Министерство образования и науки Российской Федерации Федеральное агентство по образованию

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

**Разработка пользовательского интерфейса с использований технологий Swing**

Лабораторная работа №3

по курсу «Разработка программных систем»

Выполнил студент группы ИВТб-31 /Категов А. Д./ Проверил преподаватель /Чистяков Г.А./

Киров 2024

Задание:

Выбрать и согласовать с преподавателем задачу, для решения которой может быть использована программа, разработанная в ходе предыдущей лабораторной работы. Разработать графическое приложение с использованием библиотеки Swing.

1. Исходный код с комментариями

Листинг приведен в приложении А.

1. Экранные формы

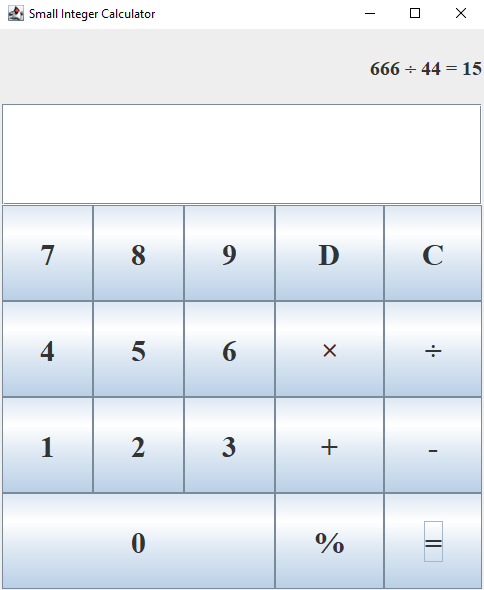


Рисунок 1 – Пользовательский интерфейс

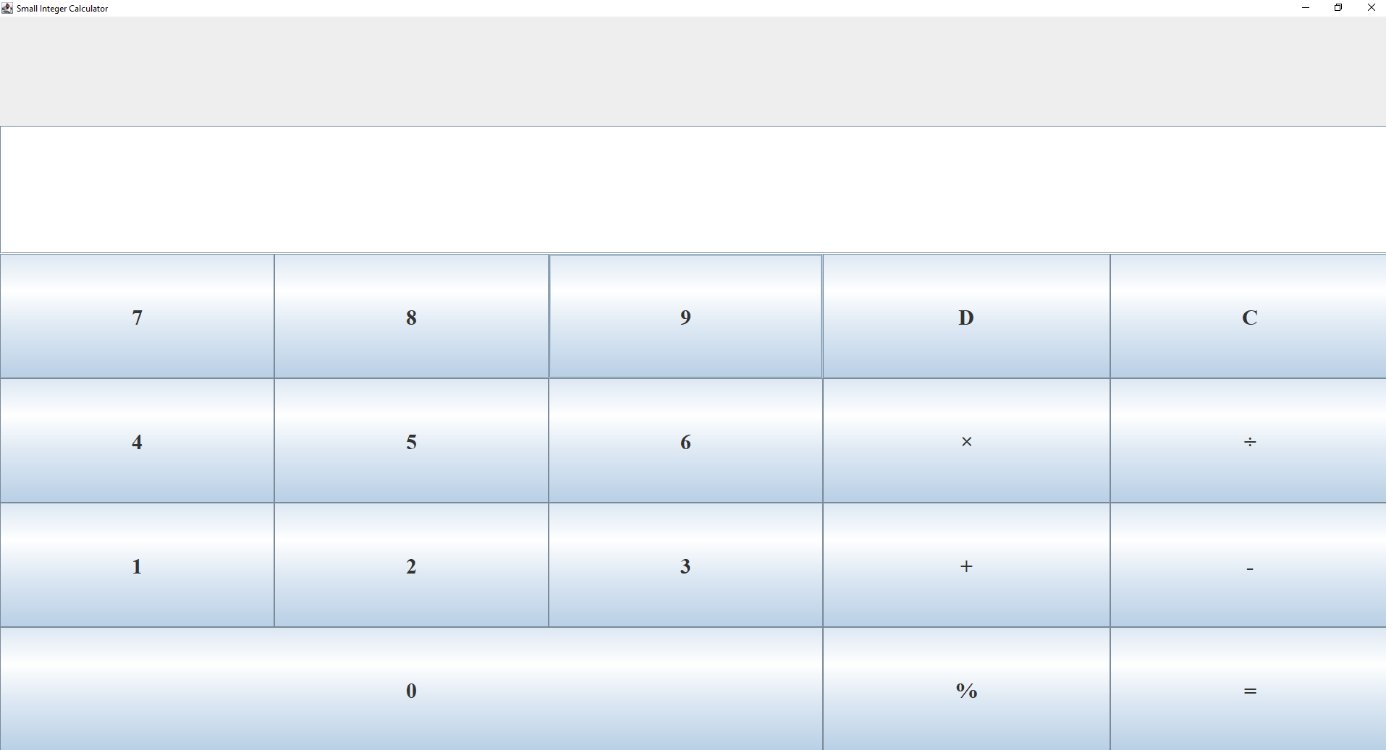


Рисунок 2 – Калькулятор в развёрнутом виде

**Вывод**: в ходе данной лабораторной работы были изучены основы технологии Swing, и её основные компоненты. На основе полученных знаний было написано приложение с графическим пользовательским интерфейсом. На основании всего сказанного выше можно сделать вывод о том, что цель достигнута, необходимые знания получены, задание выполнено верно, что доказывается полученными результатами.

Приложение А

(обязательное)

Листинг программы

**public** **class** Main {

**public** **static** **void** main(String[] args) {

CalculatorGUI app = **new** CalculatorGUI();

app.setVisible(**true**);

}

}

/\*\*

\* Class for handling integers within the range [-10000, 10000].

\*

\* **@author** Alexandr Kategov

\* **@version** 1.0

\*/

**public** **class** SmallInteger {

**private** **int** value; // Stores the value of the number

**private** **static** **final** **int** ***MAX\_VALUE*** = 10000; // Maximum allowable value

**private** **static** **final** **int** ***MIN\_VALUE*** = -10000; // Minimum allowable value

/\*\*

\* Primary constructor that accepts an integer and checks if it is within the valid range.

\*

\* **@param** value the integer value to be stored

\* **@throws** RuntimeException if the value is out of the valid range [-10000, 10000]

\*/

**public** SmallInteger(**int** value) {

**if** (value > ***MAX\_VALUE*** || value < ***MIN\_VALUE***) {

**throw** **new** RuntimeException("Value out of range");

}

**this**.value = value;

}

/\*\*

\* Copy constructor that creates a new object based on an existing one.

\*

\* **@param** other the SmallInteger object to copy

\*/

**public** SmallInteger(SmallInteger other) {

**this**(other.value); // Calls the primary constructor

}

/\*\*

\* Adds two SmallInteger objects.

\*

\* **@param** arg the SmallInteger to add

\* **@return** a new SmallInteger containing the result of the addition

\* **@throws** RuntimeException if the result is out of the valid range [-10000, 10000]

\*/

**public** SmallInteger add(SmallInteger arg) {

**int** result = **this**.value + arg.value;

**if** (result > ***MAX\_VALUE*** || result < ***MIN\_VALUE***) {

**throw** **new** RuntimeException("Result out of range");

}

**return** **new** SmallInteger(result);

}

/\*\*

\* Subtracts one SmallInteger object from another.

\*

\* **@param** arg the SmallInteger to subtract

\* **@return** a new SmallInteger containing the result of the subtraction

\* **@throws** RuntimeException if the result is out of the valid range [-10000, 10000]

\*/

**public** SmallInteger sub(SmallInteger arg) {

**int** result = **this**.value - arg.value;

**if** (result > ***MAX\_VALUE*** || result < ***MIN\_VALUE***) {

**throw** **new** RuntimeException("Result out of range");

}

**return** **new** SmallInteger(result);

}

/\*\*

\* Multiplies two SmallInteger objects.

\*

\* **@param** arg the SmallInteger to multiply by

\* **@return** a new SmallInteger containing the result of the multiplication

\* **@throws** RuntimeException if the result is out of the valid range [-10000, 10000]

\*/

**public** SmallInteger mul(SmallInteger arg) {

**int** result = **this**.value \* arg.value;

**if** (result > ***MAX\_VALUE*** || result < ***MIN\_VALUE***) {

**throw** **new** RuntimeException("Result out of range");

}

**return** **new** SmallInteger(result);

}

/\*\*

\* Performs integer division of one SmallInteger object by another.

\*

\* **@param** arg the SmallInteger to divide by

\* **@return** a new SmallInteger containing the result of the division

\* **@throws** ArithmeticException if division by zero occurs

\*/

**public** SmallInteger integerDivision(SmallInteger arg) {

**if** (arg.value == 0) {

**throw** **new** ArithmeticException("Division by zero");

}

**int** result = **this**.value / arg.value;

**return** **new** SmallInteger(result);

}

/\*\*

\* Finds the remainder of the division of one SmallInteger object by another.

\*

\* **@param** arg the SmallInteger to divide by

\* **@return** a new SmallInteger containing the remainder of the division

\* **@throws** ArithmeticException if division by zero occurs

\*/

**public** SmallInteger remainderDivision(SmallInteger arg) {

**if** (arg.value == 0) {

**throw** **new** ArithmeticException("Division by zero");

}

**int** result = **this**.value % arg.value;

**return** **new** SmallInteger(result);

}

/\*\*

\* Gets the current value.

\*

\* **@return** the integer value stored in this SmallInteger

\*/

**public** **int** getValue() {

**return** **this**.value;

}

}

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

public class CalculatorGUI extends JFrame {

/\*\*

\* Variables for the arguments

\*/

private SmallInteger arg1 = new SmallInteger(0);

private SmallInteger arg2 = new SmallInteger(0);

private SmallInteger answer = new SmallInteger(0);

/\*\*

\* Buttons for numbers

\*/

private JButton zeroButton = new JButton("0");

private JButton oneButton = new JButton("1");

private JButton twoButton = new JButton("2");

private JButton threeButton = new JButton("3");

private JButton fourButton = new JButton("4");

private JButton fiveButton = new JButton("5");

private JButton sixButton = new JButton("6");

private JButton sevenButton = new JButton("7");

private JButton eightButton = new JButton("8");

private JButton nineButton = new JButton("9");

/\*\*

\* buttons for common operations

\*/

private JButton addButton = new JButton("+");

private JButton subButton = new JButton("-");

private JButton mulButton = new JButton("×");

private JButton divButton = new JButton("÷");

private JButton modButton = new JButton("%");

private JButton answerButton = new JButton("=");

/\*\*

\* buttons for non-standart operations

\*/

private JButton clearButton = new JButton("C");

private JButton deleteButton = new JButton("D");

/\*\*

\* fields for input and output

\*/

private JLabel outputField = new JLabel("");

private JTextField inputField = new JTextField("", 1);

/\*\*

\* GridBagLayout

\*/

private GridBagLayout gridBag = new GridBagLayout();

private GridBagConstraints parametrs = new GridBagConstraints();

private byte operationCounter = 0; // +, -, \*, /, %

private String label = " ";

private boolean end = false;

/\*\*

\* Constructor for main window

\*/

public CalculatorGUI() {

super("Small Integer Calculator");

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

this.setSize(500, 600);

this.setLocationRelativeTo(null);

outputField.setText(label);

setLayout(gridBag);

parametrs.weighty = 1.0;

parametrs.fill = GridBagConstraints.BOTH;

/\*\*

\* Input\_Field

\*/

parametrs.gridwidth = GridBagConstraints.REMAINDER;

gridBag.setConstraints(outputField, parametrs);

add(outputField);

outputField.setComponentOrientation(ComponentOrientation.RIGHT\_TO\_LEFT);

outputField.setFont(new Font("Serif", Font.BOLD, 20));

/\*\*

\* Text\_Field

\*/

gridBag.setConstraints(inputField, parametrs);

add(inputField);

inputField.setComponentOrientation(ComponentOrientation.RIGHT\_TO\_LEFT);

inputField.setFont(new Font("Serif", Font.BOLD, 40));

/\*\*

\* Buttons

\*/

makeButtons(1, 1.0, 0, 2, sevenButton);

makeButtons(1, 1.0, 1, 2, eightButton);

makeButtons(1, 1.0, 2, 2, nineButton);

makeButtons(1, 1.0, 3, 2, deleteButton);

makeButtons(1, 1.0, 4, 2, clearButton);

makeButtons(1, 1.0, 0, 3, fourButton);

makeButtons(1, 1.0, 1, 3, fiveButton);

makeButtons(1, 1.0, 2, 3, sixButton);

makeButtons(1, 1.0, 3, 3, mulButton);

makeButtons(1, 1.0, 4, 3, divButton);

makeButtons(1, 1.0, 0, 4, oneButton);

makeButtons(1, 1.0, 1, 4, twoButton);

makeButtons(1, 1.0, 2, 4, threeButton);

makeButtons(1, 1.0, 3, 4, addButton);

makeButtons(1, 1.0, 4, 4, subButton);

makeButtons(3, 1.0, 0, 5, zeroButton);

makeButtons(1, 1.0, 3, 5, modButton);

makeButtons(1, 1.0, 4, 5, answerButton);

}

/\*\*

\* Function for button initialization

\*/

public void makeButtons(int gridwidth, double weightx, int gridx, int gridy, JButton button) {

parametrs.gridwidth = gridwidth;

parametrs.weightx = weightx;

parametrs.gridx = gridx;

parametrs.gridy = gridy;

gridBag.setConstraints(button, parametrs);

add(button);

button.setFont(new Font("Serif", Font.BOLD, 30));

button.addActionListener(new ButtonListener());

}

/\*\*

\* Сlass for tracking button clicks

\*/

public class ButtonListener implements ActionListener {

public void actionPerformed(ActionEvent e) {

String line = inputField.getText();

try {

if (end) {

reset(line);

}

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

if (line.length() != 0) {

arg2 = new SmallInteger(Integer.parseInt(line));

switch (operationCounter) {

case 1:

answer = arg1.add(arg2);

break;

case 2:

answer = arg1.sub(arg2);

break;

case 3:

answer = arg1.mul(arg2);

break;

case 4:

answer = arg1.integerDivision(arg2);

break;

case 5:

answer = arg1.remainderDivision(arg2);

break;

}

label = label + line + " = " + answer.getValue();

outputField.setText(label);

line = "";

inputField.setText(line);

end = true;

}

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

reset(line);

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

if (line.length() != 0) {

line = line.substring(0, line.length() - 1);

}

inputField.setText(line);

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

operationButtons((byte) 1, line, " + ");

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

operationButtons((byte) 2, line, " - ");

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

operationButtons((byte) 3, line, " × ");

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

operationButtons((byte) 4, line, " ÷ ");

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

operationButtons((byte) 5, line, " % ");

} else {

handleNumberButtons(e, line);

}

} catch (Exception ee) {

reset(line);

outputField.setText("Error");

}

}

private void handleNumberButtons(ActionEvent e, String line) {

if (line.equals("0")) {

line = "";

}

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

inputField.setText(line + "0");

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

inputField.setText(line + "1");

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

inputField.setText(line + "2");

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

inputField.setText(line + "3");

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

inputField.setText(line + "4");

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

inputField.setText(line + "5");

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

inputField.setText(line + "6");

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

inputField.setText(line + "7");

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

inputField.setText(line + "8");

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

inputField.setText(line + "9");

}

}

/\*\*

\* Function for action on operation button's press

\*/

private void operationButtons(byte operation, String line, String operationSign) {

if (operationCounter == 0) {

label = label + line + operationSign;

outputField.setText(label);

operationCounter = operation;

arg1 = new SmallInteger(Integer.parseInt(line));

line = "";

inputField.setText(line);

}

}

private void reset(String line) {

label = " ";

outputField.setText(label);

operationCounter = 0;

line = "";

inputField.setText(line);

end = false;

}

}

}