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»

Выполнили:

студенты группы 20ВВП1

Войнова Д.А.

Зиновьев Я.М.

Проверили:

Юрова О.В.

Карамышева Н.С.

Пенза 2023

**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.

**Листинг:**

import javax.swing.\*;

import javax.swing.event.TableModelEvent;

import javax.swing.event.TableModelListener;

import javax.swing.table.DefaultTableModel;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

import java.util.Vector;

public class App extends JFrame {

private JTextField up;

private JTextField step;

private JTextField down;

private JButton deleteButton;

private JButton addButton;

private JButton calculateButton;

private JPanel rootPanel;

private JTable table;

private JButton deleteTableButton;

private JButton restoreButton;

private DefaultTableModel model;

private List<RecIntegral> data = new ArrayList<>();

public App(){

setVisible(true);

setSize(800,600);

setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);

setContentPane(rootPanel);

model = (DefaultTableModel) table.getModel();

model.addColumn("up");

model.addColumn("down");

model.addColumn("step");

model.addColumn("result");

addButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String up = App.this.up.getText();

String down = App.this.down.getText();

String step = App.this.step.getText();

model.addRow(new String[]{up,down,step,"0"});

App.this.up.setText("");

App.this.down.setText("");

App.this.step.setText("");

data.add(new RecIntegral(Arrays.*stream*(new String[]{up,down,step,"0"}).toList()));

}

});

deleteButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

int temp = table.getSelectedRow();

if (temp != -1){

model.removeRow(temp);

data.remove(temp);

}

}

});

calculateButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

int temp = table.getSelectedRow();

if (temp == -1) {

return;

}

Vector localData = model.getDataVector().get(temp);

float max = Float.*parseFloat*((String) localData.get(0));

float min = Float.*parseFloat*((String) localData.get(1));

float step = Float.*parseFloat*((String) localData.get(2));

double result = 0;

for (float i = min; i < max - step; i += step) {

if (i > max) {

result += (Math.*cos*(i - step) + Math.*cos*(max)) / 2 \* step;

} else {

result += (Math.*cos*(i) + Math.*cos*(i + step)) / 2 \* step;

}

}

model.setValueAt(result, temp, 3);

data.get(temp).setDataByIndex(3, String.*valueOf*(result));

}

});

restoreButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

int temp= table.getRowCount();

for (int i= 0; i < temp;i++){

model.removeRow(0);

}

for (RecIntegral element: data) {

model.addRow(element.getRecord().toArray());

}

}

});

deleteTableButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

int temp= table.getRowCount();

for (int i= 0; i < temp;i++){

model.removeRow(0);

}

}

});

model.addTableModelListener(new TableModelListener() {

@Override

public void tableChanged(TableModelEvent e) {

if (e.getType() == TableModelEvent.*UPDATE*) {

data.get(table.getSelectedRow()).setDataByIndex(table.getSelectedColumn(),

(String) model.getDataVector().get(table.getSelectedRow()).get(table.getSelectedColumn()));

}

}

});

}

public static void main (String[] args){

new App();

}

private void createUIComponents() {

table = new JTable(){

@Override

public boolean isCellEditable(int row,int column){

return column != 3;

}

};

}

}

RecIntegral.java

import java.util.ArrayList;

import java.util.List;

public class RecIntegral {

private List<String> record;

public List<String> getRecord() {

return record;

}

public void setRecord(List<String> record) {

this.record = record;

}

public void setDataByIndex(int index, String record) {

this.record.set(index, record);

}

public RecIntegral(List<String> record){

this.record = new ArrayList<>(record);

}

}

Результат работы программы:

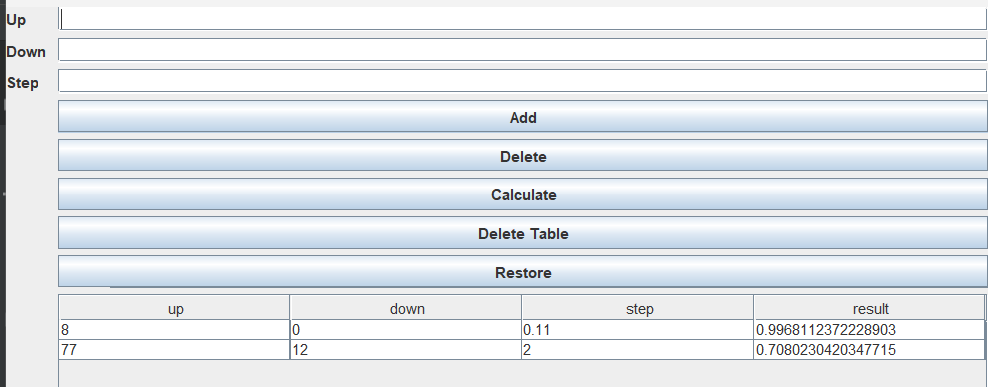


Рис. 1

Вывод: в ходе работы мы изучили библиотеку стандартных коллекций Java Collections Framework, позволяющую хранить различные структуры данных