Министерство образования Российской Федерации

Пензенский государственный университет

**Кафедра «Вычислительная техника»**

**ОТЧЁТ**

По лабораторной работе №2

По дисциплине:

«Программирование на языке JAVA»

На тему:

«Работа с коллекциями объектов»

Выполнили:

студенты группы 20ВВП1

Верховский М. В.

Лукин В. Д.

Приняли:

к.т.н., доцент, Юрова О.В.

к.т.н., доцент, 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. В качестве класса-коллекции выбрать LinkedList. Кроме того, добавить пару кнопок: очистить / заполнить, которые будут очищать таблицу и заполнять ее данными из коллекции соответственно.

**Ход выполнения лабораторной работы:**

**Листинг**

Table.java

import javax.swing.\*;  
import javax.swing.table.AbstractTableModel;  
import javax.swing.table.DefaultTableModel;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.util.Arrays;  
import java.util.Iterator;  
import java.util.LinkedList;  
import java.util.Vector;  
  
  
public class Table {  
 private LinkedList<RecIntegral> info = new LinkedList<>();  
 private JButton AddButton;  
 private JButton DelButton;  
 private JButton CalcButton;  
 private JTextField Down;  
 private JTextField Up;  
 private JTextField Step;  
 private JTable MyTable;  
 private DefaultTableModel MyModel;  
 private JPanel MyPanel;  
 private JButton ClearButton;  
 private JButton StuffButton;  
  
 public Table(){  
  
  
 //Убираем возможность редактирования столбца с результатом  
 MyTable.setModel(new DefaultTableModel() {  
  
 @Override  
 public boolean isCellEditable(int rowIndex, int columnIndex) { // Возвращает true, если ячейка в rowIndex и columnIndex доступна для редактирования  
 return columnIndex !=3;  
 }  
 });  
  
 // Чтобы добавить и удалить строки из таблицы, нужно использовать DefaultTableModel  
 MyModel = (DefaultTableModel) MyTable.getModel();  
  
 // создание столбцов  
 MyModel.addColumn("Верх");  
 MyModel.addColumn("Низ");  
 MyModel.addColumn("Шаг");  
 MyModel.addColumn("Результат");  
  
 AddButton.addActionListener(new ActionListener() { // ActionListener хранит в себе метод, который активируется при нажатии кнопки  
 @Override  
 public void actionPerformed(ActionEvent e) { // actionPerformed - метод, который активируется при нажатии кнопки  
 String Up = Table.this.Up.getText(); // getText возвращает текст, содержащийся в текстовом поле Up  
 String Down = Table.this.Down.getText();  
 String Step = Table.this.Step.getText();  
 MyModel.addRow(new String[]{Up, Down, Step}); // addRow добавляет строку  
 info.add(new RecIntegral(Arrays.stream(new String[]{Up, Down, Step, "0"}).toList()));//Работа с коллекцией  
 }  
 });  
  
 CalcButton.addActionListener(new ActionListener() { // ActionListener хранит в себе метод, который активируется при нажатии кнопки  
 @Override  
 public void actionPerformed(ActionEvent e) { // actionPerformed - метод, который активируется при нажатии кнопки  
 int indexRow = MyTable.getSelectedRow(); // Метод getSelectedRow() возвращает индекс первой выбранной строки  
  
 if (indexRow == -1) {  
 return;  
 }  
  
 double area = 0; // площадь = интеграл  
 Vector data = MyModel.getDataVector().get(indexRow);// в data записываем значения выбранной строки (значения могут меняться)  
 double a = Double.parseDouble((String) data.get(0));//получаем значение из 0 столбца выбранной строки в формате double  
 double b = Double.parseDouble((String) data.get(1));//parseDouble возвращает новое double-инициализированное значение, представленное указанным String  
 double h = Double.parseDouble((String) data.get(2));  
  
 for (int i = 0; i < (b - a) / h; i++) {  
 if (a+(i+1)\*h<=b){  
 area += h \* (0.5 \* (Math.tan(a+i\*h) + Math.tan(a+(i+1)\*h)));  
 }  
 else {  
 area += h \* (0.5 \* (Math.tan(a+i\*h) + Math.tan(b)));  
 }  
  
 }  
  
 MyModel.setValueAt(area, indexRow, 3); // установка значения интеграла в строку с индексом в столбце №3  
 info.get(indexRow).setInfo(3, String.valueOf(area));//Работа с коллекцией  
 }  
 });  
  
 DelButton.addActionListener(new ActionListener() { // ActionListener хранит в себе метод, который активируется при нажатии кнопки  
 @Override  
 public void actionPerformed(ActionEvent e) { // actionPerformed - метод, который активируется при нажатии кнопки  
 int indexRow = MyTable.getSelectedRow(); // Метод getSelectedRow() возвращает индекс первой выбранной строки  
 if (indexRow != -1){  
 MyModel.removeRow(indexRow); // Метод removeRow() имеет параметр ind - индекс строки, которая будет удалена  
 info.remove(indexRow);//Работа с коллекцией  
 }  
 }  
 });  
  
  
  
 //Очистить таблицу  
 ClearButton.addActionListener(new ActionListener() { // ActionListener хранит в себе метод, который активируется при нажатии кнопки  
 @Override  
 public void actionPerformed(ActionEvent e) { // actionPerformed - метод, который активируется при нажатии кнопки  
 //Каждый раз, когда удаляем строку, количество строк будет меняться. Продолжаем цикл до тех пор, пока не останется строк  
 while (MyModel.getRowCount()>0)  
 {  
 MyModel.removeRow(0);  
 }  
 }  
 });  
  
 //Заполнить таблицу данными из коллекции  
 StuffButton.addActionListener(new ActionListener() { // ActionListener хранит в себе метод, который активируется при нажатии кнопки  
 @Override  
 public void actionPerformed(ActionEvent e) { // actionPerformed - метод, который активируется при нажатии кнопки  
  
 while (MyModel.getRowCount()>0)  
 {  
 MyModel.removeRow(0);  
 }  
 //Заполнение таблицы данными из коллекции  
 for(RecIntegral data: info){  
 MyModel.addRow(data.getInfo().toArray()); // toArray() возвращает массив, содержащий все элементы коллекции  
 }  
  
 }  
 });  
  
  
 }  
  
  
 public static void main(String[] args) {  
 JFrame frame = new JFrame("Интегральная гипербола кибернетики"); // создаём окно с рамкой  
 frame.setVisible(true); // показывает или скрывает это окно в зависимости от значения параметра  
 frame.setContentPane(new Table().MyPanel);//задаёт содержимое окна с рамкой  
 frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);//Устанавливает операцию при закрытии окна закончить работу приложения,  
 // которая будет выполняться по умолчанию, когда пользователь инициирует «закрытие» этого окна  
 frame.setSize(450,600); // размер окна  
 frame.setLocation(120,100); // расположение окна  
  
  
  
 }  
  
  
}

RecIntegral.java

import java.util.LinkedList;  
import java.util.List;  
  
public class RecIntegral {  
  
private List<String> info;  
  
 public RecIntegral(List<String> info){  
 this.info = new LinkedList<>(info);  
 }  
  
 public List<String> getInfo() {  
 return info;  
 }  
  
 public void setInfo(int index, String info){  
 this.info.set(index, info);  
 }  
  
  
}

**Результаты выполнения программы**

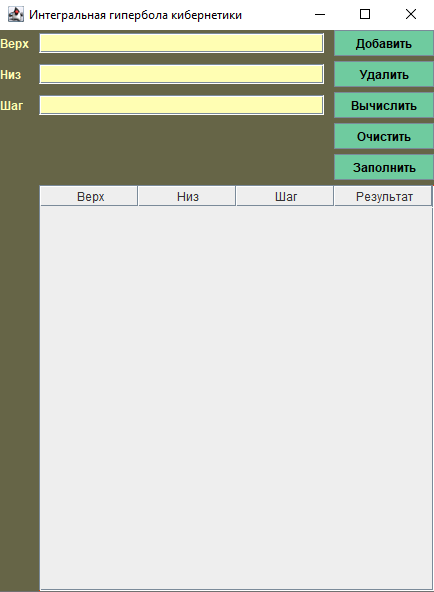


Рисунок №1 - Запуск программы

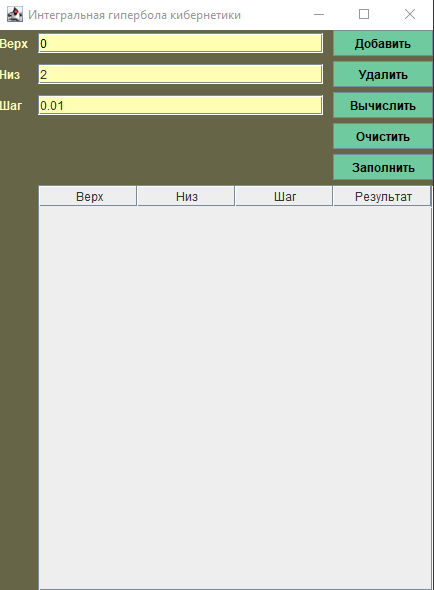


Рисунок №2 - Ввод данных для интеграла

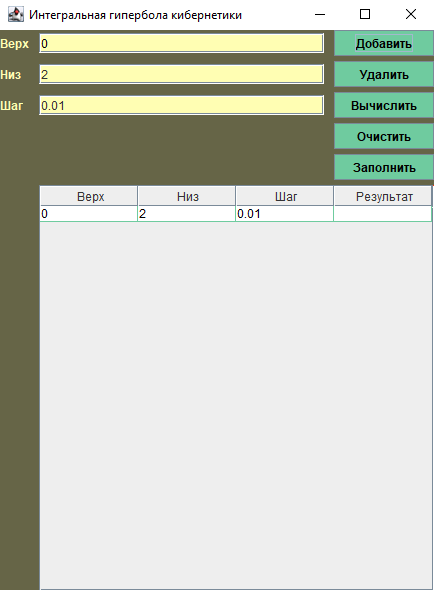


Рисунок №3 - Добавление строки с данными в таблицу и в коллекцию

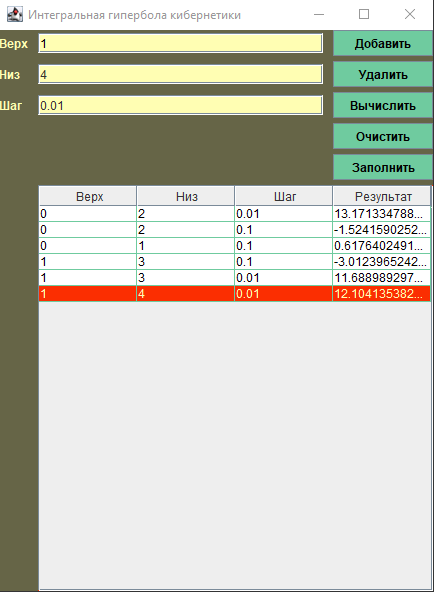


Рисунок №4 - Вычисление определённых интегралов, добавление результатов в коллекцию

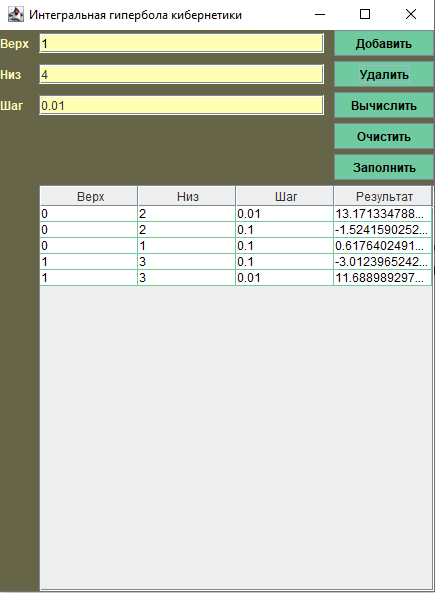


Рисунок №5 - Удаление строки с данными из таблицы и из коллекции

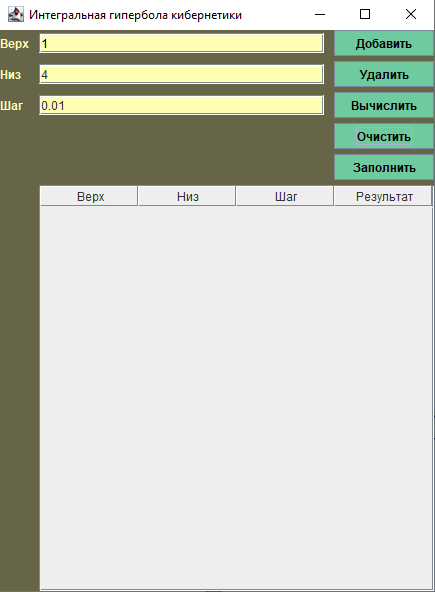


Рисунок №6 - Очищение таблицы

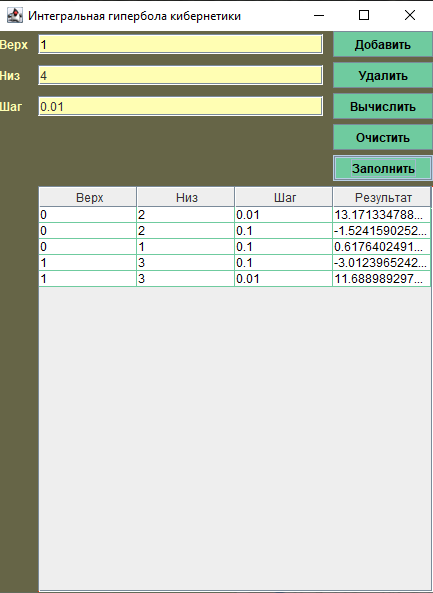


Рисунок №7 - Заполнение таблицы

**Вывод:** Мы изучили библиотеку стандартных коллекций Java Collections Framework, позволяющую хранить различные структуры данных.