Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №5

по курсу «Программные средства проектирования цифровых устройств»

на тему «Многопоточность в Java»

**Выполнили студенты группы 21ввв1:**

Лобанов Р.И.

Антропов Д.В.

**Приняли**

Юрова О.В.

Карамышева Н.С.

Пенза 2024

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

**Задание на лабораторную работу:** модифицировать приложение из предыдущей лабораторной работы, реализовав вычисление определенного интеграла в нескольких дополнительных потоках (число потоков определяется номером варианта), снимая нагрузку с основного потока и предотвращая "подвисание" графического интерфейса. Варианты с номерами до 5 включительно реализуют многопоточность путем наследования от класса Thread, остальные реализуют интерфейс Runnable. Оформление лабораторной работы должно быть выполнено в соответствии с требованиями, приведенными в Приложении 2.

**Листинг:**

**Класс NewForm.java**

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.\*;

import java.util.HashMap;

import java.util.Map;

public class NewForm {

double lowBorder = 0; //down

double highBorder = 0; //upp

double step = 0; //step

private JPanel panel1;

private JTextField lowBorderTextField;

private JTextField highBorderTextField;

private JTextField stepTextField;

private JButton CalcButton;

private JButton AddButton;

private JButton DelButton;

private JTable table1;

private JScrollPane scrollPane1;

private JButton AddRecToTableButton;

private JButton DeleteRecFromCollectionButton;

private JButton saveFileButton;

private JButton loadFileButton;

private JButton serializeButton;

private JButton deserializeButton;

private HashMap<RecIntegral,Double> collectionHashMap = new HashMap();

private HashMap<RecIntegral,Double> tableHashMap = new HashMap();

DefaultTableModel model = (DefaultTableModel) table1.getModel();

public NewForm(){

CalcButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

for(int row = 0; row < table1.getRowCount(); row++){

if(table1.getValueAt(row,0) != "" && table1.getValueAt(row,1) != "" && table1.getValueAt(row,2) != ""){

step = Double.parseDouble(table1.getValueAt(row, 0).toString());

highBorder = Double.parseDouble(table1.getValueAt(row, 1).toString());

lowBorder = Double.parseDouble(table1.getValueAt(row, 2).toString());

RecIntegral rec = new RecIntegral(lowBorder,highBorder,step);

double answer = integrate(rec);

table1.setValueAt(answer, row, 3);

tableHashMap.put(rec,answer);

collectionHashMap.put(rec,answer);

}

}

}

});

AddButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String step = stepTextField.getText();

String highBorder = highBorderTextField.getText();

String lowBorder = lowBorderTextField.getText();

double lb;

double hb;

double sp;

try{

lb= Double.parseDouble(lowBorder);

hb = Double.parseDouble(highBorder);

sp = Double.parseDouble(step);

RecIntegral rec;

if(lb<0.000001 || lb>1000000 ||hb<0.000001 || hb>1000000||sp<0.000001 || sp>1000000)

throw new MyException(lb,hb,sp);

rec = new RecIntegral(lb,hb,sp);

if(!tableHashMap.containsKey(rec)){

tableHashMap.put(rec,null);

model.addRow(new Object[]{step, highBorder, lowBorder});

}

}

catch (MyException myException){

}

catch (Exception exception){

JOptionPane.showMessageDialog(new JOptionPane(),

"Числа в полях должны быть представленны в виде действительных чисел( 0 , 1 , 1.0 и т.д)",

"MyException",

JOptionPane.ERROR\_MESSAGE);

}

}

});

DelButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

int selectedRow = table1.getSelectedRow();

if (selectedRow != -1){

tableHashMap.remove(

new RecIntegral(

Double.parseDouble(table1.getValueAt(selectedRow,2).toString()),

Double.parseDouble(table1.getValueAt(selectedRow,1).toString()),

Double.parseDouble(table1.getValueAt(selectedRow,0).toString())

)

);

model.removeRow(selectedRow);

}

}

});

AddRecToTableButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

addToModelComparedTables(tableHashMap,collectionHashMap,model);

}

});

DeleteRecFromCollectionButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

collectionHashMap.clear();

}

});

saveFileButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

FileManager.save(collectionHashMap);

}

});

loadFileButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

HashMap<RecIntegral,Double> loadHashMap = FileManager.load();

addToModelComparedTables(tableHashMap,loadHashMap,model);

}

});

serializeButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

ObjectOutputStream out = null;

try {

out = new ObjectOutputStream(new BufferedOutputStream(

new FileOutputStream("Table.ser")));

out.writeObject(tableHashMap);

out.close();

} catch ( IOException ex ) {

ex.printStackTrace();

}

}

});

deserializeButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

ObjectInputStream in = null;

HashMap<RecIntegral,Double> loadHashMap = null;

try {

in = new ObjectInputStream(new BufferedInputStream(

new FileInputStream("Table.ser")));

loadHashMap = (HashMap<RecIntegral, Double>) in.readObject();

addToModelComparedTables(tableHashMap,loadHashMap,model);

in.close();

} catch (IOException | ClassNotFoundException ex ) {

ex.printStackTrace();

}

}

});

}

public static void main(String[] args) {

JFrame frame = new JFrame("Integrator √x");

frame.setLocationRelativeTo(null);

frame.setContentPane(new NewForm().panel1);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.pack();

frame.setVisible(true);

}

public void addToModelComparedTables(HashMap<RecIntegral,Double> tableHashMap, HashMap<RecIntegral,

Double> addedHashMap,

DefaultTableModel model){

for (Map.Entry<RecIntegral,Double> entry:addedHashMap.entrySet()){

RecIntegral rec = entry.getKey();

if(!tableHashMap.containsKey(rec)){

tableHashMap.put(rec,entry.getValue());

model.addRow(new Object[]{Double.toString(rec.step),

Double.toString(rec.highBorder),

Double.toString(rec.lowBorder),

Double.toString(entry.getValue())});

}

else{

if(tableHashMap.get(rec)==null){

for(int i = 0; i<model.getRowCount() ;i++){

if(Double.parseDouble(model.getValueAt(i,0).toString()) != rec.step)

continue;

if(Double.parseDouble(model.getValueAt(i,1).toString()) != rec.highBorder)

continue;

if(Double.parseDouble(model.getValueAt(i,2).toString()) != rec.lowBorder)

continue;

model.removeRow(i);

tableHashMap.put(rec,entry.getValue());

model.addRow(new Object[]{Double.toString(rec.step),

Double.toString(rec.highBorder),

Double.toString(rec.lowBorder),

Double.toString(entry.getValue())});

break;

}

}

}

}

}

public static double integrate(RecIntegral rec){

Integrator integrator = new Integrator();

return integrator.integrate(rec.lowBorder, rec.highBorder, rec.step);

}

private void createUIComponents() {

scrollPane1 = new JScrollPane(table1);

String[] columnNames = {"Шаг", "Верхний", "Нижний", "Результат"};

String[][] data = {

};

table1 = new JTable(data, columnNames);

table1.setModel(new DefaultTableModel(data, columnNames)

{public boolean isCellEditable(int row, int column){

return column != 3;

}}

);

}

}

**Class Integrator**

public class Integrator extends Thread{

static Double answer = (double)0;

private int threadCounts;

double lowBorder;

double highBorder;

double step;

Integrator(){

answer = (double)0;

threadCounts = 5;

}

Integrator(double lowBorder,double highBorder,double step){

this.lowBorder = lowBorder;

this.highBorder = highBorder;

this.step = step;

}

@Override

public void run() {

double square;

double sum = 0;

double x;

sum += Math.exp(-lowBorder);

for (x = lowBorder+step;x < highBorder;x+=step){

if (x >= highBorder){

x-=step;

break;

}

sum += 2\*Math.exp(-x);

}

square = step\*sum/2;

square += (highBorder-x)\*Math.exp(-highBorder)/2;

synchronized (answer){

answer+=square;

}

}

public double integrate(double lowBorder,double highBorder,double step){

this.lowBorder = lowBorder;

this.highBorder = highBorder;

this.step = step;

double threadStep = (highBorder-lowBorder)/threadCounts;

Integrator []threads = new Integrator[threadCounts];

threads[0] = new Integrator(lowBorder,lowBorder+threadStep,step);

threads[0].start();

for(int i = 1; i< threadCounts;i++){

threads[i] = new Integrator(threads[i-1].highBorder,threads[i-1].highBorder+threadStep,step);

threads[i].start();

}

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

try {

threads[i].join();

}

catch (InterruptedException e){

System.out.println(e.getMessage());

}

}

return answer;

}

}

**Class RecIntegral:**

import java.io.Serializable;

import java.util.Objects;

public class RecIntegral implements Serializable {

double lowBorder;

double highBorder;

double step;

RecIntegral(){

}

RecIntegral(double lowBorder, double highBorder,double step){

this.lowBorder = lowBorder;

this.highBorder = highBorder;

this.step = step;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (!(o instanceof RecIntegral)) return false;

RecIntegral that = (RecIntegral) o;

return Double.compare(that.lowBorder, lowBorder) == 0 && Double.compare(that.highBorder, highBorder) == 0 && Double.compare(that.step, step) == 0;

}

@Override

public int hashCode() {

return Objects.hash(lowBorder, highBorder, step);

}

@Override

public String toString() {

return "RecIntegral{" +

"lowBorder=" + lowBorder +

", highBorder=" + highBorder +

", step=" + step +

'}';

}

}

**Вывод** научились создавать многопоточные приложения c использованием стандартных средств языка Java.