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

Беляков А. 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.List;  
import java.util.Vector;  
  
public class Application extends JFrame {  
 private static final List<String> *tableHeader* = List.*of*(new String[]{ "step", "min", "max", "result"});  
 private static final List<List<String>> *startData* = List.*of*(List.*of*(new String[]{"0.01", "1", "5"}),  
 List.*of*(new String[]{"0.001", "-2", "5"}));  
 private static final int *NON\_EDITABLE\_COLUMN* = 3;  
 private JTextField stepTextField;  
 private JTextField minTextField;  
 private JTextField maxTextField;  
 private JButton addButton;  
 private JButton deleteButton;  
 private JButton calculateButton;  
 private JTable table;  
 private JPanel rootPanel;  
 private JButton deleteTableButton;  
 private JButton uploadButton;  
 private DefaultTableModel defaultTableModel;  
 private List<RecIntegral> data;  
  
 public Application() {  
 super("Lab\_1");  
 setContentPane(rootPanel);  
 setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);  
 setVisible(true);  
 setSize(800, 600);  
 addButton.addActionListener(new AddButtonActionListener());  
 deleteButton.addActionListener(new DeleteButtonActionListener());  
 calculateButton.addActionListener(new CalculateButtonActionListener());  
 defaultTableModel.addTableModelListener(new ChangeTableListener());  
 deleteTableButton.addActionListener(new DeleteTableListener());  
 uploadButton.addActionListener(new UploadTableListener());  
 }  
  
 public static void main(String[] args) {  
 new Application();  
 }  
  
 private void createUIComponents() {  
 data = new ArrayList<>();  
 table = new JTable() {  
 @Override  
 public boolean isCellEditable(int row, int column) {  
 return column != *NON\_EDITABLE\_COLUMN*;  
 }  
 };  
 defaultTableModel = (DefaultTableModel) table.getModel();  
 *tableHeader*.forEach(defaultTableModel::addColumn);  
 *startData*.forEach(this::addRow);  
  
 }  
  
 private void addRow(List<String> data) {  
 defaultTableModel.addRow(data.toArray());  
 this.data.add(new RecIntegral(data));  
 }  
  
 private class AddButtonActionListener implements ActionListener {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 List<String> row = new ArrayList<>();  
 String regex = "-?\\d\*+(\\.\\d+)?";  
 if ((!stepTextField.getText()  
 .isEmpty() && stepTextField.getText()  
 .matches(regex)) &&  
 (!minTextField.getText()  
 .isEmpty() && minTextField.getText()  
 .matches(regex)) &&  
 (!maxTextField.getText()  
 .isEmpty() && maxTextField.getText()  
 .matches(regex))) {  
 row.add(stepTextField.getText());  
 stepTextField.setText("");  
 row.add(minTextField.getText());  
 minTextField.setText("");  
 row.add(maxTextField.getText());  
 maxTextField.setText("");  
 addRow(row);  
 }  
 }  
 }  
  
 private class DeleteButtonActionListener implements ActionListener {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 int selectedRow = table.getSelectedRow();  
 if (selectedRow != -1) {  
 defaultTableModel.removeRow(selectedRow);  
 data.remove(selectedRow);  
 }  
 }  
 }  
  
 private class CalculateButtonActionListener implements ActionListener {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 int selectedRow = table.getSelectedRow();  
 if (selectedRow == -1) {  
 return;  
 }  
 Vector args = defaultTableModel.getDataVector()  
 .get(selectedRow);  
 double step = Double.*parseDouble*((String) args.get(0));  
 double min = Double.*parseDouble*((String) args.get(1));  
 double max = Double.*parseDouble*((String) args.get(2));  
 double inResult = 0;  
 double i;  
 for (i = min; i <= max - step \* 2; i += step) {  
 inResult += (Math.*sin*(i) + Math.*sin*(i + step)) \* step / 2;  
 }  
 inResult += (Math.*sin*(max) + Math.*sin*(i)) \* (max - i) / 2;  
 defaultTableModel.setValueAt(inResult, selectedRow, *NON\_EDITABLE\_COLUMN*);  
 data.get(selectedRow)  
 .set(*NON\_EDITABLE\_COLUMN*, String.*valueOf*(inResult));  
 }  
 }  
  
 private class ChangeTableListener implements TableModelListener {  
 public void tableChanged(TableModelEvent e) {  
 if (e.getType() == TableModelEvent.*UPDATE*) {  
 data.get(table.getSelectedRow())  
 .set(table.getSelectedColumn(),  
 (String) defaultTableModel.getDataVector()  
 .get(table.getSelectedRow())  
 .get(table.getSelectedColumn()));  
 }  
 }  
 }  
  
 private class DeleteTableListener implements ActionListener {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 int items = defaultTableModel.getRowCount();  
 for (int i = 0; i < items; i++) {  
 defaultTableModel.removeRow(0);  
 }  
 }  
 }  
  
 private class UploadTableListener implements ActionListener {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 int items = defaultTableModel.getRowCount();  
 for (int i = 0; i < items; i++) {  
 defaultTableModel.removeRow(0);  
 }  
 data.forEach(i -> defaultTableModel.addRow(i.getData().toArray()));  
 }  
 }  
}

RecIntegral.java

import java.util.ArrayList;  
import java.util.List;  
  
public class RecIntegral {  
  
 private List<String> data;  
  
 public RecIntegral(List<String> data) {  
 this.data = new ArrayList<>(data);  
 }  
  
 public void set(int index, String data) {  
 this.data.set(index,data);  
 }  
  
 public List<String> getData() {  
 return data;  
 }  
}

Результат работы программы:

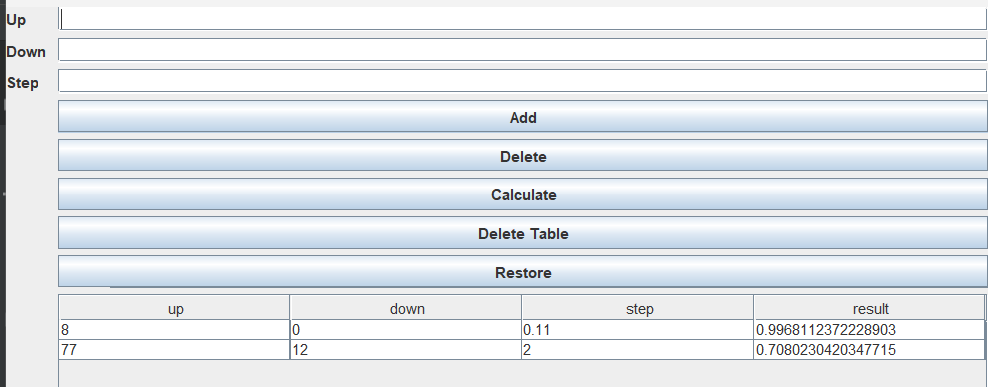


Рис. 1

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