool.

In summary, the objectives of the Java Swing Calculator project revolve around creating an efficient, interactive, and reliable computational system that enhances arithmetic processing through graphical user interface components. The project focuses on usability, accuracy, scalability, and future enhancements, ensuring that the system remains adaptable for additional functionalities. By implementing structured programming principles, event-driven mechanisms, and modular code architecture, the calculator provides a valuable educational tool for learning GUI-based application development in Java. The potential for expansion into scientific calculations, memory storage, and graphical enhancements underscores its long-term viability as a versatile computing application. With a focus on user interaction and system efficiency, the Java Swing Calculator successfully meets its objectives, offering a practical demonstration of Java-based graphical computation.

1. INTRODUCTION :-

The Java Swing Calculator project is an endeavor that merges mathematical computation with an intuitive graphical user interface (GUI), ensuring a seamless user experience. This project is designed to perform essential arithmetic operations such as addition, subtraction, multiplication, and division while providing an interactive environment that enhances usability and accessibility. The choice to implement this calculator using Java Swing stems from its flexibility and ability to create lightweight yet visually appealing applications suitable for both beginners and advanced users. Unlike traditional command-line calculators, which require users to input text commands manually, this GUI-based approach offers a more engaging and user-friendly interaction.

Mathematical calculations have always been an integral part of human activities, from everyday budgeting to complex scientific research. While physical calculators have long been used to handle computations, the advent of software-based calculators has revolutionized how users engage with numerical problem-solving. GUI-based calculators remove the need for memorizing commands or learning syntax, instead allowing users to simply press buttons that correspond to their desired arithmetic operations. This project aims to create a reliable, efficient, and visually structured digital tool that ensures accuracy while maintaining ease of use.

Java Swing is a highly effective framework for building GUI applications due to its component-based structure. It allows developers to implement buttons, text fields, labels, and interactive elements within a structured layout. The Java Swing Calculator harnesses this functionality to provide users with a well-organized interface where numbers and operators can be selected with ease. By employing event-driven programming principles, the calculator responds dynamically to user inputs, executing operations and returning instant results. This improves computational efficiency, allowing users to engage with the application naturally, as they would with a physical calculator.

The significance of GUI-based applications extends beyond simple convenience. In software development, user experience plays a crucial role in determining the success of an application. A well-designed GUI not only enhances visual appeal but also ensures smooth interaction. This project focuses on maintaining a balanced interface that is neither overly complex nor too simplistic. Through a structured layout, numeric inputs, arithmetic symbols, and evaluation mechanisms are arranged systematically, preventing confusion while enabling quick calculations.

Developing a calculator in Java Swing involves careful planning and implementation of event-driven logic. Unlike traditional console-based applications, which rely on command inputs and outputs, GUI-based applications function through event listeners that trigger actions based on user interactions. In this calculator, each button represents an arithmetic function, a numeric value, or an auxiliary operation such as clearing the display or executing calculations. When a user clicks a button, an event is generated and handled by the program to update the display accordingly.

Error handling is another crucial aspect of this project. In any application involving numerical computations, ensuring accuracy is essential. Invalid operations, such as division by zero or multiple consecutive operators, can lead to incorrect results or application failures. By implementing robust validation techniques, this calculator prevents users from entering incorrect expressions, thereby maintaining stability. The inclusion of meaningful error messages informs users about invalid inputs, allowing them to adjust their calculations without confusion. This feature significantly enhances reliability, making the calculator a dependable tool for basic arithmetic.

A noteworthy feature of this project is its dynamic expression evaluation using JavaScript’s ScriptEngine. Rather than manually handling arithmetic operations through individual method calls, this calculator leverages ScriptEngine to process expressions in real-time. When users input a complete mathematical statement, the engine interprets and executes the calculation, delivering instant results. This approach simplifies code structure while improving efficiency, allowing for flexible and accurate processing of user inputs.

The graphical elements in this calculator are designed to prioritize usability. Java Swing components such as JButton, JTextField, and JFrame play a fundamental role in creating the interactive interface. Buttons are arranged in a grid layout, ensuring logical positioning of numbers and arithmetic operators. The text field serves as both the input and output display, updating dynamically based on user actions. These components work cohesively to deliver a smooth, responsive experience that enhances computation efficiency

One of the overarching goals of this project is to create a scalable and adaptable system. While the initial version of the calculator focuses on basic arithmetic, future developments can introduce more advanced mathematical functions. Scientific calculations, including logarithmic operations, trigonometric functions, and exponential computations, could be integrated within the existing framework. The calculator’s modular structure allows easy expansion, making it a potential candidate for further enhancement beyond fundamental arithmetic.

Another possible evolution of this project is transitioning from Java Swing to JavaFX. While Swing provides excellent functionality for desktop applications, JavaFX introduces more sophisticated graphical capabilities. With features such as smoother animations, improved responsiveness, and better UI components, JavaFX could enhance the overall look and feel of the calculator. Moreover, JavaFX’s compatibility with touch-based interfaces allows for mobile adaptability, enabling users to interact with the calculator on tablets and smartphones.

Memory storage features could also be incorporated to improve usability. Many advanced calculators offer memory recall functions that allow users to store previous calculations for future reference. By integrating a memory system within this calculator, users would have the ability to track their previous computations, retrieve stored values, and perform further operations using recalled numbers. This would significantly increase the utility of the application, making it more than just a simple arithmetic tool.

As a learning project, the Java Swing Calculator provides insights into structured programming, GUI development, and event-driven logic. It serves as an excellent introduction to Java’s capabilities, showcasing how graphical applications can enhance usability while maintaining computational integrity. Beginners can use this project to familiarize themselves with Swing components, while experienced developers can modify and expand its functionality according to specific requirements.

Performance optimization is another key factor in ensuring the calculator operates efficiently. Since GUI-based applications rely on event handling and graphical rendering, maintaining responsiveness is essential. The project incorporates efficient memory management techniques to prevent unnecessary resource consumption, ensuring smooth execution across different system environments. The responsiveness of the application is carefully tested to avoid lag in button interactions, ensuring a fluid experience for users.

Testing plays a vital role in validating the accuracy and reliability of this calculator. Before deployment, multiple test scenarios are conducted to verify proper execution of arithmetic operations. Each function, from numeric input handling to final calculation processing, undergoes rigorous checks to ensure correctness. Edge cases such as division by zero, invalid expressions, and extreme numerical values are assessed to verify that error-handling mechanisms function appropriately. This testing phase ensures that users receive accurate results without encountering unexpected issues.

The ultimate goal of this project is to create a well-rounded, interactive calculator that provides accurate computations while maintaining ease of use. Through careful implementation of Java Swing components, event-driven logic, and error-handling techniques, this project successfully demonstrates the principles of GUI-based application development. The calculator is not only a functional arithmetic tool but also a foundational project for learning structured programming and software design.

As technological advancements continue to improve digital applications, GUI-based systems are becoming more prevalent. The Java Swing Calculator project showcases how graphical interfaces enhance usability and interactivity, making computational tools more intuitive and user-friendly. Its modular structure ensures future expandability, allowing additional features to be integrated seamlessly. Whether used for basic arithmetic or extended into a scientific calculator, this project highlights the importance of structured programming, graphical design, and event-driven application development within Java.

In conclusion, the Java Swing Calculator project serves as a well-designed computational tool that prioritizes usability, efficiency, and scalability. Its intuitive interface, combined with dynamic expression evaluation and error-handling mechanisms, ensures that users can perform arithmetic operations with minimal effort. Future enhancements, including scientific functions, memory storage, and JavaFX integration, further highlight its adaptability and long-term usability. As a demonstration of Java’s capabilities, this project stands as a valuable example of GUI-based application development, reinforcing the significance of structured programming and graphical user interfaces in modern computational tools.

1. METHODOLOGY :-

The methodology of the Java Swing Calculator project follows a structured and systematic approach to ensure efficiency, accuracy, and usability in its implementation. Developing a graphical calculator requires careful planning, execution, and refinement to deliver a smooth and intuitive user experience. This project adheres to established software development principles, including modular coding, event-driven programming, and graphical interface design, all of which contribute to the functionality and reliability of the calculator.

The first step in the methodology involves requirement analysis, where the primary objectives and functionalities of the calculator are defined. The application is designed to handle fundamental arithmetic operations such as addition, subtraction, multiplication, and division. It must provide a graphical interface that allows users to interact dynamically through button clicks and numerical inputs. The requirement analysis also considers potential error scenarios, such as division by zero or invalid expressions, and integrates validation mechanisms to prevent computational failures. The layout and user interface must be optimized for readability and accessibility, ensuring that users can perform calculations intuitively.

Once the requirements are established, the next phase involves designing the graphical user interface using Java Swing components. The interface consists of interactive buttons representing numerical values and arithmetic operators, along with a text field for displaying user inputs and computed results. A grid-based button layout is implemented using the GridLayout manager to ensure a structured arrangement, enhancing usability. The text field serves as the primary input and output section, dynamically updating values as users enter numbers and execute calculations. Fonts and styles are carefully selected to enhance visibility, ensuring a smooth interaction between users and the application.

The implementation of event-driven programming forms a crucial aspect of the methodology. Java Swing applications rely on event listeners to capture user interactions and process them dynamically. Each button in the calculator is assigned an action listener that responds to user clicks, triggering corresponding mathematical operations. When a user presses a numeric button, the input is appended to the display field, allowing real-time updates. Similarly, when arithmetic operators are selected, the application stores the input values and prepares them for computation upon pressing the equals button. This event-driven approach ensures seamless user interaction and enhances the responsiveness of the application.

A significant component of the methodology is the evaluation mechanism, which processes mathematical expressions entered by the user. Instead of manually handling individual arithmetic operations, the calculator employs JavaScript’s ScriptEngine to dynamically evaluate expressions. This approach simplifies computational logic, allowing users to enter complete mathematical statements that are parsed and executed efficiently. The ScriptEngine interprets user-provided input as mathematical expressions and returns accurate results, eliminating the need for manual function implementations for each operation. By leveraging this dynamic evaluation method, the calculator ensures flexibility in processing numerical inputs.

Error handling and validation mechanisms are integrated into the methodology to enhance application reliability. One of the key challenges in developing a calculator is preventing invalid input sequences that may cause unexpected computational errors. The application includes validation checks to ensure that users do not enter multiple consecutive operators, which would result in ambiguous expressions. Additionally, division by zero is handled gracefully, with the system displaying a meaningful error message instead of performing an undefined operation. These safeguards improve the robustness of the calculator, preventing computational failures and ensuring a smooth user experience.

Testing and optimization form an essential phase in the methodology. Rigorous testing ensures that the calculator performs arithmetic operations accurately and responds appropriately to user interactions. Unit tests are conducted to verify the correctness of numerical computations, evaluating edge cases such as large decimal values, invalid expressions, and extreme calculations. The performance of the application is optimized to minimize resource consumption, ensuring that the graphical interface remains responsive even when handling multiple computations. The program undergoes several iterations of refinement based on testing feedback, improving stability and reliability.

The final phase of the methodology involves deployment and scalability planning. The Java Swing Calculator is designed to be compatible across multiple operating systems, including Windows, Linux, and macOS. The cross-platform nature of Java ensures that the application can run seamlessly without requiring modifications for different environments. The modular coding structure allows future enhancements to be integrated easily, ensuring long-term usability. Potential expansions include incorporating scientific operations such as trigonometric functions, logarithmic calculations, and memory storage features to improve the calculator’s capabilities.

Beyond core functionalities, the methodology considers potential graphical enhancements. While Java Swing provides a structured framework for GUI development, transitioning to JavaFX in future iterations could improve visual aesthetics. JavaFX offers advanced graphical capabilities, such as smoother animations, improved UI responsiveness, and touch-based interaction support for mobile applications. By adapting the calculator to JavaFX, users could benefit from enhanced graphical elements, making the application more visually appealing and modern.

Memory storage integration is another aspect considered in the methodology. Advanced calculators often provide memory recall functions that allow users to store and retrieve previous calculations. Implementing a memory system within the calculator would enable users to save intermediate results and perform additional computations based on stored values. This enhancement would extend the calculator’s usability, making it a more functional and versatile tool.

The methodology also ensures structured documentation, facilitating maintainability and collaboration in future developments. By adopting modular programming techniques, individual components such as button interactions, display field management, and error-handling mechanisms can be modified independently without affecting the overall functionality of the calculator. This approach simplifies debugging and allows future contributors to enhance or modify the application efficiently.

Usability testing is an integral part of the methodology, focusing on user interaction and efficiency. The calculator undergoes extensive testing to validate its responsiveness, ensuring that button clicks register correctly and arithmetic computations execute without delays. The graphical layout is examined for clarity, confirming that numerical entries and operator selections are displayed accurately. Feedback from testing phases helps refine the interface, optimizing button placement, font sizes, and overall design for better readability.

Another aspect covered in the methodology is future adaptability. The calculator is structured to accommodate additional functionalities, making it an expandable system. Scientific operations, graphical improvements, mobile compatibility, and external API integrations are potential areas for future enhancements. By maintaining a well-documented and scalable codebase, the project remains adaptable for continued development.

The overall methodology ensures that the Java Swing Calculator is built with efficiency, accuracy, and usability in mind. From requirement analysis to deployment and scalability planning, each phase contributes to a structured approach that enhances the reliability of the application. By integrating event-driven programming, dynamic expression evaluation, error handling, and graphical interface design, this project successfully delivers an interactive computational tool. The methodology supports long-term viability, allowing modifications and improvements to be introduced seamlessly, ensuring that the calculator remains a functional and valuable application in graphical user interface-based arithmetic computations.

1. CODE :-

import javax.swing.\*; // Import Java Swing for GUI components

import java.awt.\*; // Import AWT for layout management

import java.text.DecimalFormat; // Import DecimalFormat for numeric precision formatting

import javax.script.ScriptEngineManager;

import javax.script.ScriptEngine;

import javax.script.ScriptException;

// Calculator class extends JFrame for GUI

public class Calculator extends JFrame {

private JTextField display; // Text field for displaying user input and results

// Constructor to initialize the calculator

public Calculator() {

setTitle("Simple Calculator"); // Set window title

setSize(300, 400); // Set window size

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); // Close app when window is closed

setLocationRelativeTo(null); // Center window on screen

initUI(); // Call method to initialize UI components

}

// Method to initialize UI components

private void initUI() {

// Creating the display text field

display = new JTextField();

display.setEditable(false); // Prevent users from manually editing text

display.setFont(new Font("Arial", Font.BOLD, 24)); // Set font size and style

add(display, BorderLayout.NORTH); // Add text field at the top of the layout

// Creating a panel for buttons

JPanel panel = new JPanel();

panel.setLayout(new GridLayout(4, 4, 5, 5)); // Set grid layout with spacing

// Array to define calculator button labels

String[] buttons = {

"7", "8", "9", "/", "4", "5", "6", "\*",

"1", "2", "3", "-", "CLR", "0", "=", "+"

};

// Loop to create and add buttons to the panel

for (String label : buttons) {

JButton button = new JButton(label); // Create button with label

button.setFont(new Font("Arial", Font.BOLD, 20)); // Set font size

panel.add(button); // Add button to panel

// Add an action listener to handle button clicks

button.addActionListener(e -> handleButtonClick(e.getActionCommand()));

}

add(panel, BorderLayout.CENTER); // Add panel to the frame's center layout

}

// Method to handle button clicks

private void handleButtonClick(String command) {

if (command.equals("CLR")) { // Clear button resets display

display.setText("");

} else if (command.equals("=")) { // Equals button evaluates the expression

evaluate();

} else { // Otherwise, append button value to display

display.setText(display.getText() + command);

}

}

// Method to evaluate the mathematical expression

private void evaluate() {

String expression = display.getText(); // Get input from text field

// Validate input to prevent consecutive operators

if (expression.isEmpty() || expression.matches(".\*[/\*+-]{2,}.\*")) {

display.setText("Invalid Input"); // Show error for invalid expressions

return;

}

// Using JavaScript engine to evaluate expressions

ScriptEngine engine = new ScriptEngineManager().getEngineByName("JavaScript");

try {

Object result = engine.eval(expression); // Evaluate expression

// Format the output to restrict decimal places

DecimalFormat df = new DecimalFormat("#.#####");

display.setText(df.format(result));

} catch (ScriptException e) {

display.setText("Error"); // Display error message for invalid calculations

}

}

// Main method to launch the calculator application

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> new Calculator().setVisible(true)); // Create and show GUI

}

}

CODE OUTPUT :-

The output of the Java Swing Calculator is a graphical user interface (GUI) that allows users to perform basic arithmetic operations efficiently. When executed, the program launches a window displaying a **calculator interface** with a text field at the top, where user inputs and computed results appear dynamically. Below this field, numeric buttons (0-9), arithmetic operators (+, -, \*, /), and additional functional buttons such as "CLR" (clear) and "=" (evaluate) are arranged systematically in a **grid layout**, ensuring a logical and intuitive arrangement.

When a user **clicks a numeric button**, the corresponding number appears in the text field, allowing sequential entry of values. Similarly, selecting an operator adds it to the expression, enabling users to construct complete mathematical statements. Unlike traditional calculators that require individual computations for each operation, this application allows users to input **entire expressions**, which are then evaluated upon pressing the "=" button.

The evaluation process leverages **JavaScript’s ScriptEngine**, an embedded scripting mechanism that interprets user-entered expressions and processes them dynamically. This ensures **accurate computation**, providing results instantly. For example, entering "25+5\*2" and pressing "=" will yield "35" instead of "60", as **operator precedence rules** are maintained.

Error handling mechanisms significantly improve the application’s **reliability and usability**. If a user attempts an invalid operation, such as **division by zero (**10/0**)**, the system detects this and prevents the computation, displaying an "Error" message instead of executing an undefined operation. Similarly, incorrect sequences like multiple consecutive operators (++--//) are validated, and users receive "Invalid Input" notifications.

The **user experience** is smooth, with **responsive button clicks** ensuring real-time interaction. The text field updates dynamically, reflecting the entered values without noticeable delays. The graphical layout is optimized for clarity, with **large button sizes** ensuring ease of use, particularly for touchscreen devices.

In addition to basic functionality, the calculator maintains **formatted numerical output** using DecimalFormat. This ensures that results appear **precisely** without excessive decimal places. For instance, dividing 10/3 returns "3.33333" instead of a long, unreadable fraction, improving readability.

The **application operates efficiently** with minimal system resource consumption, making it suitable for multiple platforms without performance issues. Whether on Windows, Linux, or macOS, the calculator runs **seamlessly** due to Java’s **cross-platform compatibility**. The structured **modular coding approach** ensures that future enhancements, such as **scientific operations, memory storage, or JavaFX-based graphical improvements**, can be integrated smoothly.

Overall, the output of the Java Swing Calculator delivers **a reliable and intuitive arithmetic tool**, ensuring **dynamic input recognition, real-time computation, and structured error handling**. Its **smooth graphical interface**, combined with **accurate arithmetic processing**, makes it an effective solution for users requiring quick and easy mathematical calculations.





1. CODE CONCLUSION :-

The Java Swing Calculator project successfully demonstrates the integration of graphical user interface (GUI) design with functional arithmetic processing, providing an interactive and efficient tool for basic mathematical computations. Through the implementation of Java Swing components, event-driven programming, and dynamic expression evaluation, the calculator offers a structured and user-friendly approach to performing numerical operations. The project highlights the importance of designing software that prioritizes usability while maintaining accuracy and efficiency.

The completion of this calculator reinforces the effectiveness of GUI-based applications in enhancing user interaction. Unlike traditional console-based calculators that require manual text input, this application simplifies arithmetic operations by allowing users to click buttons and input expressions naturally. The inclusion of intuitive interface elements such as a text field for displaying results, buttons for numeric entry, and operators for computation significantly improves usability. The structured layout ensures logical flow, making it easy for users to navigate the application without confusion.

A key achievement of this project is the successful implementation of event-driven programming, ensuring real-time responsiveness. Java Swing's ActionListener mechanism enables the calculator to process user inputs dynamically, capturing button clicks and executing corresponding mathematical functions. This approach provides an interactive experience, allowing users to see immediate updates in the display field based on their selections. The graphical interface reacts instantaneously to user interactions, ensuring fluid computational processing without delays.

The integration of JavaScript’s ScriptEngine to handle dynamic mathematical expression evaluation stands out as one of the major strengths of the project. Traditional calculators often require manual handling of individual arithmetic operations, with each function needing separate implementation. By leveraging the ScriptEngine, the application interprets complete expressions, allowing users to input mathematical statements such as "5+3\*2" and receive the correct result following standard order of operations. This automated evaluation approach simplifies computational logic while maintaining precision.

Another critical aspect of the project's success lies in its robust error-handling mechanisms. In software dealing with numerical calculations, preventing invalid operations is essential to maintain reliability. The calculator ensures that improper expressions, such as division by zero or consecutive arithmetic operators, do not disrupt computations. Instead of executing undefined operations, the system displays meaningful error messages, guiding users toward correcting their input. This validation process enhances application stability and user trust by minimizing unintended errors.

The structured development methodology adopted in this project ensures modular coding practices, making future scalability possible. The calculator follows a well-defined framework where individual components, such as button interactions, display updates, and arithmetic processing, are implemented as separate entities. This modular approach enhances maintainability, allowing future developers to expand functionality without affecting existing core operations. The project serves as a foundation for further enhancements, including additional mathematical functions, improved graphical aesthetics, and mobile adaptability.

Future expansions of this calculator could include incorporating scientific functions such as logarithmic calculations, trigonometric operations, and exponential functions. These features would transform the application from a basic arithmetic calculator to a comprehensive computational tool, making it useful for students, engineers, and professionals requiring more advanced calculations. Additionally, the integration of a memory storage system could enhance usability by allowing users to save previous calculations and retrieve stored values for further operations. This would extend the functionality beyond simple computations, enabling a history-tracking system for more complex mathematical problem-solving.

Transitioning from Java Swing to JavaFX presents another possible evolution for the project, offering advanced graphical capabilities, smoother animations, and improved user interface responsiveness. JavaFX’s modern rendering system allows for a more refined and visually appealing calculator design, making interactions more fluid. By adopting JavaFX, the application could become adaptable for touchscreen interactions, making it suitable for mobile and tablet devices. This shift would broaden the accessibility of the calculator, providing users with a more versatile and visually engaging tool.

Performance optimization is another area where future improvements could be made. While the current implementation is efficient, additional memory management techniques could be explored to minimize resource consumption further. Ensuring that the calculator remains lightweight and responsive across various platforms would contribute to its usability in different system environments. Optimization strategies focusing on reducing processing overhead, refining graphical rendering, and enhancing computational efficiency could further elevate the application's performance.

The process of developing this calculator has reinforced key programming principles, including structured coding, event-driven logic, and modular component design. Throughout the project, emphasis was placed on maintaining readability, ensuring that the codebase remains accessible for future modifications. The systematic approach to GUI design facilitated the creation of a clean and organized interface, optimizing navigation while preserving functionality. Each decision made during development aimed to improve the overall user experience, delivering a reliable and efficient arithmetic tool.

In addition to its practical functionality, the Java Swing Calculator serves as an educational resource for learning GUI-based application development in Java. The project showcases how Java Swing components can be used to construct interactive interfaces while demonstrating event-driven programming concepts. Students and developers seeking to understand GUI design principles can analyze the project's structure to gain insights into building responsive applications. The integration of JavaScript’s ScriptEngine also provides an example of how external processing engines can be leveraged within Java applications to enhance computational efficiency.

Testing and optimization played a crucial role in ensuring the calculator functions as expected. Multiple test cases were conducted to validate the correctness of arithmetic computations, verifying that the calculator processes expressions accurately. Edge cases such as large decimal values, invalid inputs, and extreme calculations were assessed to confirm the effectiveness of error-handling mechanisms. Performance evaluations ensured that the calculator remained responsive, even when handling multiple user interactions simultaneously. These refinements contributed to the robustness of the application, making it a dependable tool for mathematical operations.

The significance of GUI-based applications in modern computing cannot be understated, as they provide intuitive ways for users to interact with software. The Java Swing Calculator project exemplifies the power of graphical interfaces in enhancing usability and accessibility. By combining structured programming techniques with interactive design elements, this project highlights the benefits of developing applications that prioritize user experience while maintaining technical efficiency. As GUI technology continues to evolve, projects like this one serve as stepping stones toward more sophisticated software solutions.

In conclusion, the Java Swing Calculator project achieves its goal of creating a fully functional arithmetic tool with a graphical interface, event-driven programming, and dynamic expression evaluation. The calculator successfully integrates Java Swing components to deliver an intuitive and efficient user experience. Through its modular coding structure and systematic development approach, the project remains adaptable for future enhancements. The inclusion of error-handling mechanisms ensures reliability, preventing invalid operations from disrupting calculations. The potential for scalability allows the application to be expanded into a more advanced computational system, incorporating scientific functions, graphical improvements, and mobile compatibility.

The completion of this project demonstrates the effectiveness of GUI-based applications in simplifying mathematical computations while providing an engaging and responsive interface. The Java Swing Calculator is not only a functional arithmetic tool but also a valuable learning resource for developers exploring graphical programming, event-driven logic, and structured software design. The insights gained from this project reinforce the importance of usability, accuracy, and efficiency in software development, showcasing how interactive applications can be designed to improve user interaction with computational tools. Whether used for basic arithmetic or extended into a scientific calculator, this project highlights the practical application of Java-based graphical computation while paving the way for future advancements in digital calculators.

The final phases involved testing, optimization, and scalability planning. Multiple test scenarios were conducted to validate accuracy, responsiveness, and usability. Performance refinements minimized resource consumption, ensuring smooth execution across different operating systems. With a modular coding structure, the project remains open to future enhancements, including scientific functions, memory storage, and JavaFX-based graphical improvements.

Reflecting on the entire process, this project was an engaging and valuable learning experience, demonstrating the power of GUI-based applications in Java. It would not have been possible without dedication to structured programming, attention to user interaction, and meticulous refinement. A sincere thank you to everyone involved in conceptualizing and refining this project. Your collaboration, insights, and feedback have contributed to making the Java Swing Calculator a functional and user-friendly computational tool. This journey of development has been truly rewarding, and I look forward to future enhancements and innovations in graphical software design.

SUBMITTED BY

NAME - MAHASWETA TALIK

EMAIL - [23053218@kiit.ac.in](mailto:23053218@kiit.ac.in)

MOBILE NO. - 8777467809

DOMAIN - JAVA DEVELOPMENT