Министерство Образования Республики Молдова

Технический Университет Молдовы

Кафедра Автоматики и Информационных Технологий

Лабораторная работа №3

По дисциплине: «MIDPS»

Тема: Java Калькулятор

Выполнил: студент группы TI-145:

Батенко Вадим

Кишинёв 2016

```
importjava.awt.BorderLayout;
importjava.awt.Color;
importjava.awt.Container;
importjava.awt.FlowLayout;
importjava.awt.Font;
importjava.awt.GridLayout;
importjava.awt.Window;
importjava.awt.event.ActionEvent;
importjava.awt.event.ActionListener;
importjava.awt.event.KeyEvent;
importjava.awt.event.WindowAdapter;
importjava.awt.event.WindowEvent;

importjavax.swing.JButton;
importjavax.swing.JDialog;
importjavax.swing.JFrame;
importjavax.swing.JLabel;
importjavax.swing.JMenu;
importjavax.swing.JMenuBar;
importjavax.swing.JMenuItem;
importjavax.swing.JPanel;
importjavax.swing.JTextArea;
importjavax.swing.KeyStroke;

publicclassMidps3extendsJFrameimplementsActionListener{
      finalint MAX_INPUT_LENGTH = 20;
      finalint INPUT_MODE = 0;
      finalint RESULT_MODE = 1;
      finalint ERROR_MODE = 2;
      intdisplayMode;

      booleanclearOnNextDigit, percent;
      doublelastNumber;
      StringlastOperator;

      privateJMenujmenuFile;
      privateJMenuItemjmenuitemExit, jmenuitemAbout;

      privateJLabeljlbOutput;
      privateJButtonjbnButtons[];
      privateJPaneljplMaster, jplBackSpace, jplControl;

      Font f12 = newFont("Times New Roman", 0, 12);
      Font f121 = newFont("Times New Roman", 1, 12);

      publicMidps3()
      {

            jmenuFile = newJMenu("File");
            jmenuFile.setFont(f121);
            jmenuFile.setMnemonic(KeyEvent.VK_F);

            jmenuitemExit = newJMenuItem("Exit");
            jmenuitemExit.setFont(f12);
            jmenuitemExit.setAccelerator(KeyStroke.getKeyStroke( KeyEvent.VK_X,
                                                  ActionEvent.CTRL_MASK));
            jmenuFile.add(jmenuitemExit);



            jmenuitemAbout = newJMenuItem("About Calculator");
            jmenuitemAbout.setFont(f12);
```

```
		JMenuBarmb = newJMenuBar();
		mb.add(jmenuFile);
		setJMenuBar(mb);

		//Set frame layout manager

		setBackground(Color.gray);

		jplMaster = newJPanel();

		jlbOutput = newJLabel("0");
		jlbOutput.setHorizontalTextPosition(JLabel.RIGHT);
		jlbOutput.setBackground(Color.WHITE);
		jlbOutput.setOpaque(true);

		// Add components to frame
		getContentPane().add(jlbOutput, BorderLayout.NORTH);

		jbnButtons = newJButton[23];
//		GridLayout(int rows, intcols, inthgap, intvgap)

		JPaneljplButtons = newJPanel();		// container for Jbuttons

		for (inti=0; i<=9; i++)
		{
			// set each Jbutton label to the value of index
			jbnButtons[i] = newJButton(String.valueOf(i));
		}

		// Create operator Jbuttons
		jbnButtons[10] = newJButton("+/-");
		jbnButtons[11] = newJButton(".");
		jbnButtons[12] = newJButton("=");
		jbnButtons[13] = newJButton("/");
		jbnButtons[14] = newJButton("*");
		jbnButtons[15] = newJButton("-");
		jbnButtons[16] = newJButton("+");
		jbnButtons[17] = newJButton("sqrt");
		jbnButtons[18] = newJButton("1/x");
		jbnButtons[19] = newJButton("%");

		jplBackSpace = newJPanel();
		jplBackSpace.setLayout(newGridLayout(1, 1, 2, 2));

		jbnButtons[20] = newJButton("Backspace");
		jplBackSpace.add(jbnButtons[20]);

		jplControl = newJPanel();
		jplControl.setLayout(newGridLayout(1, 2, 2 ,2));

		jbnButtons[21] = newJButton(" CE ");
		jbnButtons[22] = newJButton("C");

		jplControl.add(jbnButtons[21]);
		jplControl.add(jbnButtons[22]);

//		Setting all Numbered JButton's to Blue. The rest to Red
		for (inti=0; i<jbnButtons.length; i++) {
			jbnButtons[i].setFont(f12);

			if (i<10)
```

```
            jbnButtons[i].setForeground(Color.blue);

      else
            jbnButtons[i].setForeground(Color.red);
}

// Set panel layout manager for a 4 by 5 grid
jplButtons.setLayout(newGridLayout(4, 5, 2, 2));

//Add buttons to keypad panel starting at top left
// First row
for(inti=7; i<=9; i++)          {
      jplButtons.add(jbnButtons[i]);
}

// add button / and sqrt
jplButtons.add(jbnButtons[13]);
jplButtons.add(jbnButtons[17]);

// Second row
for(inti=4; i<=6; i++)
{
      jplButtons.add(jbnButtons[i]);
}

// add button * and x^2
jplButtons.add(jbnButtons[14]);
jplButtons.add(jbnButtons[18]);

// Third row
for( inti=1; i<=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(newBorderLayout());
jplMaster.add(jplBackSpace, BorderLayout.WEST);
jplMaster.add(jplControl, BorderLayout.EAST);
jplMaster.add(jplButtons, BorderLayout.SOUTH);

getContentPane().add(jplMaster, BorderLayout.SOUTH);
requestFocus();

for (inti=0; i<jbnButtons.length; i++){
      jbnButtons[i].addActionListener(this);
}

jmenuitemAbout.addActionListener(this);
jmenuitemExit.addActionListener(this);
```

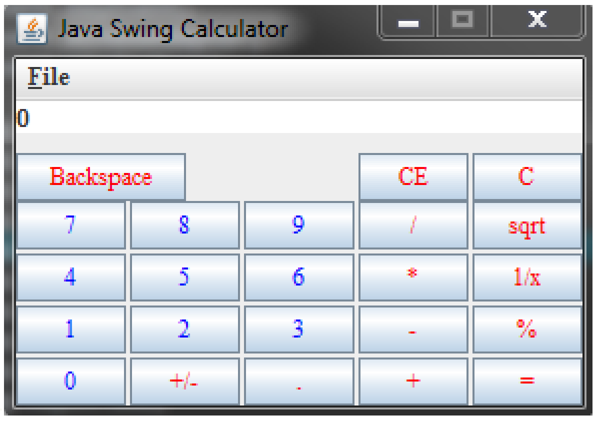
```
		clearAll();


		addWindowListener(newWindowAdapter() {

				publicvoidwindowClosed(WindowEvent e)
				{
					System.exit(0);
				}
			}
		);
	}	//End of Contructor Calculator

	// Perform action
	publicvoidactionPerformed(ActionEvent e){
		doubleresult = 0;

		if(e.getSource() == jmenuitemAbout){
		JDialogdlgAbout = newCustomABOUTDialog(this,
								"About Java Swing Calculator",
true);
			dlgAbout.setVisible(true);
		}elseif(e.getSource() == jmenuitemExit){
			System.exit(0);
		}

		// Search for the button pressed until end of array or key found
		for (inti=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:        // decimal point
                                        addDecimalPoint();
                                        break;

                                case 12:        // =
                                        processEquals();
                                        break;

                                case 13:        // divide
                                        processOperator("/");
                                        break;

                                case 14:        // *
                                        processOperator("*");
                                        break;

                                case 15:        // -
                                        processOperator("-");
                                        break;

                                case 16:        // +
                                        processOperator("+");
                                        break;

                                case 17:        // sqrt
                                        if (displayMode != ERROR_MODE)
                                        {
                                        try
                                                {
                                                        if (getDisplayString().indexOf("-
") == 0)

                                                        displayError("Invalid input for

function!");


                                                        result =

Math.sqrt(getNumberInDisplay());

                                                        displayResult(result);
                                                }


                                                catch(Exceptionex)
                                                {
                                                        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(Exceptionex) {
                                                  displayError("Cannot divide by
zero!");

                                                  displayMode = ERROR_MODE;
                                            }
                                      }
                                      break;

                                case 19:     // %
                                      if (displayMode != ERROR_MODE){
                                            try   {
                                                  result = getNumberInDisplay() /
100;

                                                  displayResult(result);
                                            }

                                            catch(Exceptionex) {
                                                  displayError("Invalid input for
function!");

                                                  displayMode = ERROR_MODE;
                                            }
                                      }
                                      break;

                                case 20:     // backspace
                                      if (displayMode != ERROR_MODE){

      setDisplayString(getDisplayString().substring(0,
                                                        getDisplayString().length()
- 1));

                                            if (getDisplayString().length() < 1)
                                                  setDisplayString("0");
                                      }
                                      break;

                                case 21:     // CE
                                      clearExisting();
                                      break;

                                case 22:     // C
                                      clearAll();
                                      break;
                          }
                    }
              }
      }
```

```
        voidsetDisplayString(String s){
                jlbOutput.setText(s);
        }

        StringgetDisplayString(){
                returnjlbOutput.getText();
        }

        voidaddDigitToDisplay(int digit){
                if (clearOnNextDigit)
                        setDisplayString("");

                StringinputString = 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;
        }

        voidaddDecimalPoint(){
                displayMode = INPUT_MODE;

                if (clearOnNextDigit)
                        setDisplayString("");

                StringinputString = getDisplayString();

                // If the input string already contains a decimal point, don't
                //  do anything to it.
                if (inputString.indexOf(".") < 0)
                        setDisplayString(new String(inputString + "."));
        }

        voidprocessSignChange(){
                if (displayMode == INPUT_MODE)
                {
                        Stringinput = getDisplayString();

                        if (input.length() > 0 && !input.equals("0"))
                        {
                                if (input.indexOf("-") == 0)
                                        setDisplayString(input.substring(1));

                                else
                                        setDisplayString("-" + input);
                        }

                }

                elseif (displayMode == RESULT_MODE)
                {
                        doublenumberInDisplay = getNumberInDisplay();
```

```
            if (numberInDisplay != 0)
                displayResult(-numberInDisplay);
        }
    }

    voidclearAll()     {
        setDisplayString("0");
        lastOperator = "0";
        lastNumber = 0;
        displayMode = INPUT_MODE;
        clearOnNextDigit = true;
    }

    voidclearExisting(){
        setDisplayString("0");
        clearOnNextDigit = true;
        displayMode = INPUT_MODE;
    }

    doublegetNumberInDisplay()      {
        Stringinput = jlbOutput.getText();
        returnDouble.parseDouble(input);
    }

    voidprocessOperator(String op) {
        if (displayMode != ERROR_MODE)
        {
            doublenumberInDisplay = getNumberInDisplay();

            if(!lastOperator.equals("0"))
            {
                try
                {
                    doubleresult = processLastOperator();
                    displayResult(result);
                    lastNumber = result;
                }

                catch (DivideByZeroExceptione)
                {
                }
            }

            else
            {
                lastNumber = numberInDisplay;
            }

            clearOnNextDigit = true;
            lastOperator = op;
        }
    }

    voidprocessEquals(){
        doubleresult = 0;

        if (displayMode != ERROR_MODE){
            try
            {
                result = processLastOperator();
                displayResult(result);
                }

                catch (DivideByZeroExceptione)  {
                      displayError("Cannot divide by zero!");
                }

                lastOperator = "0";
          }
    }

    doubleprocessLastOperator() throwsDivideByZeroException {
          doubleresult = 0;
          doublenumberInDisplay = getNumberInDisplay();

          if (lastOperator.equals("/"))
          {
                if (numberInDisplay == 0)
                      throw (newDivideByZeroException());

                result = lastNumber / numberInDisplay;
          }

          if (lastOperator.equals("*"))
                result = lastNumber * numberInDisplay;

          if (lastOperator.equals("-"))
                result = lastNumber - numberInDisplay;

          if (lastOperator.equals("+"))
                result = lastNumber + numberInDisplay;

          return result;
    }

    voiddisplayResult(double result){
          setDisplayString(Double.toString(result));
          lastNumber = result;
          displayMode = RESULT_MODE;
          clearOnNextDigit = true;
    }

    voiddisplayError(StringerrorMessage){
          setDisplayString(errorMessage);
          lastNumber = 0;
          displayMode = ERROR_MODE;
          clearOnNextDigit = true;
    }

    publicstaticvoidmain(Stringargs[]) {
          Midps3calci = new Midps3();
          ContainercontentPane = 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.
```

```
classDivideByZeroExceptionextendsException{
	publicDivideByZeroException()
	{
		super();
	}

	publicDivideByZeroException(String s)
	{
		super(s);
	}
}

classCustomABOUTDialogextendsJDialogimplementsActionListener {
	JButtonjbnOk;

	CustomABOUTDialog(JFrame parent, String title, boolean modal){
		super(parent, title, modal);
		setBackground(Color.black);

		JPanelp1 = newJPanel(newFlowLayout(FlowLayout.CENTER));

		StringBuffertext = newStringBuffer();


		JTextAreajtAreaAbout = newJTextArea(5, 21);
		jtAreaAbout.setText(text.toString());
		jtAreaAbout.setFont(newFont("Times New Roman", 1, 13));
		jtAreaAbout.setEditable(false);

		p1.add(jtAreaAbout);
		p1.setBackground(Color.red);
		getContentPane().add(p1, BorderLayout.CENTER);

		JPanelp2 = newJPanel(newFlowLayout(FlowLayout.CENTER));
		jbnOk = newJButton(" OK ");
		jbnOk.addActionListener(this);

		p2.add(jbnOk);
		getContentPane().add(p2, BorderLayout.SOUTH);

		setLocation(408, 270);
		setResizable(false);

		addWindowListener(newWindowAdapter() {
				publicvoidwindowClosing(WindowEvent e)
				{
					WindowaboutDialog = e.getWindow();
					aboutDialog.dispose();
				}
			}
		);

		pack();
	}

	publicvoidactionPerformed(ActionEvent e)
	{
		if(e.getSource() == jbnOk)	{
			this.dispose();
		}
	}
```


Java Swing Calculator
File
0
Backspace
CE
C
7
8
9
/
sqrt
4
5
6
*
1/x
1
2
3
-
%
0
+/-
.
+
=