МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра «ВТ»

ОТЧЁТ

по лабораторной работе № 5

Оформление документов по управлению проектом

по дисциплине **«Программирование на языке Java»**

Выполнили: ст. гр. 21ВВП1

Назаров Е.А.

Макаров И.С  
 Козлова К.С.

Принял:

Юрова О.В.  
Карамышева Н.С.

2024

**Цель работы:** научиться создавать многопоточные приложения c использованием стандартных средств языка Java.

**Вариант:** №1

**Ход работы:**

Реализовали поток для расчёта интеграла.

class Button3EventListener implements ActionListener {  
 public void actionPerformed(ActionEvent e) {  
 // Создаем и запускаем новый поток для выполнения операции по кнопке 3  
 Thread thread = new Thread(new Button3Thread());  
 thread.start();  
 }

И добавили все ранее содержимое Button3EventListener в новый класс Button3Thread.

**Листинг:  
 main**import javax.swing.\*;  
  
public class Main {  
 public static void main(String[] args) throws UnsupportedLookAndFeelException, ClassNotFoundException, InstantiationException, IllegalAccessException {  
 UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());  
 lab app = new lab();  
 app.setVisible(true);  
 }  
}  
  
 **lab**import javax.swing.\*;  
import javax.swing.table.DefaultTableModel;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.io.Serializable;  
import java.util.ArrayList;  
import java.util.concurrent.TimeUnit;  
  
public class lab extends JFrame {  
 private JTextField input1;  
 private JTextField input2;  
 private JTextField input3;  
 private JTable table;  
 private JButton button1;  
 private JButton button2;  
 private JButton button3;  
 private JButton button4;  
 private JButton button5;  
 private JLabel label1;  
 private JLabel label2;  
 private JLabel label3;  
 private JPanel rootPanel;  
 private JButton saveTxtButton;  
 private JButton readTxtButton;  
 private JButton saveSerButton;  
 private JButton readSerButton;  
 private static DefaultTableModel *model*;  
 private static ArrayList<Recintegral> *contenttable* = new ArrayList<>();  
 public lab() {  
 add(rootPanel);  
 setTitle("Java");  
 setSize(600, 400);  
  
 // Добавляем обработчики событий к кнопкам  
 button1.addActionListener(new Button1EventListener());  
 button2.addActionListener(new Button2EventListener());  
 button3.addActionListener(new Button3EventListener());  
 button4.addActionListener(new Button4EventListener());  
 button5.addActionListener(new Button5EventListener());  
 saveTxtButton.addActionListener(new Button6EventListener());  
 readTxtButton.addActionListener(new Button7EventListener());  
 saveSerButton.addActionListener(new Button8EventListener());  
 readSerButton.addActionListener(new Button9EventListener());  
 }  
  
 public double parseDouble(String input) throws MyException {  
 input = input.replace(',', '.');  
 if (input.matches("-?\\d+(\\.\\d+)?")) {  
 double tmp = Double.*parseDouble*(input);  
 if(tmp > 1000000 || tmp< 0.000001)  
 throw new MyException("Выход за диапазон значений");  
 return tmp;  
 } else {  
 throw new MyException("Неверный формат числа");  
 }  
 }  
 public boolean IsCorrect(double d1, double d2, double d3) throws MyException {  
 if(d1 >= d2 || d3 >= d2-d1 )  
 throw new MyException("Некоректно заполненные данные");  
 return true;  
 }  
  
 private void createUIComponents() {  
 String[] columns = {"Нижняя граница", "Верхняя граница", "Шаг", "Результат"};  
 String[][] data = {};  
  
 *model* = new DefaultTableModel(data, columns);  
 table = new JTable(*model*);  
 }  
  
 class Button1EventListener implements ActionListener {  
 public void actionPerformed(ActionEvent e) {  
 try {  
 double d1=parseDouble(input1.getText() );  
 double d2=parseDouble(input2.getText() );  
 double d3=parseDouble(input3.getText() );  
 if(IsCorrect(d1,d2,d3)) {  
 Recintegral temp = new Recintegral(d1, d2, d3);  
 *model*.addRow(new Object[]{input1.getText(), input2.getText(), input3.getText()});  
 *contenttable*.add(temp);  
 }  
 }  
 catch (MyException ex) {  
 ex.GetFormMessage();  
 }  
 }  
 }  
  
 class Button2EventListener implements ActionListener {  
 public void actionPerformed(ActionEvent e) {  
 int temp=table.getSelectedRow();  
 *model*.removeRow(temp);  
 *contenttable*.remove(temp);  
 }  
 }  
 class Button3EventListener implements ActionListener {  
 public void actionPerformed(ActionEvent e) {  
 // Создаем и запускаем новый поток для выполнения операции по кнопке 3  
 Thread thread = new Thread(new Button3Thread());  
 thread.start();  
 }  
 }  
 // Дополнительный поток для выполнения операции по кнопке 3  
 class Button3Thread implements Runnable {  
 public void run() {  
 int n = table.getSelectedRow();  
 if (n != -1 && n < *model*.getRowCount()) {  
 try {  
 TimeUnit.*SECONDS*.sleep(5);  
 }catch (Exception ex){}  
 double a = Double.*parseDouble*(*model*.getValueAt(n, 0).toString());  
 double b = Double.*parseDouble*(*model*.getValueAt(n, 1).toString());  
 double h = Double.*parseDouble*(*model*.getValueAt(n, 2).toString());  
 String Res = String.*valueOf*(Level1.*Trap*(a, b, h));  
 *model*.setValueAt(Res, table.getSelectedRow(), 3);  
 Recintegral temp = new Recintegral(a, b, h, Res);  
 *contenttable*.set(n, temp);  
 if (*contenttable*.size() > n) {  
 *contenttable*.set(n, temp);  
 } else {  
 *contenttable*.add(temp);  
 }  
 } else {  
 JOptionPane.*showMessageDialog*(null, "Выберите строку для редактирования!");  
 }  
 }  
 }  
 class Button4EventListener implements ActionListener {  
 public void actionPerformed(ActionEvent e) {  
 *model*.setRowCount(0);  
 for (Recintegral con:*contenttable*){  
 if (con.Result==null)  
 *model*.addRow(new Object[]{con.LowBorder,con.HighBorder,con.Step});  
 else  
 *model*.addRow(new Object[]{con.LowBorder,con.HighBorder,con.Step,con.Result});  
 }  
 }  
 }  
  
 class Button5EventListener implements ActionListener {  
 public void actionPerformed(ActionEvent e) {  
 *model*.setRowCount(0);  
 }  
 }  
 class Button6EventListener implements ActionListener {  
 public void actionPerformed(ActionEvent e) {  
 Serializate.*saveToFileTxt*(*contenttable*);  
 }  
 }  
  
 class Button7EventListener implements ActionListener {  
 public void actionPerformed(ActionEvent e) {  
 Serializate.*loadFromFileTxt*(*contenttable*,*model*);  
 }  
 }  
  
 class Button8EventListener implements ActionListener {  
 public void actionPerformed(ActionEvent e) {  
 Serializate.*saveToFileSer*(*contenttable*);  
 }  
 }  
  
 class Button9EventListener implements ActionListener {  
 public void actionPerformed(ActionEvent e) {  
 Serializate.*loadFromFileSer*(*contenttable*,*model*);  
 }  
 }  
  
 public static class Level1 {  
 public static double f(double x) {  
 return 1/x;  
 }  
 public static double Trap(double a, double b, double h){  
 double result = 0;  
 for(double i = a; i < b; i+=h) {  
 if((i+h) < b )  
 result += (*f*(i) + *f*(h+i)) \* h/2;  
 else  
 result += (*f*(i)+*f*(b)) \* (b-i)/2;  
 }  
  
 return result;  
 }  
 };  
 static public class Recintegral implements Serializable {  
 double LowBorder;  
 double HighBorder;  
 double Step;  
 String Result;  
  
 public Recintegral(double LowBorder,double HighBorder, double Step, String Result ) {  
 this.LowBorder=LowBorder;  
 this.HighBorder=HighBorder;  
 this.Step=Step;  
 this.Result=Result;  
 }  
  
 public Recintegral(double LowBorder,double HighBorder, double Step) {  
 this.LowBorder=LowBorder;  
 this.HighBorder=HighBorder;  
 this.Step=Step;  
 this.Result=null;  
 }  
  
 }  
  
}

**MyException**import javax.swing.\*;  
public class MyException extends Exception{  
 private String message;  
 public MyException(String msg){  
 super(msg);  
 this.message = msg;  
 }  
  
 public void GetFormMessage(){  
 JOptionPane.*showMessageDialog*(null, message, "InfoBox", JOptionPane.*INFORMATION\_MESSAGE*);  
 }  
  
}

**Serializate**

import java.io.\*;  
import java.util.ArrayList;  
import javax.swing.\*;  
import javax.swing.table.DefaultTableModel;  
  
public class Serializate {  
  
 static public void saveToFileSer(ArrayList<lab.Recintegral> contenttable) {  
 try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream("data.ser"))) {  
 out.writeObject(contenttable);  
 JOptionPane.*showMessageDialog*(null, "Данные успешно сохранены!");  
 } catch (IOException ex) {  
 JOptionPane.*showMessageDialog*(null, "Ошибка при сохранении данных в файл!");  
 }  
 }  
  
 static public void loadFromFileSer(ArrayList<lab.Recintegral> contenttable, DefaultTableModel model) {  
 try (ObjectInputStream in = new ObjectInputStream(new FileInputStream("data.ser"))) {  
 contenttable = (ArrayList<lab.Recintegral>) in.readObject();  
 model.setRowCount(0);  
 for (lab.Recintegral integral : contenttable) {  
 model.addRow(new Object[]{integral.LowBorder, integral.HighBorder, integral.Step, integral.Result});  
 }  
 JOptionPane.*showMessageDialog*(null, "Данные успешно загружены!");  
 } catch (IOException | ClassNotFoundException ex) {  
 JOptionPane.*showMessageDialog*(null, "Ошибка при загрузке данных из файла!");  
 }  
 }  
  
 static public void saveToFileTxt(ArrayList<lab.Recintegral> contenttable) {  
 try (BufferedWriter writer = new BufferedWriter(new FileWriter("data.txt"))) {  
 for (lab.Recintegral integral : contenttable) {  
 writer.write(integral.LowBorder + "," + integral.HighBorder + "," + integral.Step + "," + integral.Result + "\n");  
 }  
 JOptionPane.*showMessageDialog*(null, "Данные успешно сохранены в файле data.txt!");  
 writer.close();  
 } catch (IOException ex) {  
 JOptionPane.*showMessageDialog*(null, "Ошибка при сохранении данных в файл!");  
 }  
 }  
  
 static public void loadFromFileTxt(ArrayList<lab.Recintegral> contenttable, DefaultTableModel model) {  
 try (BufferedReader reader = new BufferedReader(new FileReader("data.txt"))) {  
 String line;  
 model.setRowCount(0); // Очищаем таблицу перед загрузкой данных  
 while ((line = reader.readLine()) != null) {  
 String[] parts = line.split(",");  
 double lowBorder = Double.*parseDouble*(parts[0]);  
 double highBorder = Double.*parseDouble*(parts[1]);  
 double step = Double.*parseDouble*(parts[2]);  
 String result = parts[3];  
 model.addRow(new Object[]{lowBorder, highBorder, step, result});  
 // lab.Recintegral temp = new lab.Recintegral(lowBorder, highBorder, step, result);  
 // contenttable.add(temp);  
 }  
 JOptionPane.*showMessageDialog*(null, "Данные успешно загружены из файла data.txt!");  
 } catch (IOException | NumberFormatException ex) {  
 JOptionPane.*showMessageDialog*(null, "Ошибка при загрузке данных из файла!");  
 }  
 }  
}

**Вывод.** изучили работу с файлами и механизмы сериализации данных. 