Facultatea Calculatoare, Informatica si Microelectronica

Universitatea Tehnica a Moldovei

----------------------------------------------------------------

Medii Interactive de Dezvoltare a Produselor Soft

----------------------------------------------------------------

Lucrar de Laborator#3

GUI DEVELOPMENT

Autor: Lector Asistent: Carp Ion Irina Cojanu

**Lucrare de Laborator nr. 3**

**Scopul lucrarii:**

Realizarea unui simplu GUI Calculator

**Obiective:**

* Realizeaza un simplu GUI Calculator
* Operatiile simple: +,-,\*,/,putere,radical,InversareSemn(+/-),operatii cu numere zecimale.
* Divizare proiectului in doua module - Interfata grafica(Modul GUI) si Modulul de baza(Core Module).

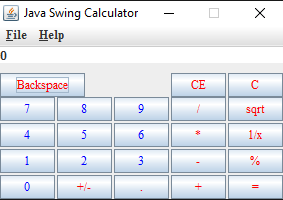
**Implimentarea Programului**

**Listingul Programului**

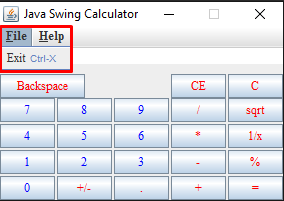
import java.awt.BorderLayout;  
import java.awt.Color;  
import java.awt.Container;  
import java.awt.FlowLayout;  
import java.awt.Font;  
import java.awt.GridLayout;  
import java.awt.Window;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.awt.event.KeyEvent;  
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  
 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 f12 = new Font("Times New Roman", 0, 12);  
 Font f121 = new Font("Times New Roman", 1, 12);  
   
 public Calculator()  
 {  
  
 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);  
  
  
 setBackground(Color.*gray*);  
  
 jplMaster = new JPanel();  
  
 jlbOutput = new JLabel("0");  
 jlbOutput.setHorizontalTextPosition(JLabel.*RIGHT*);  
 jlbOutput.setBackground(Color.*WHITE*);  
 jlbOutput.setOpaque(true);  
  
 getContentPane().add(jlbOutput, BorderLayout.*NORTH*);  
  
 jbnButtons = new JButton[23];  
  
 JPanel jplButtons = new JPanel();  
  
 for (int i=0; i<=9; i++)  
 {  
 jbnButtons[i] = new JButton(String.*valueOf*(i));  
 }  
  
 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<jbnButtons.length; i++) {  
 jbnButtons[i].setFont(f12);  
  
 if (i<10)  
 jbnButtons[i].setForeground(Color.*blue*);  
  
 else  
 jbnButtons[i].setForeground(Color.*red*);  
 }  
  
 jplButtons.setLayout(new GridLayout(4, 5, 2, 2));  
  
  
 for(int i=7; i<=9; i++) {  
 jplButtons.add(jbnButtons[i]);  
 }  
  
 jplButtons.add(jbnButtons[13]);  
 jplButtons.add(jbnButtons[17]);  
  
 for(int i=4; i<=6; i++)  
 {  
 jplButtons.add(jbnButtons[i]);  
 }  
  
 jplButtons.add(jbnButtons[14]);  
 jplButtons.add(jbnButtons[18]);  
  
 for( int i=1; i<=3; i++)  
 {  
 jplButtons.add(jbnButtons[i]);  
 }  
  
 jplButtons.add(jbnButtons[15]);  
 jplButtons.add(jbnButtons[19]);  
  
 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*);  
  
 getContentPane().add(jplMaster, BorderLayout.*SOUTH*);  
 requestFocus();  
  
 for (int i=0; i<jbnButtons.length; i++){  
 jbnButtons[i].addActionListener(this);  
 }  
  
 jmenuitemAbout.addActionListener(this);  
 jmenuitemExit.addActionListener(this);  
  
 clearAll();  
  
 addWindowListener(new WindowAdapter() {  
  
 public void windowClosed(WindowEvent e)  
 {  
 System.*exit*(0);  
 }  
 }  
 );  
 this.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 }  
  
 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);  
 }  
  
 for (int i=0; i<jbnButtons.length; i++)  
 {  
 if(e.getSource() == jbnButtons[i])  
 {  
 switch(i)  
 {  
 case 0:  
 addDigitToDisplay(i);  
 break;  
  
 case 1:  
 addDigitToDisplay(i);  
 break;  
  
 case 2:  
 addDigitToDisplay(i);  
 break;  
  
 case 3:  
 addDigitToDisplay(i);  
 break;  
  
 case 4:  
 addDigitToDisplay(i);  
 break;  
  
 case 5:  
 addDigitToDisplay(i);  
 break;  
  
 case 6:  
 addDigitToDisplay(i);  
 break;  
  
 case 7:  
 addDigitToDisplay(i);  
 break;  
  
 case 8:  
 addDigitToDisplay(i);  
 break;  
  
 case 9:  
 addDigitToDisplay(i);  
 break;  
  
 case 10:  
 processSignChange();  
 break;  
  
 case 11:  
 addDecimalPoint();  
 break;  
  
 case 12:  
 processEquals();  
 break;  
  
 case 13:  
 processOperator("/");  
 break;  
  
 case 14:  
 processOperator("\*");  
 break;  
  
 case 15:  
 processOperator("-");  
 break;  
  
 case 16:  
 processOperator("+");  
 break;  
  
 case 17:  
 if (displayMode != ERROR\_MODE)  
 {  
 try  
 {  
 if (getDisplayString().indexOf("-") == 0)  
 displayError("Invalid input for function!");  
  
 result = Math.*sqrt*(getNumberInDisplay());  
 displayResult(result);  
 }  
  
 catch(Exception ex)  
 {  
 displayError("Invalid input for function!");  
 displayMode = ERROR\_MODE;  
 }  
 }  
 break;  
  
 case 18: // 1/x  
 if (displayMode != ERROR\_MODE){  
 try  
 {  
 if (getNumberInDisplay() == 0)  
 displayError("Cannot divide by zero!");  
  
 result = 1 / getNumberInDisplay();  
 displayResult(result);  
 }  
  
 catch(Exception ex) {  
 displayError("Cannot divide by zero!");  
 displayMode = ERROR\_MODE;  
 }  
 }  
 break;  
  
 case 19:  
 if (displayMode != ERROR\_MODE){  
 try {  
 result = getNumberInDisplay() / 100;  
 displayResult(result);  
 }  
  
 catch(Exception ex) {  
 displayError("Invalid input for function!");  
 displayMode = ERROR\_MODE;  
 }  
 }  
 break;  
  
 case 20:  
 if (displayMode != ERROR\_MODE){  
 setDisplayString(getDisplayString().substring(0,  
 getDisplayString().length() - 1));  
  
 if (getDisplayString().length() < 1)  
 setDisplayString("0");  
 }  
 break;  
  
 case 21:  
 clearExisting();  
 break;  
  
 case 22:  
 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 > 0) && inputString.length() < MAX\_INPUT\_LENGTH){  
 setDisplayString(inputString + digit);  
 }  
  
  
 displayMode = INPUT\_MODE;  
 clearOnNextDigit = false;  
 }  
  
 void addDecimalPoint(){  
 displayMode = INPUT\_MODE;  
  
 if (clearOnNextDigit)  
 setDisplayString("");  
  
 String inputString = getDisplayString();  
  
 if (inputString.indexOf(".") < 0)  
 setDisplayString(new String(inputString + "."));  
 }  
  
 void processSignChange(){  
 if (displayMode == INPUT\_MODE)  
 {  
 String input = getDisplayString();  
  
 if (input.length() > 0 && !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();  
 calci.setTitle("Java Swing Calculator");  
 calci.setSize(241, 217);  
 calci.pack();  
 calci.setLocation(400, 250);  
 calci.setVisible(true);  
 calci.setResizable(false);  
 }  
  
}  
  
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: UTM-Carp Ion-TI142\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();  
 }  
 }  
  
}

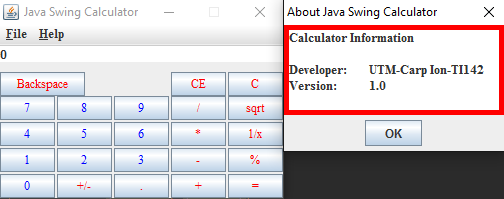
**--------------------------------------------------------------------------------**

**Captura de ecran**

****

**--------------------------------------------------------------------------------**

****

****

**Concluzie:**

In urma efectuarii acestei lucrari de laborator am facut cunostinta cu modulul GDI al programului NetBeans astfel am creat un simplu calculator in limbajul Java, avind functiile de baza +,-,\*,/,putere,radical,schimbarea semnului. Efectuind aceasta sarcina am luat cunostinta cu limbajul Java care este un limbaj usor de implimentat in cod si poate fi usor construit un calculator simplu utilizind butoane, si casete de text.