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 20ВВП2:

Горожанин Я. А.

Скирдова В. М.

Приняли:

Юрова О. В.

Карамышева Н. С.

Пенза 2023

**Цель работы:** изучитизучить библиотеку стандартных коллекций Java Collections Framework, позволяющую хранить различные структуры данных.ь библиотеку стандартных коллекций Java Collections Framework, позволяющую хранить различные структуры данных.

**Лабораторное задание:**

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. 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.

**Листинг:**

package com.company;  
  
import javax.swing.\*;  
import javax.swing.table.DefaultTableModel;  
import java.awt.event.\*;  
import java.util.\*;  
  
public class Form extends JDialog {  
  
 private JPanel rootPanel;  
 private JButton buttonOK;  
 private JButton buttonCancel;  
  
 private JButton addButton;  
 private JButton delButton;  
 private JButton calkButton;  
 private JButton clearButton;  
 private JButton fillButton;  
  
 private JTextField input1;  
 private JTextField input2;  
 private JTextField input3;  
 private JTable table1;  
  
 class RecIntegral {  
 String Upper, Lower, Step;  
  
 String getUpper() {  
 return Upper;  
 }  
  
 String getLower() {  
 return Lower;  
 }  
  
 String getStep() {  
 return Step;  
 }  
  
 void setUpper(String Temp) {  
 Upper = Temp;  
 }  
  
 void setLower(String Temp) {  
 Lower = Temp;  
 }  
  
 void setStep(String Temp) {  
 this.Step = Temp;  
 }  
  
 double Calk() {  
 double sum = 0;  
 double limUp = Double.*parseDouble*(Upper);  
 double limDown = Double.*parseDouble*(Lower);  
 double limStep = Double.*parseDouble*(Step);  
 while (limDown + limStep < limUp) {  
 sum += ((Math.*exp*(-limDown) + Math.*exp*(-(limDown + limStep))) / 2) \* limStep;  
 limDown += limStep;  
 }  
 sum += ((Math.*exp*(-limDown) + Math.*exp*(-limUp)) / 2) \* limStep;  
 return sum;  
 }  
 }  
  
 List<RecIntegral> listA = new ArrayList();  
  
 public Form() {  
 setContentPane(rootPanel);  
 setModal(true);  
 getRootPane().setDefaultButton(buttonOK);  
  
 createTable();  
  
 addButton.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent ae) {  
 double limUp, limDown, step;  
 String str\_limUp, str\_limDown, str\_step;  
 str\_limUp = input1.getText();  
 str\_limDown = input2.getText();  
 str\_step = input3.getText();  
  
 try {  
 limUp = Double.*parseDouble*(str\_limUp);  
 limDown = Double.*parseDouble*(str\_limDown);  
 step = Double.*parseDouble*(str\_step);  
 } catch (Exception e) {  
 ShowMsg("Введено некорректное значение");  
 return;  
 }  
  
 DefaultTableModel model = (DefaultTableModel) table1.getModel();  
 model.addRow(new Object[]{model.getRowCount()+1, str\_limUp, str\_limDown, str\_step});  
 RecIntegral temp = new RecIntegral();  
 temp.setUpper(str\_limUp);  
 temp.setLower(str\_limDown);  
 temp.setStep(str\_step);  
 listA.add(temp);  
 input1.setText("");  
 input2.setText("");  
 input3.setText("");  
 UpdateWindow();  
 }  
 });  
  
 delButton.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 int SelectedRow = table1.getSelectedRow();  
 int RowCount = table1.getRowCount();  
  
 if (SelectedRow == -1) {  
 ShowMsg("Не выбрана строка в таблице ");  
 return;  
 }  
 DefaultTableModel model = (DefaultTableModel) table1.getModel();  
 listA.remove(SelectedRow);  
 model.removeRow(SelectedRow);  
 table1.setModel(model);  
 if (SelectedRow == RowCount - 1) {  
 table1.changeSelection(SelectedRow - 1, 0, false, false);  
 } else {  
 table1.changeSelection(SelectedRow, 0, false, false);  
 }  
 UpdateWindow();  
 }  
 });  
  
 calkButton.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 int row = table1.getSelectedRow();  
 DefaultTableModel model = (DefaultTableModel) table1.getModel();  
 RecIntegral temp = listA.get(row);  
 double sum = temp.Calk();  
 model.setValueAt(sum, row, 4);  
 UpdateWindow();  
 }  
 });  
  
 fillButton.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 DefaultTableModel model = (DefaultTableModel) table1.getModel();  
 RecIntegral temp;  
  
 for (RecIntegral recIntegral : listA) {  
 temp = recIntegral;  
 model.addRow(new Object[]{model.getRowCount()+1, recIntegral.getUpper(), recIntegral.getLower(), recIntegral.getStep()});  
 }  
 listA.addAll(listA);  
 UpdateWindow();  
 }  
 });  
  
 clearButton.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 DefaultTableModel model = (DefaultTableModel) table1.getModel();  
 while (model.getRowCount() != 0) {  
 model.removeRow(0);  
 }  
 UpdateWindow();  
 }  
 });  
  
 buttonOK.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 onOK();  
 }  
 });  
  
 buttonCancel.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 onCancel();  
 }  
 });  
  
 *// call onCancel() when cross is clicked* setDefaultCloseOperation(*DO\_NOTHING\_ON\_CLOSE*);  
 addWindowListener(new WindowAdapter() {  
 public void windowClosing(WindowEvent e) {  
 onCancel();  
 }  
 });  
  
 *// call onCancel() on ESCAPE* rootPanel.registerKeyboardAction(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 onCancel();  
 }  
 }, KeyStroke.*getKeyStroke*(KeyEvent.*VK\_ESCAPE*, 0), JComponent.*WHEN\_ANCESTOR\_OF\_FOCUSED\_COMPONENT*);  
 }  
  
 private void onOK() {  
 *// add your code here* dispose();  
 }  
  
 private void onCancel() {  
 *// add your code here if necessary* dispose();  
 }  
  
 private void createTable() {  
  
 table1.setModel(new DefaultTableModel(  
 null,  
 new String[]{  
 "#", "Верхняя граница интегрирования", "Нижняя граница интегрирования",  
 "Шаг интегрирования", "Результат"}  
 ) {  
 @Override  
 public boolean isCellEditable(int row, int column) {  
 return column != 4;  
 }  
 });  
 }  
  
 private void ShowMsg(String s) {  
 this.setVisible(true);  
 JOptionPane.*showMessageDialog*(null, s);  
 this.setVisible(true);  
 }  
  
 private void UpdateWindow() {  
 this.setVisible(true);  
 }  
  
 public static void main(String[] args) {  
 Form dialog = new Form();  
 dialog.pack();  
 dialog.setVisible(true);  
 System.*exit*(0);  
 }  
  
}

**Результат выполнения программы:**

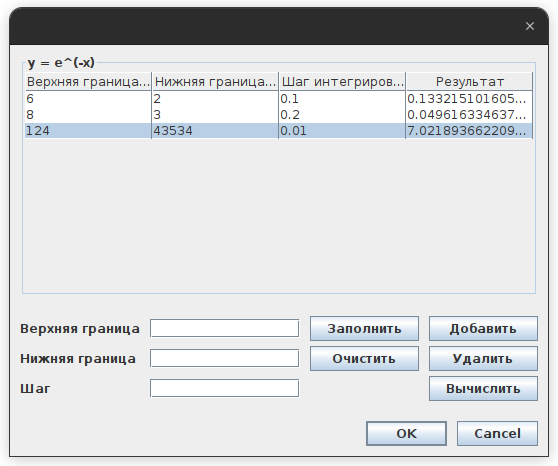


Рисунок 1 — Результаты работы программы до очистки

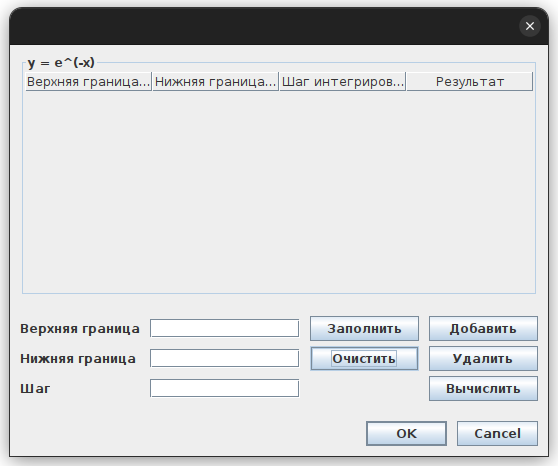


Рисунок 2 — Результаты работы программы после очистки

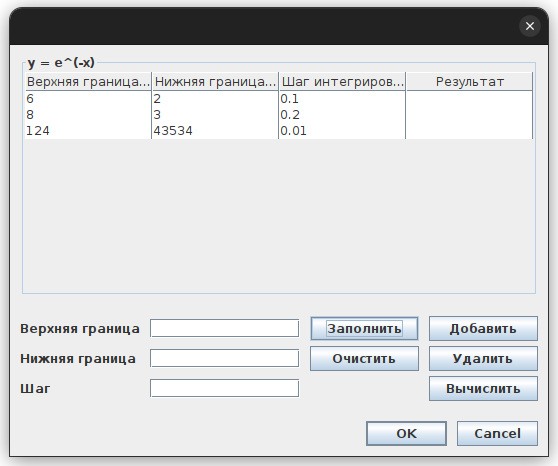


Рисунок 3 — Результаты работы программы после заполнения

**Вывод:** в ходе выполнения данной лабораторной работы была изучена библиотеку стандартных коллекций Java Collections Framework, позволяющая хранить различные структуры данных.