Report On

Calculator

Submitted in partial fulfillment of the requirements of the Course project in Semester III of Second Year Artificial Intelligence and Data Science

by

Dhruv Gharat (Roll No.11)

Preeti Prajapati(Roll No.48)

Yash Patil (Roll No. 45)

Supervisor

Miss Sneha Yadav

University of Mumbai

Vidyavardhini's College of Engineering & Technology

Department of Artificial Intelligence and Data Science

(2023-24)

Vidyavardhini's College of Engineering & Technology
Department of Artificial Intelligence and Data Science

CERTIFICATE

This is to certify that the project entitled “Calculator” is a bonafide work of "Dhruv Gharat (Roll No.11), Yash Patil (Roll No. 45), Preeti Prajapati (Roll No.48.),submitted to the University of Mumbai in partial fulfillment of the requirement for the Course project in semester III of Second Year Artificial Intelligence and Data Science engineering.

Supervisor

Miss Sneha Yadav

Dr. Tatwadarshi P. N.
Head of Department

Table of Contents

Pg. No

ABSTRACT

The provided code is a simple Java program that implements a basic calculator using the AWT (Abstract Window Toolkit) library.

The "MyCalc" Java program is a graphical user interface (GUI) calculator created using the AWT library. It offers a simple and intuitive way to perform basic arithmetic calculations. The calculator features a graphical interface with buttons for digits 0-9, decimal point, arithmetic operations (addition, subtraction, multiplication, division, and modulus), and special functions like clearing the input, toggling the sign of the number, and backspacing. It also displays the calculation results on a labeled area within the application window.

Users can input numbers and perform operations by clicking the corresponding buttons, and the program maintains and updates the input and operation state accordingly. Once a calculation is complete, users can obtain the result by clicking the "=" button.

The code utilizes event handling to respond to button clicks and perform the necessary arithmetic operations. It efficiently handles user inputs, checks for errors, and provides feedback to the user in case of invalid or incomplete inputs.

The "MyCalc" calculator application is a straightforward Java program that demonstrates the use of AWT for creating a simple, interactive calculator with essential arithmetic capabilities.

Problem Statement

The provided Java code for the "MyCalc" calculator application presents several challenges and shortcomings that need attention for enhanced functionality and user experience. Firstly, the code lacks robust error handling, leaving potential issues like division by zero unaddressed. Additionally, it fails to provide adequate user feedback, leaving users unsure of the current operation or result.

Furthermore, the calculator is limited in terms of supported arithmetic operations, covering only basic operations like addition, subtraction, multiplication, division, and modulus, with no provision for more advanced functions. The code also employs inefficient string

manipulation for numeric input, potentially leading to conversion errors and localization issues.

The user interface's inconsistent layout, with non-standardized button sizes and positions, affects the calculator's visual appeal and usability. Moreover, the code lacks comprehensive testing and validation mechanisms to ensure reliable performance across various scenarios. It also lacks memory functions commonly found in calculators, limiting its utility for more complex calculations.

Module Description

The "MyCalc" calculator application can be dissected into several distinct modules, each serving a specific purpose within the program.

The User Interface Module governs the creation and arrangement of graphical user interface elements, including buttons for digits, arithmetic operations, and special functions, along with a result label. These components are visually organized within the calculator window.

The Event Handling Module takes charge of user interactions with the calculator. It encompasses event listeners for each button, ensuring that appropriate actions are triggered when a button is pressed. These actions range from updating user input to performing arithmetic operations and displaying results.

In the absence of comprehensive error handling, there is an opportunity for an **Error Handling Module** to be introduced. This module would be responsible for detecting and managing potential errors, such as division by zero or non-numeric inputs, while providing clear and informative feedback to the user.

The Arithmetic Operations Module is central to the calculator's functionality. It contains the logic for addition, subtraction, multiplication, division, and modulus operations, processing user input to generate results and display them through the result label.

A Result Display Module ensures that the calculated results are accurately presented to the user. It involves updating the result label with the outcome of arithmetic operations, while also providing clear feedback in cases of incomplete or invalid operations.

An Optional Memory and Advanced Operations Module could extend the calculator's capabilities, introducing features like memory storage and recall functions, as well as advanced mathematical operations such as square root and exponentiation.

To address potential issues with numeric input and localization, a module dedicated to Localization and Input Handlingcould be implemented. This module would manage numeric input more efficiently, validate user input, and accommodate variations in decimal separators.

Additionally, a Testing and Validation Module could be integrated to develop a comprehensive testing framework, ensuring the calculator functions correctly under diverse scenarios and inputs.

For improved code comprehensibility and maintainability, a module focused on Documentation and Code Comment could be added. This module would include detailed code comments and documentation, explaining the program's functionality and assisting developers in understanding and modifying the code. By organizing the "MyCalc" calculator application into these modules, it becomes easier to enhance its structure, maintainability, and functionality, while also offering a clearer roadmap for addressing identified issues and expanding the application's capabilities.

Brief Description of Software and Hardware Used and Its Programming

Software Used:

- Java Programming Language:The code for the "MyCalc" calculator is written in Java. Java is a platform-independent, object-oriented programming language known for its portability and suitability for developing GUI applications.

- Java AWT (Abstract Window Toolkit): The code relies on the Java AWT library for creating the graphical user interface. AWT provides classes and components for building GUI applications.

Hardware Requirements:

-No Hardware required in this project.

Programming Description:

- User Interface Creation: The code initializes the user interface components such as buttons for digits, arithmetic operations, and labels within a window frame created using Java AWT classes. It sets the size and layout of these components to create the calculator's graphical interface.

- Event Handling: The code utilizes event handling techniques to respond to user interactions with the calculator buttons. Event listeners are implemented to detect button clicks and trigger corresponding actions. User inputs, arithmetic operations, and results are managed through event handling.

- Arithmetic Operations: The code includes logic for fundamental arithmetic operations, including addition, subtraction, multiplication, division, and modulus. It captures user input, performs the specified operation based on the user's choice, and displays the results.

- Error Handling: While the code lacks comprehensive error handling, it attempts to handle certain error scenarios, such as invalid numeric inputs, by providing user feedback in the form of error messages displayed in the result label.

- Memory and Advanced Operations (Optional): The code doesn't include memory or advanced mathematical operations, but there's potential to extend the calculator's functionality by implementing these features in a dedicated module.

- Localization and Input Handling (Potential Improvement): The code could benefit from improvements in input handling and localization to ensure compatibility with different regional formats for numeric input.

- Testing and Validation (Potential Improvement): For enhanced reliability, the code could benefit from a more comprehensive testing and validation module to verify correct behavior under various scenarios.

- Documentation and Code Comments (Potential Improvement): To improve code maintainability and understanding, comprehensive code comments and documentation should be added, explaining the code's functionality and usage.

CODES

```
import java.awt.*;   import
java.awt.event.*;
class MyCalc extends WindowAdapter implements ActionListener{
  Frame f;
Label l1;
Button b1,b2,b3,b4,b5,b6,b7,b8,b9,b0;
Button badd,bsub,bmult,bdiv,bmod,bcalc,bclr,bpts,bneg,bback;
double xd; double num1,num2,check;

MyCalc(){            f=  new  Frame("MY
CALCULATOR"); // INSTANTIATING
COMPONENETS
l1=new Label();
l1.setBackground(Color.LIGHT_GRAY);
l1.setBounds(50,50,260,60);


b1=new Button("1");
b1.setBounds(50,340,50,50); b2=new
Button("2");
b2.setBounds(120,340,50,50); b3=new
Button("3");
b3.setBounds(190,340,50,50); b4=new
Button("4");
b4.setBounds(50,270,50,50); b5=new
Button("5");
b5.setBounds(120,270,50,50);
b6=new Button("6");
b6.setBounds(190,270,50,50); b7=new
Button("7");
b7.setBounds(50,200,50,50); b8=new
Button("8");
```

```
b8.setBounds(120,200,50,50); b9=new
Button("9");
b9.setBounds(190,200,50,50); b0=new
Button("0");
b0.setBounds(120,410,50,50);
bneg=new Button("+/-");
bneg.setBounds(50,410,50,50);
bpts=new Button(".");
bpts.setBounds(190,410,50,50);
bback=new Button("back");
bback.setBounds(120,130,50,50);

badd=new Button("+");
badd.setBounds(260,340,50,50); bsub=new
Button("-");
bsub.setBounds(260,270,50,50);
bmult=new Button("*");
bmult.setBounds(260,200,50,50); bdiv=new
Button("/");
bdiv.setBounds(260,130,50,50); bmod=new
Button("%");
bmod.setBounds(190,130,50,50);
bcalc=new Button("=");
bcalc.setBounds(245,410,65,50); bclr=new
Button("CE");
bclr.setBounds(50,130,65,50);


b1.addActionListener(this);
b2.addActionListener(this);
b3.addActionListener(this);
b4.addActionListener(this);
b5.addActionListener(this);
b6.addActionListener(this);
```

```
b7.addActionListener(this);
b8.addActionListener(this);
b9.addActionListener(this);
b0.addActionListener(this);

bpts.addActionListener(this);
bneg.addActionListener(this);
bback.addActionListener(this);

badd.addActionListener(this);
bsub.addActionListener(this);
bmult.addActionListener(this);
bdiv.addActionListener(this);
bmod.addActionListener(this);
bcalc.addActionListener(this);
bclr.addActionListener(this);

f.addWindowListener(this); //ADDING
TO FRAME
f.add(l1);
f.add(b1);      f.add(b2);      f.add(b3);      f.add(b4);      f.add(b5);f.add(b6);
f.add(b7); f.add(b8);f.add(b9);f.add(b0);

f.add(badd);    f.add(bsub);    f.add(bmod);   f.add(bmult);   f.add(bdiv);
f.add(bmod);f.add(bcalc);

f.add(bclr); f.add(bpts);f.add(bneg); f.add(bback);

f.setSize(360,500);
f.setLayout(null);
f.setVisible(true);
}
                //FOR CLOSING THE WINDOW
public void windowClosing(WindowEvent e) {
```

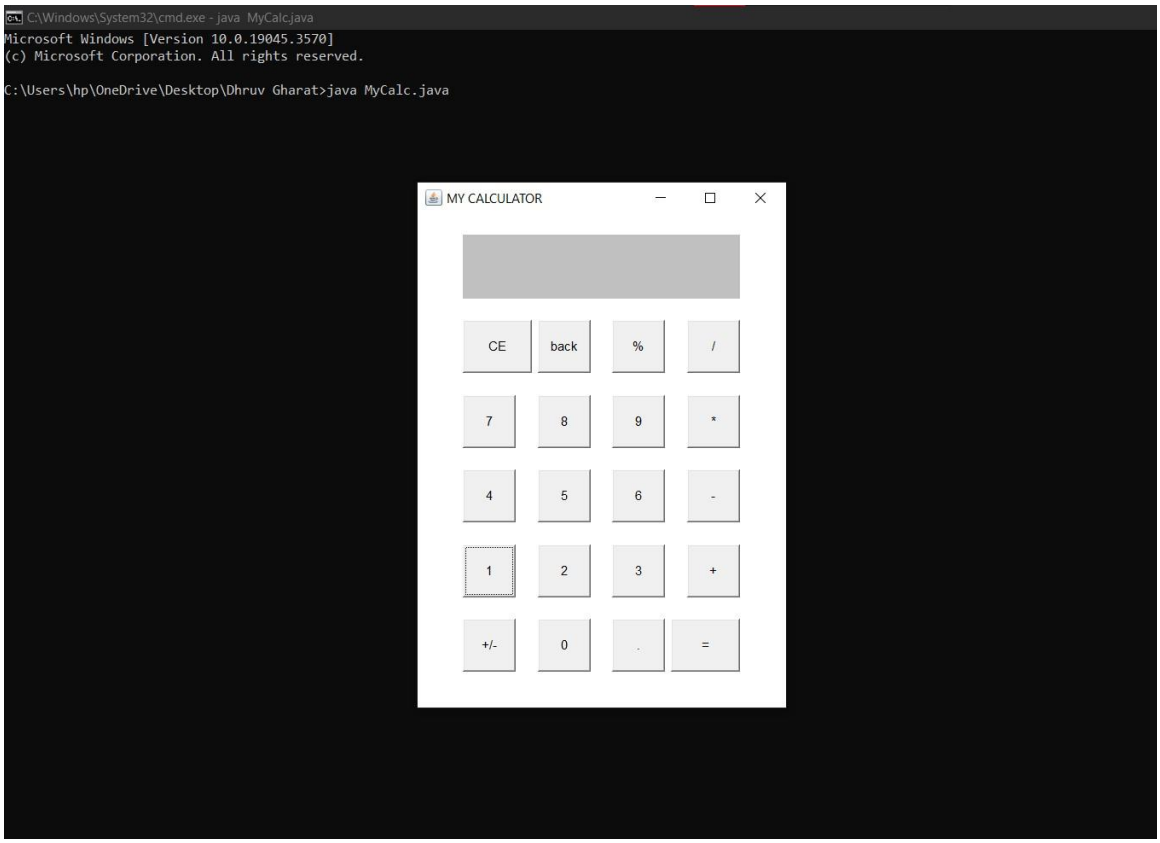
```
 f.dispose();
}

public void actionPerformed(ActionEvent e){
 String z,zt;
                    //NUMBER BUTTON
if(e.getSource()==b1){
zt=l1.getText();   z=zt+"1";
l1.setText(z); }
if(e.getSource()==b2){
zt=l1.getText(); z=zt+"2";
l1.setText(z); }
if(e.getSource()==b3){
zt=l1.getText();   z=zt+"3";
l1.setText(z); }
if(e.getSource()==b4){
zt=l1.getText();   z=zt+"4";
l1.setText(z); }
if(e.getSource()==b5){
zt=l1.getText();   z=zt+"5";
l1.setText(z); }
if(e.getSource()==b6){
zt=l1.getText();
 z=zt+"6";
l1.setText(z); }
if(e.getSource()==b7){
zt=l1.getText();
z=zt+"7";
l1.setText(z); }
if(e.getSource()==b8){
zt=l1.getText();
z=zt+"8";
l1.setText(z); }
if(e.getSource()==b9){
```

```
zt=l1.getText();
z=zt+"9";
l1.setText(z); }
if(e.getSource()==b0){
zt=l1.getText();
z=zt+"0";
l1.setText(z);
}

if(e.getSource()==bpts){  //ADD DECIMAL PTS
zt=l1.getText();   z=zt+".";   l1.setText(z);
}
if(e.getSource()==bneg){ //FOR NEGATIVE
zt=l1.getText();   z="-"+zt;   l1.setText(z);
}

if(e.getSource()==bback){  // FOR  BACKSPACE
zt=l1.getText();        try{            z=zt.substring(0,
zt.length()-1);
   }catch(StringIndexOutOfBoundsException                    f){return;}
l1.setText(z);
}
            //AIRTHMETIC                          BUTTON
if(e.getSource()==badd){                  //FOR ADDITION
try{                num1=Double.parseDouble(l1.getText());
}catch(NumberFormatException                              f){
l1.setText("Invalid Format");       return;
   }   z="";   l1.setText(z);   check=1; }
if(e.getSource()==bsub){                  //FOR SUBTRACTION
try{
   num1=Double.parseDouble(l1.getText());
}catch(NumberFormatException f){
l1.setText("Invalid Format");       return;
```

```
   }   z="";   l1.setText(z);   check=2; } if(e.getSource()==bmult){
//FOR MULTIPLICATION   try{
num1=Double.parseDouble(l1.getText());
}catch(NumberFormatException f){       l1.setText("Invalid
Format");       return;
   }          z="";          l1.setText(z);          check=3;    }
if(e.getSource()==bdiv){                  //FOR DIVISION
try{              num1=Double.parseDouble(l1.getText());
}catch(NumberFormatException                          f){
l1.setText("Invalid Format");       return;
   }   z="";   l1.setText(z);   check=4; } if(e.getSource()==bmod){
//FOR            MOD/REMAINDER                                try{
num1=Double.parseDouble(l1.getText());
}catch(NumberFormatException  f){              l1.setText("Invalid
Format");       return;
   }   z="";
l1.setText(z);
check=5;
}
                     //RESULT BUTTON
if(e.getSource()==bcalc){            try{
num2=Double.parseDouble(l1.getText());
}catch(Exception f){
l1.setText("ENTER NUMBER FIRST ");
return;
   }
 if(check==1)     xd
=num1+num2;   if(check==2)
xd =num1-num2;   if(check==3)
xd =num1*num2;
if(check==4)     xd
=num1/num2;    if(check==5)
xd =num1%num2;
l1.setText(String.valueOf(xd));
```

```
}
                    //FOR CLEARING THE LABEL and Memory
if(e.getSource()==bclr){   num1=0;   num2=0;   check=0;
xd=0;    z="";    l1.setText(z);
  }

}
//MAIN METHOD where objects of MyCalc is instantaiated
public static void main(String args[]){          new MyCalc();
  }
}
```

RESULT & CONCLUSION



CONCLUSION

The "MyCalc" calculator code is a basic Java application that creates a graphical user interface for performing simple arithmetic operations. While it successfully implements core calculator functionality, it has several limitations. The code lacks robust error handling, clear user feedback, support for advanced functions, and efficient numeric input handling. Additionally, it does not consider internationalization aspects related to numeric input. Furthermore, comprehensive testing and documentation are absent. Addressing these issues would significantly improve the calculator's reliability, user experience, and maintainability.