NAME – RAJDEEP JAISWAL DATE – 19 NOV 2021

BRANCH – BTECH CSE SEC = 608 - A

UID -20BCS2761 Subject – JAVA

AIM –

Design a simple calculator (called SwingCalculator).
Hints:
· Set the ContentPane to BorderLayout. Add a JTextField (tfDisplay) to the NORHT. Add a JPanel (panelButtons) to the CENTER. Set the JPanel to GridLayout of 4x4, and add the 16 buttons.
· Operator buttons "+", "-", "*", "/", "%" and "=".

Code in Text form –

```
package  com.company;

import java.awt.BorderLayout;
import java.awt.Color;

import java.awt.Container;
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.GridLayout;
import java.awt.Window;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JPanel;
import javax.swing.JTextArea;
import javax.swing.KeyStroke;

public class Calculator extends JFrame implements ActionListener {
```

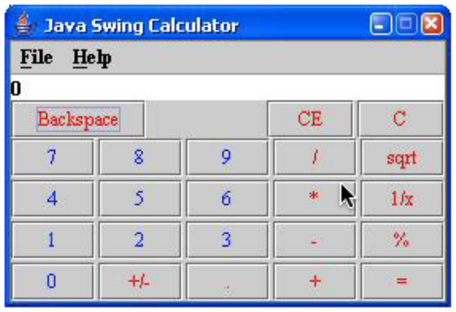
```
// Variables
final int MAX_INPUT_LENGTH = 20;
final int INPUT_MODE = 0;
final int RESULT_MODE = 1;
final int ERROR_MODE = 2;
int displayMode;
boolean clearOnNextDigit, percent;
double lastNumber;
String lastOperator;
private JMenu jmenuFile, jmenuHelp;
private JMenuItem jmenuitemExit, jmenuitemAbout;
private JLabel jlbOutput;
private JButton jbnButtons[];
private JPanel jplMaster, jplBackSpace, jplControl;
/*

 * Font(String name, int style, int size)

 Creates a new Font from the specified name, style and point size.

 */
Font f12 = new Font("Times New Roman", 0, 12);
Font f121 = new Font("Times New Roman", 1, 12);
// Constructor
public Calculator() {
    /* Set Up the JMenuBar.

     * Have Provided All JMenu's with Mnemonics

     * Have Provided some JMenuItem components with Keyboard Accelerators

     */
    jmenuFile = new JMenu("File");
    jmenuFile.setFont(f121);
    jmenuFile.setMnemonic(KeyEvent.VK_F);
    jmenuitemExit = new JMenuItem("Exit");
    jmenuitemExit.setFont(f12);
    jmenuitemExit.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_X,
            ActionEvent.CTRL_MASK));
    jmenuFile.add(jmenuitemExit);
    jmenuHelp = new JMenu("Help");
    jmenuHelp.setFont(f121);
    jmenuHelp.setMnemonic(KeyEvent.VK_H);
    jmenuitemAbout = new JMenuItem("About Calculator");
    jmenuitemAbout.setFont(f12);
    jmenuHelp.add(jmenuitemAbout);
    JMenuBar mb = new JMenuBar();
    mb.add(jmenuFile);
    mb.add(jmenuHelp);
```

```
        setJMenuBar(mb);
        //Set frame layout manager
        setBackground(Color.gray);
        jplMaster = new JPanel();
        jlbOutput = new JLabel("0");
        jlbOutput.setHorizontalTextPosition(JLabel.RIGHT);
        jlbOutput.setBackground(Color.WHITE);
        jlbOutput.setOpaque(true);
        // Add components to frame
        getContentPane().add(jlbOutput, BorderLayout.NORTH);
        jbnButtons = new JButton[23];
        //    GridLayout(int rows, int cols, int hgap, int vgap)
        JPanel jplButtons = new JPanel(); // container for Jbuttons
        // Create numeric Jbuttons
        for (int i = 0; i &lt;= 9; i++) {
            // set each Jbutton label to the value of index
            jbnButtons[i] = new JButton(String.valueOf(i));
        }
        // Create operator Jbuttons
        jbnButtons[10] = new JButton("+/-");
        jbnButtons[11] = new JButton(".");
        jbnButtons[12] = new JButton("=");
        jbnButtons[13] = new JButton("/");
        jbnButtons[14] = new JButton("*");
        jbnButtons[15] = new JButton("-");
        jbnButtons[16] = new JButton("+");
        jbnButtons[17] = new JButton("sqrt");
        jbnButtons[18] = new JButton("1/x");
        jbnButtons[19] = new JButton("%");
        jplBackSpace = new JPanel();
        jplBackSpace.setLayout(new GridLayout(1, 1, 2, 2));
        jbnButtons[20] = new JButton("Backspace");
        jplBackSpace.add(jbnButtons[20]);
        jplControl = new JPanel();
        jplControl.setLayout(new GridLayout(1, 2, 2, 2));
        jbnButtons[21] = new JButton(" CE ");
        jbnButtons[22] = new JButton("C");
        jplControl.add(jbnButtons[21]);
        jplControl.add(jbnButtons[22]);
        //Setting all Numbered JButton's to Blue. The rest to Red
        for (int i = 0; i &lt; jbnButtons.length; i++) {
            jbnButtons[i].setFont(f12);
            if (i &lt; 10)
            jbnButtons[i].setForeground(Color.blue);
         else
            jbnButtons[i].setForeground(Color.red);
        }
        // Set panel layout manager for a 4 by 5 grid
        jplButtons.setLayout(new GridLayout(4, 5, 2, 2));
        //Add buttons to keypad panel starting at top left
        // First row
        for (int i = 7; i &lt;= 9; i++) {
            jplButtons.add(jbnButtons[i]);
        }
        // add button / and sqrt
        jplButtons.add(jbnButtons[13]);
        jplButtons.add(jbnButtons[17]);
        // Second row
        for (int i = 4; i &lt;= 6; i++) {
            jplButtons.add(jbnButtons[i]);
        }
        // add button * and x^2
        jplButtons.add(jbnButtons[14]);
        jplButtons.add(jbnButtons[18]);
        // Third row
        for (int i = 1; i &lt;= 3; i++) {
            jplButtons.add(jbnButtons[i]);
        }
        //adds button - and %
        jplButtons.add(jbnButtons[15]);
        jplButtons.add(jbnButtons[19]);
        //Fourth Row
        // add 0, +/-, ., +, and =
        jplButtons.add(jbnButtons[0]);
        jplButtons.add(jbnButtons[10]);
        jplButtons.add(jbnButtons[11]);
        jplButtons.add(jbnButtons[16]);
        jplButtons.add(jbnButtons[12]);
        jplMaster.setLayout(new BorderLayout());
        jplMaster.add(jplBackSpace, BorderLayout.WEST);
        jplMaster.add(jplControl, BorderLayout.EAST);
        jplMaster.add(jplButtons, BorderLayout.SOUTH);
        // Add components to frame
        getContentPane().add(jplMaster, BorderLayout.SOUTH);
        requestFocus();
        //activate ActionListener
        for (int i = 0; i &lt; jbnButtons.length; i++) {
            jbnButtons[i].addActionListener(this);
        }
        jmenuitemAbout.addActionListener(this);
        jmenuitemExit.addActionListener(this);
        clearAll();
        //add WindowListener for closing frame and ending program
        addWindowListener(new WindowAdapter() {

            public void windowClosed(WindowEvent e) {
                System.exit(0);
            }
        });
    } //End of Contructor Calculator
    // Perform action
    public void actionPerformed(ActionEvent e){

        double result = 0;

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

            JDialog dlgAbout = new CustomABOUTDialog(this,

                    "About Java Swing Calculator", true);

            dlgAbout.setVisible(true);

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

            System.exit(0);

        }

        // Search for the button pressed until end of array or key found

        for (int i=0; i&lt; 1)
            setDisplayString("0");

    }
              break;

         case 21:   // CE
    clearExisting();

            break;

         case 22:   // C

    clearAll();
            break;
}
         }
                    }

                    }  void setDisplayString(String s)  {
                    jlbOutput.setText(s);
                    }
                    String getDisplayString()  {
                    return jlbOutput.getText();
                    }
                    void addDigitToDisplay(int digit)  {
                    if  (clearOnNextDigit)
                    setDisplayString("");
                    String inputString = getDisplayString();
                    if (inputString.indexOf("0") == 0) {
                    inputString = inputString.substring(1);
                    }
                    if ((!inputString.equals("0") || digit &gt; 0)
                    &amp;&amp; inputString.length() &lt; MAX_INPUT_LENGTH) {
                    setDisplayString(inputString + digit);
                    }
                    displayMode = INPUT_MODE;
                    clearOnNextDigit = false;
                    }
                    void addDecimalPoint() {
                    displayMode = INPUT_MODE;
                    if (clearOnNextDigit)
                    setDisplayString("");
                    String inputString = getDisplayString();
                    // If the input string already contains a decimal point,
don't

                    //  do anything to it.
                    if (inputString.indexOf(".") &lt; 0)
                    setDisplayString(new String(inputString + "."));
                    }
                    void processSignChange() {
                    if (displayMode == INPUT_MODE) {
                    String input = getDisplayString();
                    if (input.length() &gt; 0 &amp;&amp; !input.equals("0"))
{

                    if (input.indexOf("-") == 0)
                    setDisplayString(input.substring(1));
                    else
                    setDisplayString("-" + input);
                    }
                    } else if (displayMode == RESULT_MODE) {
                    double numberInDisplay = getNumberInDisplay();
                    if (numberInDisplay != 0)
                    displayResult(-numberInDisplay);
                    }
                    }
                    void clearAll() {
                    setDisplayString("0");
                    lastOperator = "0";
                    lastNumber = 0;
                    displayMode = INPUT_MODE;
                    clearOnNextDigit = true;
                    }
                    void clearExisting() {
                    setDisplayString("0");
                    clearOnNextDigit = true;
                    displayMode = INPUT_MODE;
                    }
                    double getNumberInDisplay() {
```

```
String input = jlbOutput.getText();
return Double.parseDouble(input);
}
void processOperator(String op) {
if (displayMode != ERROR_MODE) {
double numberInDisplay = getNumberInDisplay();
if (!lastOperator.equals("0")) {
try {
double result = processLastOperator();
displayResult(result);
lastNumber = result;
} catch (DivideByZeroException e) {
}
} else {
lastNumber = numberInDisplay;
}
clearOnNextDigit = true;
lastOperator = op;
}
}
void processEquals() {
double result = 0;
if (displayMode != ERROR_MODE) {
try {
result = processLastOperator();
displayResult(result);
} catch (DivideByZeroException e) {
displayError("Cannot divide by zero!");
}
lastOperator = "0";
}
}
double processLastOperator() throws DivideByZeroException
{

double result = 0;
double numberInDisplay = getNumberInDisplay();
if (lastOperator.equals("/")) {
if (numberInDisplay == 0)
throw (new DivideByZeroException());
result = lastNumber / numberInDisplay;
}
if (lastOperator.equals("*"))
result = lastNumber * numberInDisplay;
if (lastOperator.equals("-"))
result = lastNumber - numberInDisplay;
if (lastOperator.equals("+"))
result = lastNumber + numberInDisplay;
return result;
}
void displayResult(double result) {
```

```
                    setDisplayString(Double.toString(result));
                    lastNumber = result;
                    displayMode = RESULT_MODE;
                    clearOnNextDigit = true;
                    }
                    void displayError(String errorMessage) {
                    setDisplayString(errorMessage);
                    lastNumber = 0;
                    displayMode = ERROR_MODE;
                    clearOnNextDigit = true;
                    }
public static void main(String args[]) {
        Calculator calci = new Calculator();
        Container contentPane = calci.getContentPane();
        //    contentPane.setLayout(new BorderLayout());
        calci.setTitle("Java Swing Calculator");
        calci.setSize(241, 217);
        calci.pack();
        calci.setLocation(400, 250);
        calci.setVisible(true);
        calci.setResizable(false);
        }
        } //End of Swing Calculator Class.

class DivideByZeroException extends Exception {

    public DivideByZeroException() {
        super();
    }
    public DivideByZeroException(String s) {
        super(s);
    }
}

class CustomABOUTDialog extends JDialog implements ActionListener {

    JButton jbnOk;
    CustomABOUTDialog(JFrame parent, String title, boolean modal) {
        super(parent, title, modal);
        setBackground(Color.black);
        JPanel p1 = new JPanel(new FlowLayout(FlowLayout.CENTER));
        StringBuffer text = new StringBuffer();
        text.append("Calculator Information\n\n");
        text.append("Developer:    Hemanth\n");
        text.append("Version:  1.0");
        JTextArea jtAreaAbout = new JTextArea(5, 21);
        jtAreaAbout.setText(text.toString());
        jtAreaAbout.setFont(new Font("Times New Roman", 1, 13));
        jtAreaAbout.setEditable(false);
        p1.add(jtAreaAbout);
        p1.setBackground(Color.red);
        getContentPane().add(p1, BorderLayout.CENTER);
        JPanel p2 = new JPanel(new FlowLayout(FlowLayout.CENTER));
        jbnOk = new JButton(" OK ");
        jbnOk.addActionListener(this);
        p2.add(jbnOk);
        getContentPane().add(p2, BorderLayout.SOUTH);
        setLocation(408, 270);
        setResizable(false);
        addWindowListener(new WindowAdapter() {

            public void windowClosing(WindowEvent e) {
                Window aboutDialog = e.getWindow();
                aboutDialog.dispose();
            }
        });
        pack();
    }
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == jbnOk) {
            this.dispose();
        }
    }
}
```

OUTPUT



Java simple calculator

Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.			
2.			
3.			