Министерство науки и высшего образования Российской федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №2

по курсу «Программирование на языке Java»

на тему «Работа с коллекциями объектов»

Вариант 5

Выполнили

студенты группы 21ВВП2:

Лукина Е.Ю.

Ванюшин И.А.

Тарасов Н.А.

Приняли:

Юрова О.В.

Карамышева Н.С.

Пенза 2024

**Цель работы**

Изучить библиотеку стандартных коллекций Java Collections Framework, позволяющую хранить различные структуры данных.

**Задание на лабораторную работу**

Модифицировать приложение из предыдущей лабораторной работы, реализовав хранение данных таблицы с использованием библиотеки коллекций. Для этого реализовать класс RecIntegral, способный хранить одну запись таблицы. Для нечетных вариантов в качестве класса-коллекции выбрать ArrayList, для четных - LinkedList. Кроме того, добавить пару кнопок: очистить / заполнить, которые будут очищать таблицу и заполнять ее данными из коллекции соответственно. Оформление лабораторной работы должно быть выполнено в соответствии с требованиями, приведенными в Приложении 2.

|  |  |  |
| --- | --- | --- |
| Номер варианта | Функция | Класс-коллекция |
| 5 |  | ArrayList |

**Ход работы**

Написали класс для хранения записей таблицы.

Ниже представлены перегруженные конструкторы класса RecIntegral.

public RecIntegral()

{

limitL = "";

limitR = "";

dx = "";

result = "";

}

public RecIntegral(IntegralCalculator src)

{

limitL = String.valueOf(src.A());

limitR = String.valueOf(src.B());

dx = String.valueOf(src.E());

result = String.valueOf(src.F());

}

public RecIntegral(double \_limitL, double \_limitR, double \_dx, double \_result)

{

limitL = String.valueOf(\_limitL);

limitR = String.valueOf(\_limitR);

dx = String.valueOf(\_dx);

result = String.valueOf(\_result);

}

Дополнили графическое оформление для приложения, позволяющее сохранять результаты вычислений в таблице и коллекции, очищать таблицу, удалять строки таблицы и коллекции, загружать данные из коллекции в таблицу.

**Листинг**

RecIntegral.java

package lab1;

/\*\*

\*

\* @author User

\*/

public class RecIntegral {

public RecIntegral()

{

limitL = "";

limitR = "";

dx = "";

result = "";

}

public RecIntegral(IntegralCalculator src)

{

limitL = String.valueOf(src.A());

limitR = String.valueOf(src.B());

dx = String.valueOf(src.E());

result = String.valueOf(src.F());

}

public RecIntegral(double \_limitL, double \_limitR, double \_dx, double \_result)

{

limitL = String.valueOf(\_limitL);

limitR = String.valueOf(\_limitR);

dx = String.valueOf(\_dx);

result = String.valueOf(\_result);

}

private String limitL, limitR, dx, result;

public String LimitL() { return limitL; }

public String LimitR() { return limitR; }

public String Dx() { return dx; }

public String Result() { return result; }

public void SetLimitL(double \_limitL) { limitL = String.valueOf(\_limitL); }

public void SetLimitR(double \_limitR) { limitR = String.valueOf(\_limitR); }

public void SetDx(double \_dx) { dx = String.valueOf(\_dx); }

public void SetResult(double \_result) { result = String.valueOf(\_result); }

public void Clear()

{

limitL = "";

limitR = "";

dx = "";

result = "";

}

}

MyWin.java

package lab1;

import java.awt.Container;

import java.awt.Dimension;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.ArrayList;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.JTextField;

import javax.swing.table.DefaultTableModel;

public class MyWin extends JFrame implements ActionListener

{

// Some window's elements

private final JTextField jtfA;

private final JTextField jtfB;

private final JTextField jtfE;

private final DefaultTableModel dtResultsModel;

private final JTable jtResults;

private final ArrayList alRecs;

// Класс для вычисления определенного интеграла

private final IntegralCalculator ic;

// Classes serial number

private static final long serialVersionUID = 1L;

public MyWin()

{

// Initialization block

alRecs = new ArrayList();

ic = new IntegralCalculator(0, 1, 1e-6);

jtfA = new JTextField();

jtfB = new JTextField();

jtfE = new JTextField();

//----------------==Table Filling==------------------------------

Object[] columnNames = new String[]{"L limit", "R limit", "Step", "Result"};

Object[][] data = new String[][]{};

dtResultsModel = new DefaultTableModel(data, columnNames);

dtResultsModel.setColumnIdentifiers(columnNames);

jtResults = new JTable(dtResultsModel);

JScrollPane sp = new JScrollPane(jtResults);

Container c = getContentPane(); // Client rect

c.setLayout(null); // НЕНАВИЖУ КОМПАНОВЩИКИ

//----------------==Left Panel's Filling==-----------------------

// Creates a left sided panel

JPanel jpLeftPanel = new JPanel();

jpLeftPanel.setLayout(null); // АГРЕССИЯ И ЗУБЫ СКРИПЯТ

jpLeftPanel.setBounds(0, 0, 200, 480); // Resizes left panel

// Creates buttons.

// The one that Adds new records & the other that Clears all inputs.

JButton jbtnAdd = new JButton("Add");

JButton jbtnClear = new JButton("Clear");

JButton jbtnDelete = new JButton("Delete");

JButton jbtnLoad = new JButton("Load");

// Buttons's resize.

jbtnAdd.setBounds(10, 200, 180, 50);

jbtnClear.setBounds(10, 260, 180, 50);

jbtnLoad.setBounds(10, 320, 180, 50);

jbtnDelete.setBounds(10, 380, 180, 50);

// Adds action listeners for the objects.

jbtnAdd.addActionListener(this);

jbtnClear.addActionListener(this);

jbtnDelete.addActionListener(this);

jbtnLoad.addActionListener(this);

jtfA.setBounds(10, 10, 180, 50);

jtfB.setBounds(10, 70, 180, 50);

jtfE.setBounds(10, 130, 180, 50);

jtfA.setText(Double.toString(ic.A()));

jtfB.setText(Double.toString(ic.B()));

jtfE.setText(Double.toString(ic.E()));

sp.setBounds(200, 0, 427, 480);

// Fills panel with early created buttons & txtFields.

jpLeftPanel.add(jtfA);

jpLeftPanel.add(jtfB);

jpLeftPanel.add(jtfE);

jpLeftPanel.add(jbtnAdd);

jpLeftPanel.add(jbtnClear);

jpLeftPanel.add(jbtnDelete);

jpLeftPanel.add(jbtnLoad);

//----------------==Client rect's Filling==-----------------------

// Adds left panel to actually left side of client rect.

c.add(jpLeftPanel);

c.add(sp);

//----------------==Setting window's properties==-----------------

// Window's properties.

setTitle("Integral Calculator"); // Window's title. Obviously.

// Sets preffered size to window.

setPreferredSize(new Dimension(640, 480));

// Exit application on Exit button.

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

pack(); // Setting preffered sizes.

setVisible(true); // Makes window visible.

}

@Override

public void actionPerformed(ActionEvent e) {

// "Calculate" case. Calculates the integral with setted parameters.

if (e.getActionCommand().equals("Add"))

{

ic.SetArea(Double.parseDouble(jtfA.getText()), Double.parseDouble(jtfB.getText()));

ic.SetE(Double.parseDouble(jtfE.getText()));

alRecs.add(new RecIntegral(ic));

jtfE.setText(((RecIntegral)alRecs.getLast()).Dx());

dtResultsModel.addRow(new String[]

{

String.valueOf(ic.A()),

String.valueOf(ic.B()),

String.valueOf(ic.E()),

((RecIntegral)alRecs.getLast()).Result()

});

}

else if (e.getActionCommand().equals("Clear"))

{

dtResultsModel.setRowCount(0);

}

else if (e.getActionCommand().equals("Load"))

{

dtResultsModel.setRowCount(0);

for (int i = 0; i < alRecs.size(); i++)

{

dtResultsModel.addRow(new String[]

{

((RecIntegral)alRecs.get(i)).LimitL(),

((RecIntegral)alRecs.get(i)).LimitR(),

((RecIntegral)alRecs.get(i)).Dx(),

((RecIntegral)alRecs.get(i)).Result()

});

}

}

else if (e.getActionCommand().equals("Delete"))

{

int i = jtResults.getSelectedRow();

if (i != -1)

{

dtResultsModel.removeRow(i);

alRecs.remove(i);

}

}

else throw new UnsupportedOperationException("Not supported yet."); // Unprocessed action case.

}

// запуск оконного приложения

public static void main(String args[]) {

new MyWin();

}

}

**Результат работы программы**

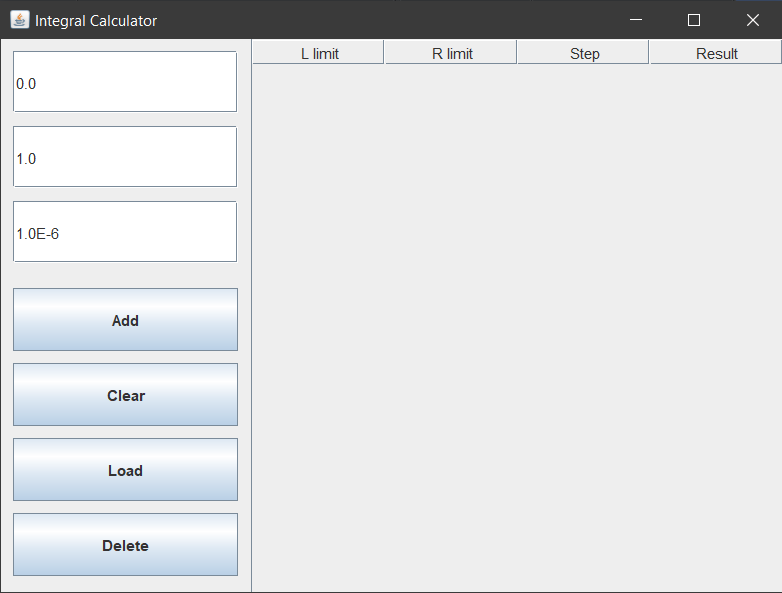
Результаты работы программы представлены на рисунках 1-5. 

Рис.1 – Состояние программы после запуска

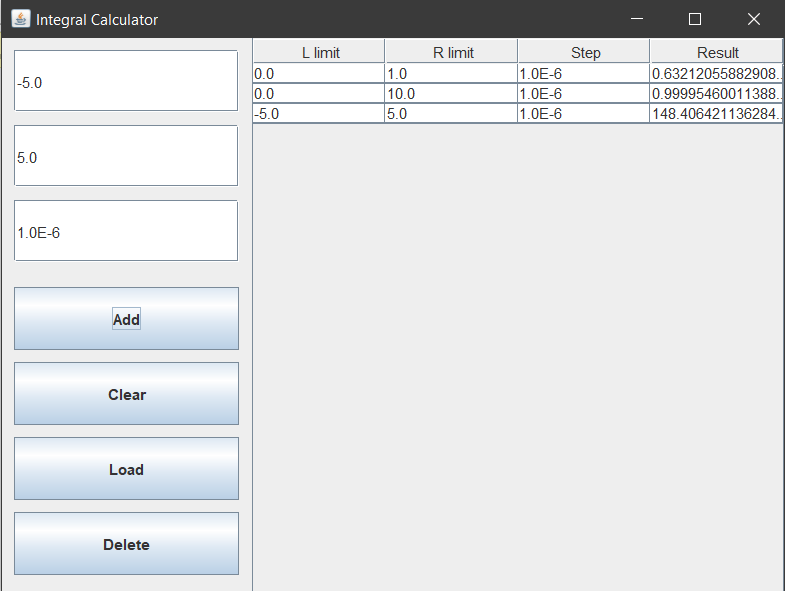


Рис.2 – Состояние программы после добавления нескольких записей в таблицу и коллекцию

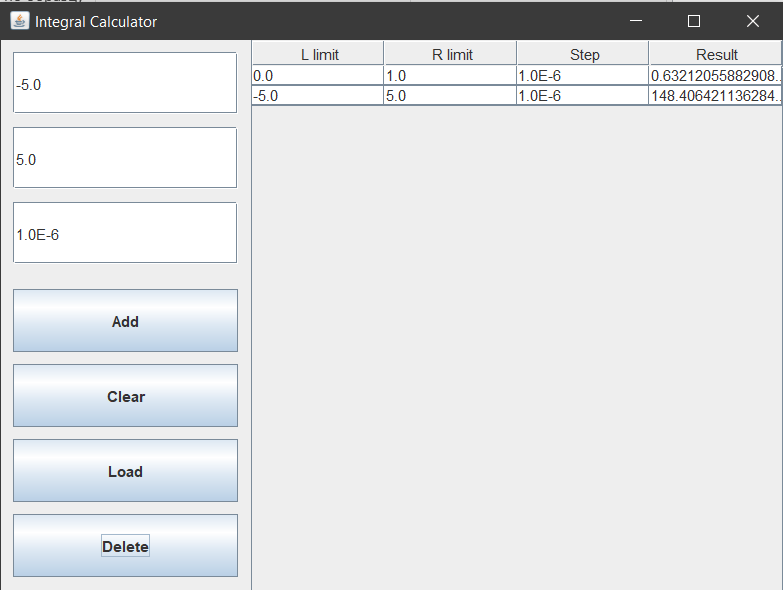


Рис.3 – Состояние программы после удаления строки 2

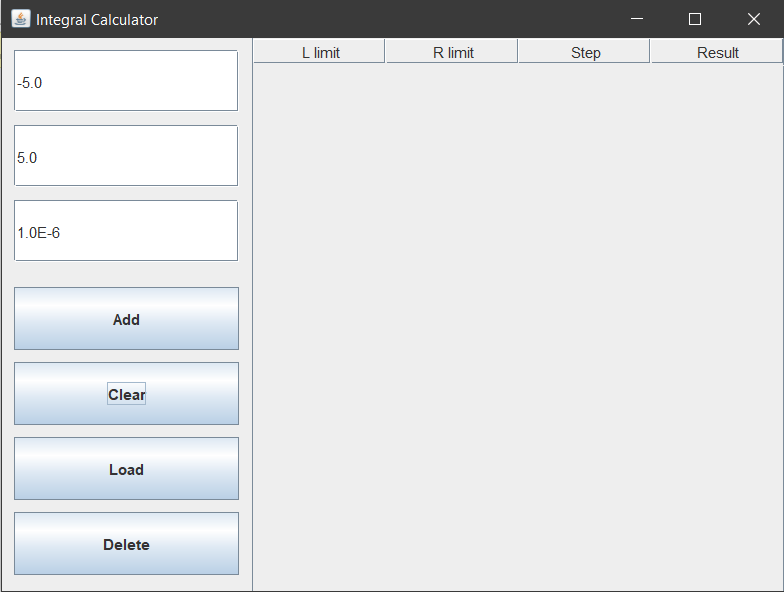


Рис.4 – Состояние программы после очистки таблицы кнопкой Clear

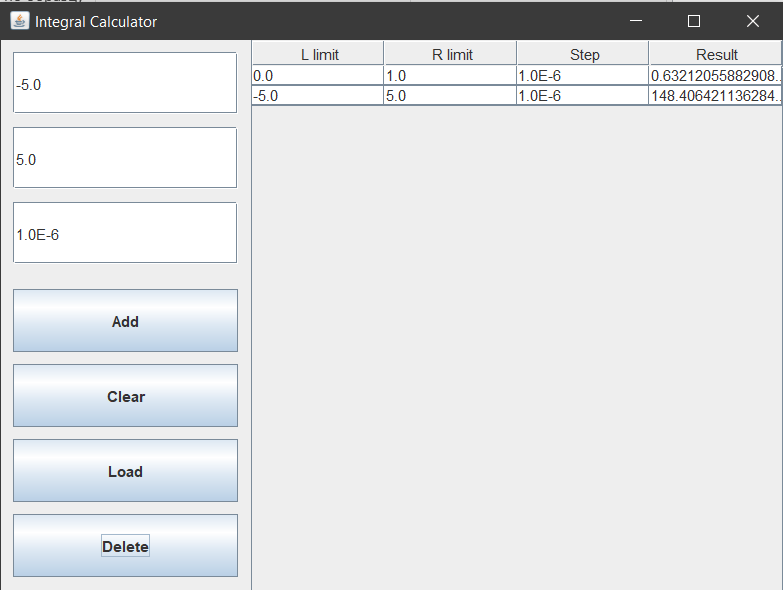


Рис.5 – Состояние программы после загрузки данных в таблицу из коллекции кнопкой Load

Результат программы совпал с ожидаемым, следовательно, программа составлена верно.

**Вывод**

В результате выполнения методических указаний к лабораторной работе были получены навыки работы с графической оболочкой Java swing и коллекциями.