Green University of Bangladesh

Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering, Semester: Spring, Year: 2024, B.Sc. in CSE (weekend)

Lab Report # 06

Course Title: Object Oriented Programming Lab

Course Code: CSE-202

Section: 223_E1

Student Details

	Name	ID
1.	Waliullah	223015026

Date	**: 24-05-2024**
Submission Date	**: 31-05-2024**
Course Teacher's Name	**: Abdullah Al Farhad**

Assignment Status

Marks: ………………………………… **Signature:....................**

Comments:... **Date:.............................**

Title: Creating GUI Calculation in Java.

1. Introduction:

- The Calculator project is a simple GUI-based application developed using Java's AWT library to perform basic arithmetic operations. It aims to provide a functional and user-friendly interface for performing addition, subtraction, multiplication, and division.

2. Objective:

- **Create a Functional GUI Calculator:** Develop a simple, graphical user interface (GUI) based
  calculator using Java's AWT library to perform basic arithmetic operations such as addition,
  subtraction, multiplication, and division.
- **Implement Basic Error Handling:** Ensure the calculator can handle basic errors such as division by zero and invalid inputs gracefully, providing appropriate feedback to the user.
- **Enhance User Experience:** Design the calculator to be user-friendly and intuitive, including clear button labels and a straightforward layout for easy interaction.

3. Exploring Access Modifiers:

- In Java, access modifiers are keywords used to control the visibility of classes, variables, methods, and constructors within a program. The four main access modifiers are public, private, protected, and default (package-private). Public access allows members to be accessed from anywhere, while private access restricts access to within the same class. Protected access allows access within the same package or subclasses, and default access allows access only within the same package. Understanding and appropriately using access modifiers is crucial for encapsulation, data hiding, and maintaining code integrity in Java programs.

4. Procedure:

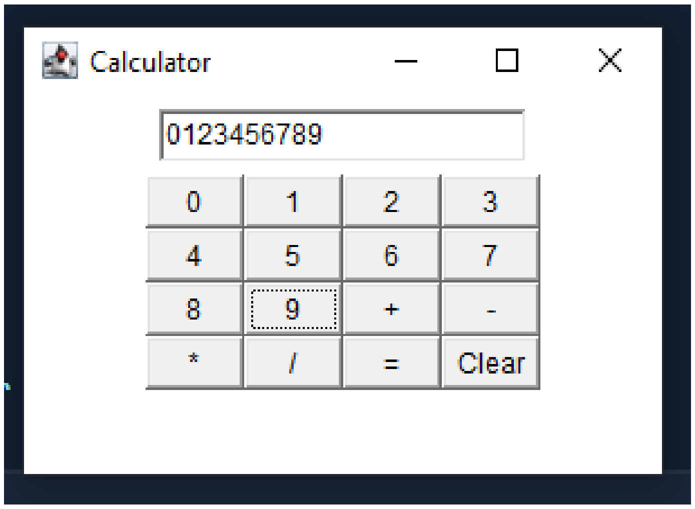
- **Launch the Calculator:** Execute the Java program to launch the Calculator application, which opens a graphical user interface (GUI) window displaying the calculator interface.
- **Input Arithmetic Operations:** Use the buttons provided in the GUI to input arithmetic operations such as addition (+), subtraction (-), multiplication (*), and division (/), along with numeric operands.
- **View Results:** After inputting the desired arithmetic expression, press the "=" button to calculate and display the result in the text field.

5. Implementations and Output:

- Code:

```
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

public class Main implements ActionListener {
    String left_value;
    String right_value;
    int flag = 0;
    Frame f;
    Panel p;
    Button b0, b1,b2, b3, b4, b5, b6, b7, b8, b9,badd, bsub, bmul, bdiv, beq, bclr;
    TextField t;
    GridLayout g;
    Main(){
        f = new Frame("Calculator");
        p = new Panel();
        f.setLayout(new FlowLayout());
        b0 = new Button("0");
        b0.addActionListener(this);

        b1 = new Button("1");
        b1.addActionListener(this);

        b2 = new Button("2");
        b2.addActionListener(this);

        b3 = new Button("3");
        b3.addActionListener(this);

        b4 = new Button("4");
        b4.addActionListener(this);

        b5 = new Button("5");
        b5.addActionListener(this);

        b6 = new Button("6");
        b6.addActionListener(this);

        b7 = new Button("7");
        b7.addActionListener(this);

        b8 = new Button("8");
        b8.addActionListener(this);

        b9 = new Button("9");
        b9.addActionListener(this);

        badd = new Button("+");
        badd.addActionListener(this);

        bsub = new Button("-");
        bsub.addActionListener(this);

        bmul = new Button("*");
        bmul.addActionListener(this);

        bdiv = new Button("/");
        bdiv.addActionListener(this);

        beq = new Button("=");
        beq.addActionListener(this);

        bclr = new Button("Clear");
        bclr.addActionListener(this);

        t = new TextField(20);
        f.add(t);

        g = new GridLayout(4,4);
        p.setLayout(g);

        p.add(b0);
        p.add(b1);
        p.add(b2);
        p.add(b3);
        p.add(b4);
        p.add(b5);
        p.add(b6);
        p.add(b7);
        p.add(b8);
        p.add(b9);
        p.add(badd);
        p.add(bsub);
        p.add(bmul);
        p.add(bdiv);
        p.add(beq);
        p.add(bclr);

        f.add(p);
        f.setSize(300, 200);
        f.setVisible(true);

        f.addWindowListener(new WindowAdapter() {
            @Override
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });
    }
    public static void main(String[] args) {
        Main a = new Main();
    }

    @Override
    public void actionPerformed(ActionEvent e) {

        if(e.getSource() == b0){
            String s = t.getText();
            t.setText(s+"0");
        }
        if(e.getSource() == b1){
            String s = t.getText();
            t.setText(s+"1");
        }
        if(e.getSource() == b2){
            String s = t.getText();
            t.setText(s+"2");
        }
        if(e.getSource() == b3){
            String s = t.getText();
            t.setText(s+"3");
        }
        if(e.getSource() == b4){
            String s = t.getText();
            t.setText(s+"4");
        }
        if(e.getSource() == b5){
            String s = t.getText();
            t.setText(s+"5");
        }
        if(e.getSource() == b6){
            String s = t.getText();
            t.setText(s+"6");
        }
        if(e.getSource() == b7){
            String s = t.getText();
            t.setText(s+"7");
        }
        if(e.getSource() == b8){
            String s = t.getText();
            t.setText(s+"8");
            }
            if(e.getSource() == b9){
                String s = t.getText();
                t.setText(s+"9");
            }
            if(e.getSource() == badd){
                left_value = t.getText();
                t.setText("");
                flag = 1;
            }
            if(e.getSource() == bsub){
                left_value = t.getText();
                t.setText("");
                flag = 2;
            }
            if(e.getSource() == bmul){
                left_value = t.getText();
                t.setText("");
                flag = 3;
            }
            if(e.getSource() == bdiv){
                left_value = t.getText();
                t.setText("");
                flag = 4;
            }
            if(e.getSource() == bclr){
                left_value = t.getText();
                t.setText("");
                flag = 5;
            }
            if(e.getSource() == beq){
                right_value = t.getText();
                if(flag == 1){
                    int a = Integer.parseInt(left_value);
                    int b = Integer.parseInt(right_value);
                    t.setText(a+b+"");
                }
                if(flag == 2){
                    int a = Integer.parseInt(left_value);
                    int b = Integer.parseInt(right_value);
                    t.setText(a-b+"");
                }
                if(flag == 3){
                    int a = Integer.parseInt(left_value);
                    int b = Integer.parseInt(right_value);
                    t.setText(a*b+"");
                }
                if(flag == 4){
                    double a = Integer.parseInt(left_value);
                    double b = Integer.parseInt(right_value);
                    t.setText(a/b+"");
                }
                if(flag == 5){
                    t.setText("");
                }
            }
        }
    }

```

- **Output:**



6. Limitations:

Multiple inheritance, which allows a class to inherit from more than one parent class, has been intentionally avoided in Java.

- The calculator only supports integer operations and does not handle decimal numbers.
- There is no handling for division by zero, which will cause a runtime error (ArithmeticException).
- The calculator can only perform one operation at a time and does not support complex expressions with multiple operations.
- The calculator does not support direct input of negative numbers, requiring users to subtract from zero instead.
- The user interface is basic and lacks advanced features like memory functions or a clear entry button(CE).

7. Conclusion:

The current implementation of the Calculator class is a functional, albeit basic, tool for performing simple arithmetic operations. It successfully demonstrates the use of the AWT library to create a graphical user interface in Java. However, it has several limitations that restrict its usability and robustness. The lack of support for decimal numbers and the inability to handle division by zero are significant drawbacks. Additionally, the calculator's inability to process multiple operations in sequence limits its functionality for more complex calculations. The absence of a method to input negative numbers directly also detracts from the user experience. The user interface, while clear and straightforward, could benefit from additional features such as memory storage and a clear entry button. Despite these limitations, the project provides a solid foundation for further development and enhancements.