Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

ОТЧЁТ

По лабораторной работе №5

По курсу «Программирование на языке JAVA»

на тему: «Многопоточность в Java»

Выполнили:

студенты группы 20ВВП1:

Сурков Максим

Пантелеев Иван

Принял:

Юрова О.В.

Карамышева Н.С.

Пенза 2023

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

**Ход работы:**

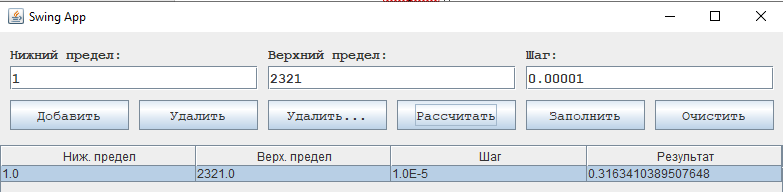
Модифицировали приложение из предыдущей лабораторной работы, реализовав вычисление определенного интеграла в восьми дополнительных потоках, для этого изменили функцию подсчета значение 4 поля таблицы (integralFunc):

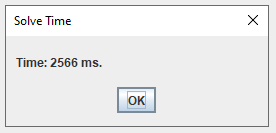
private static double areaFunc(double a, double b,double h){  
 return ((Math.*sin*(a\*a)+Math.*sin*(b\*b))/2)\*h;  
}  
private double fff(double a, double b,double c)  
{  
 double res =0;  
 while(a<b){  
 res+= *areaFunc*(a,a+c,c);  
 a+=c;  
 }  
 return res;  
}  
private void integralFunc(){  
  
 double a = this.downLimit;  
 double b = this.upLimit;  
 double c = this.step;  
 int THREADCOUNT = 8;  
 double ost = (b-a)%c;  
 a+=ost;  
 int steps = (int)((b-a)/c);  
 final double[] result = {0};  
 int cut = steps/THREADCOUNT;  
 int cutOst = steps%THREADCOUNT;  
  
 if(steps<THREADCOUNT) {  
 THREADCOUNT = steps;  
 cut = 1;  
 cutOst = 0;  
 }  
 Thread[] threadArr = new Thread[THREADCOUNT];  
  
 for (int i = 0;i<THREADCOUNT;i++){  
 int finalI = i;  
 double finalCutOst = cutOst;  
 double finalA = a+c\*finalCutOst;  
 int finalCut = cut;  
 Runnable solve = new Runnable() {  
 @Override  
 public void run() {  
 double localResult = 0;  
 if(finalI==0){  
 if(ost>0) {  
 localResult += fff(finalA - ost, finalA, ost);  
 }  
 localResult += fff(finalA-c\*finalCutOst, finalA, c);  
 }  
 localResult += fff(finalA+(c\*finalCut\*finalI),finalA+(c\*finalCut\*(finalI+1)),c);  
 synchronized(this) {  
 result[0] += localResult;  
 }  
 }  
 };  
 threadArr[i]=new Thread(solve);  
 threadArr[i].start();  
 }  
 for (Thread it : threadArr) {  
 try {  
 it.join();  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 this.result = Double.*toString*(result[0]);  
}

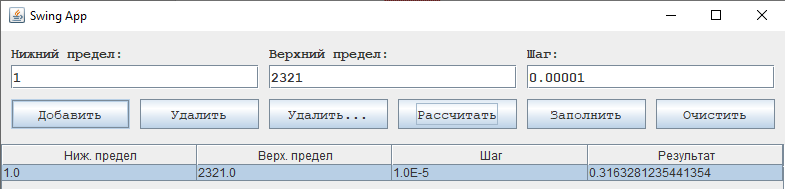
Так же, для достоверности работы многопоточности выполнили проверку времени исполнения метода integralFunc(), используя метод currentTimeMillis() в программах с использованием множественных потоков и без:

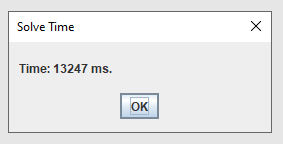
long time = System.*currentTimeMillis*();  
dataList.get(curIndex).integralFunc();  
System.*out*.println(System.*currentTimeMillis*() - time);

Результаты времени подсчета методов:







****

Из этого можно сделать вывод, что многопоточность работает и улучшает производительность программы в ≈ 5 раз.

**Листинг:**

import javax.swing.\*;  
import javax.swing.filechooser.FileFilter;  
import javax.swing.table.DefaultTableModel;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.io.\*;  
import java.util.LinkedList;  
import java.util.Scanner;  
  
  
public class LabApp extends JFrame  
{  
 private final JTextField upLimitField;  
 private final JTextField stepField;  
 private final JTextField downLimitField;  
 private final DefaultTableModel tableModel;  
 private final JTable table1;  
 LinkedList<RecordIntegral>dataList;  
  
  
 static class SimpleException extends Exception{  
 public SimpleException(String msg){  
 JOptionPane.*showMessageDialog*(null,msg,"Exception",JOptionPane.*PLAIN\_MESSAGE*);  
 }  
 }  
 static class RecordIntegral implements Serializable{  
 private double upLimit;  
 private double downLimit;  
 private double step;  
 private String result;  
  
 public RecordIntegral(double downLimit,double upLimit,double step) throws SimpleException {  
 if(upLimit<0.000001||upLimit>1000000||downLimit<0.000001||downLimit>1000000||step<0.000001||step>1000000) throw new SimpleException("Введены некорректные данные");  
 this.upLimit=upLimit;  
 this.downLimit=downLimit;  
 this.step = step;  
 result = "";  
 }  
 public double getUpLimit() {  
 return upLimit;  
 }  
 public double getDownLimit() {  
 return downLimit;  
 }  
 public double getStep() {  
 return step;  
 }  
 public String getResult() {  
 return result;  
 }  
  
 public void setResult(String result) {  
 this.result = result;  
 }  
 public String toString(){  
 return downLimit+" "+upLimit+" "+step+" "+result+"\n";  
 }  
  
 private static double areaFunc(double a, double b,double h){  
 return ((Math.*sin*(a\*a)+Math.*sin*(b\*b))/2)\*h;  
 }  
 private double fff(double a, double b,double c)  
 {  
 double res =0;  
 while(a<b){  
 res+= *areaFunc*(a,a+c,c);  
 a+=c;  
 }  
 return res;  
 }  
 private void integralFunc(){  
  
 double a = this.downLimit;  
 double b = this.upLimit;  
 double c = this.step;  
  
 double ost = (b-a)%c;  
 a+=ost;  
 int steps = (int)((b-a)/c);  
 final double[] result = {0};  
  
 int THREADCOUNT = 8;  
 int cut = steps/THREADCOUNT;  
 int cutOst = steps%THREADCOUNT;  
  
 if(steps<THREADCOUNT) {  
 THREADCOUNT = steps;  
 cut = 1;  
 cutOst = 0;  
 }  
 Thread[] threadArr = new Thread[THREADCOUNT];  
 double finalCutOst = cutOst;  
 double finalA = a+c\*finalCutOst;  
 int finalCut = cut;  
 for (int i = 0;i<THREADCOUNT;i++){  
 int finalI = i;  
 Runnable solve = new Runnable() {  
 @Override  
 public void run() {  
 double localResult = 0;  
 if(finalI==0){  
 if(ost>0) {  
 localResult += fff(finalA - ost, finalA, ost);  
 }  
 localResult += fff(finalA-c\*finalCutOst, finalA, c);  
 }  
 localResult += fff(finalA+(c\*finalCut\*finalI),finalA+(c\*finalCut\*(finalI+1)),c);  
 synchronized(this) {  
 result[0] += localResult;  
 }  
 }  
 };  
 threadArr[i]=new Thread(solve);  
 threadArr[i].start();  
 }  
 for (Thread it : threadArr) {  
 try {  
 it.join();  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 this.result = Double.*toString*(result[0]);  
 }  
  
  
  
 public void notMultiIntegralFunc(){  
 double a = this.downLimit;  
 double b = this.upLimit;  
 double c = this.step;  
  
 double res = 0 ;  
 double ost = (b-a)%c;  
 res+=*areaFunc*(a,a+ost,c);  
 a+=ost;  
 while(a<b){  
 res+= *areaFunc*(a,a+c,c);  
 a+=c;  
 }  
 this.result = Double.*toString*(res);  
 }  
  
 }  
  
 public LabApp()  
 {  
 super("Swing App");  
 setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);  
  
 tableModel = new DefaultTableModel(){  
 @Override  
 public boolean isCellEditable(int i, int i1) {  
 return i1 != 3;  
 }  
 };  
 Object[] columnsHeader = new String[]{"Ниж. предел", "Верх. предел", "Шаг", "Результат"};  
 tableModel.setColumnIdentifiers(columnsHeader);  
 upLimitField = new JTextField(10);  
 downLimitField = new JTextField(10);  
 stepField = new JTextField(10);  
 JLabel label1 = new JLabel("Нижний предел:");  
 JLabel label2 = new JLabel("Верхний предел:");  
 JLabel label3 = new JLabel("Шаг:");  
 table1 = new JTable(tableModel);  
 dataList = new LinkedList<>();  
  
 JButton add = new JButton("Добавить");  
 add.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 try {  
 double a = Double.*parseDouble*(downLimitField.getText());  
 double b = Double.*parseDouble*(upLimitField.getText());  
 double c = Double.*parseDouble*(stepField.getText());  
 if((b-a)<c)throw new SimpleException("Шаг превышает длинну отрезка интегрирования");  
 dataList.add(new RecordIntegral(a, b, c));  
 tableModel.addRow(new Object [] {a,b,c});  
 }  
 catch (SimpleException ignored){}  
 catch (NumberFormatException ex){  
 JOptionPane.*showMessageDialog*(null,"Не все поля заполнены","Exception",JOptionPane.*PLAIN\_MESSAGE*);  
 }  
 }  
 });  
  
 JButton remove = new JButton("Удалить");  
 remove.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 try{  
 int index = table1.getSelectedRow();  
 if(index==-1){throw new SimpleException("Не выбрана строка");}  
 int rowCount = tableModel.getRowCount();  
 int curIndex = dataList.size()-rowCount+index;  
 tableModel.removeRow(index);  
 dataList.remove(curIndex);  
 }catch (SimpleException ignored){}  
 }  
 });  
  
 JButton removeAll = new JButton("Удалить все");  
 removeAll.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 try{  
 if(table1.getRowCount()==0){throw new SimpleException("Таблица пуста");}  
 while (tableModel.getRowCount()!=0){  
 tableModel.removeRow(0);  
 }  
 dataList.clear();  
 }catch (SimpleException ignored){}  
 }  
 });  
  
 JButton solve = new JButton("Рассчитать");  
 solve.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e){  
 try {  
 int index = table1.getSelectedRow();  
 if(index==-1){throw new SimpleException("Не выбрана строка");}  
 int rowCount = tableModel.getRowCount();  
 int curIndex = dataList.size() - rowCount + index;  
  
 long time = System.*currentTimeMillis*();  
 dataList.get(curIndex).integralFunc();  
 time = System.*currentTimeMillis*()-time;  
 tableModel.setValueAt(dataList.get(curIndex).getResult(), index, 3);  
 JOptionPane.*showMessageDialog*(null,"Time: "+Long.*toString*(time)+" ms.","Solve Time",JOptionPane.*PLAIN\_MESSAGE*);  
 }  
 catch (SimpleException ignored){}  
 }  
 });  
  
 JButton clear = new JButton("Очистить");  
 clear.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 while (tableModel.getRowCount()!=0){  
 tableModel.removeRow(0);  
 }  
 }  
 });  
  
 JButton fill = new JButton("Заполнить");  
 fill.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 while (tableModel.getRowCount()!=0){  
 tableModel.removeRow(0);  
 }  
 for(RecordIntegral i : dataList){  
 tableModel.addRow(new Object[]{i.getDownLimit(), i.getUpLimit(), i.getStep(),i.getResult()});  
 }  
 }  
 });  
  
 JButton serializeBit = new JButton("Записать в bin");  
 serializeBit.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 JFileChooser fileChooser = new JFileChooser();  
 fileChooser.setCurrentDirectory(new File("./src"));  
 fileChooser.setFileSelectionMode(JFileChooser.*FILES\_ONLY*);  
 fileChooser.setFileFilter(new FileFilter() {  
 @Override  
 public boolean accept(File f){  
 return f.getName().endsWith("bin");  
 }  
  
 @Override  
 public String getDescription() {  
 return "Только используемые";  
 }  
 });  
  
 if(fileChooser.showDialog(getContentPane(),"Выбрать")==0){  
 String fileName = fileChooser.getSelectedFile().getName();  
 try(ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("src/"+fileName))){  
 if(!fileName.endsWith("bin")) throw new SimpleException("Выбран неверный тип файла");  
 oos.writeObject(dataList);  
 }catch(IOException ex){  
 JOptionPane.*showMessageDialog*(null,"Ошибка сериализации","Exception",JOptionPane.*PLAIN\_MESSAGE*);  
 }catch (SimpleException ignored){}  
 }  
 }  
 });  
  
 JButton serializeTxt = new JButton("Записать в txt");  
 serializeTxt.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 JFileChooser fileChooser = new JFileChooser();  
 fileChooser.setCurrentDirectory(new File("./src"));  
 fileChooser.setFileSelectionMode(JFileChooser.*FILES\_ONLY*);  
 fileChooser.setFileFilter(new FileFilter() {  
 @Override  
 public boolean accept(File f){  
 return f.getName().endsWith("txt");  
 }  
  
 @Override  
 public String getDescription() {  
 return "Только используемые";  
 }  
 });  
  
  
  
 if(fileChooser.showDialog(getContentPane(),"Выбрать")==0){  
 String fileName = fileChooser.getSelectedFile().getName();  
 try(FileWriter fw = new FileWriter("src/"+fileName)){  
 if(!fileName.endsWith("txt")) throw new SimpleException("Выбран неверный тип файла");  
 for (RecordIntegral i : dataList){  
 fw.write(i.toString());  
 }  
  
 }catch(IOException ex){  
 JOptionPane.*showMessageDialog*(null,"Ошибка сериализации","Exception",JOptionPane.*PLAIN\_MESSAGE*);  
 }catch (SimpleException ignored){}  
 }  
 }  
 });  
  
  
 JButton deserializeBit = new JButton("Считать из bin файла");  
 deserializeBit.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 JFileChooser fileChooser = new JFileChooser();  
 fileChooser.setCurrentDirectory(new File("./src"));  
 fileChooser.setFileSelectionMode(JFileChooser.*FILES\_ONLY*);  
 fileChooser.setFileFilter(new FileFilter() {  
 @Override  
 public boolean accept(File f){  
 return f.getName().endsWith("bin");  
 }  
  
 @Override  
 public String getDescription() {  
 return "Только используемые";  
 }  
 });  
  
 if(fileChooser.showDialog(getContentPane(),"Выбрать")==0){  
 String fileName = fileChooser.getSelectedFile().getName();  
 try(ObjectInputStream ois = new ObjectInputStream(new FileInputStream("src/"+fileName))){  
 if(!fileName.endsWith("bin")) throw new SimpleException("Выбран неверный тип файла");  
 while (tableModel.getRowCount()!=0){  
 tableModel.removeRow(0);  
 }  
 dataList.clear();  
 dataList = (LinkedList<RecordIntegral>) ois.readObject();  
  
 for(RecordIntegral i : dataList){  
 tableModel.addRow(new Object[]{i.getDownLimit(), i.getUpLimit(), i.getStep(),i.getResult()});  
 }  
 }catch(IOException ex){  
 JOptionPane.*showMessageDialog*(null,"Ошибка десериализации","Exception",JOptionPane.*PLAIN\_MESSAGE*);  
 }  
 catch(ClassNotFoundException ex){  
 JOptionPane.*showMessageDialog*(null,"В программе отсутствует соответствующий класс","Exception",JOptionPane.*PLAIN\_MESSAGE*);  
 }  
 catch(SimpleException ignored){}  
  
 }  
  
 }  
 });  
  
 JButton deserializeTxt = new JButton("Считать из txt файла");  
 deserializeTxt.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 JFileChooser fileChooser = new JFileChooser();  
 fileChooser.setCurrentDirectory(new File("./src"));  
 fileChooser.setFileSelectionMode(JFileChooser.*FILES\_ONLY*);  
 fileChooser.setFileFilter(new FileFilter() {  
 @Override  
 public boolean accept(File f){  
 return f.getName().endsWith("txt");  
 }  
  
 @Override  
 public String getDescription() {  
 return "Только используемые";  
 }  
 });  
  
 if(fileChooser.showDialog(getContentPane(),"Выбрать")==0){  
 String fileName = fileChooser.getSelectedFile().getName();  
 try(Scanner fr = new Scanner(new FileReader("src/"+fileName))){  
 if(!fileName.endsWith("txt")) throw new SimpleException("Выбран неверный тип файла");  
 while (tableModel.getRowCount()!=0){  
 tableModel.removeRow(0);  
 }  
 dataList.clear();  
  
 while(fr.hasNextLine()){  
 String line = fr.nextLine();  
 String[]values = line.split(" ");  
 dataList.add(new RecordIntegral(Double.*parseDouble*(values[0]),Double.*parseDouble*(values[1]),Double.*parseDouble*(values[2])));  
 if(values.length>3)dataList.getLast().setResult(values[3]);  
 }  
 for(RecordIntegral i : dataList){  
 tableModel.addRow(new Object[]{i.getDownLimit(), i.getUpLimit(), i.getStep(),i.getResult()});  
 }  
 }catch(IOException ex){  
 JOptionPane.*showMessageDialog*(null,"Ошибка десериализации","Exception",JOptionPane.*PLAIN\_MESSAGE*);  
 }  
 catch(SimpleException ignored){}  
  
 }  
  
 }  
 });  
  
*// Расположение элементов* JPanel mainPanel = new JPanel();  
 JPanel upPanel = new JPanel();  
 JPanel centerPanel = new JPanel();  
 JPanel downPanel = new JPanel();  
 mainPanel.setLayout(new BorderLayout());  
 centerPanel.setLayout(new BorderLayout());  
 upPanel.setLayout(new BoxLayout(upPanel,BoxLayout.*Y\_AXIS*));  
  
 JPanel upButtons = new JPanel();  
 JPanel fields = new JPanel();  
  
 upButtons.setLayout(new GridLayout(1,6,10,0));  
 upButtons.add(add);  
 upButtons.add(remove);  
 upButtons.add(removeAll);  
 upButtons.add(solve);  
 upButtons.add(fill);  
 upButtons.add(clear);  
 upButtons.setBorder(BorderFactory.*createEmptyBorder*(10, 0, 5, 0));  
  
  
 upPanel.add(fields);  
 upPanel.add(upButtons);  
 upPanel.setBorder(BorderFactory.*createEmptyBorder*(10, 10, 10, 10));  
  
 fields.setLayout(new GridLayout(2,3,10,0));  
 fields.add(label1);  
 fields.add(label2);  
 fields.add(label3);  
 fields.add(downLimitField);  
 fields.add(upLimitField);  
 fields.add(stepField);  
  
 centerPanel.add(new JScrollPane(table1),BorderLayout.*CENTER*);  
  
 downPanel.add(serializeBit);  
 downPanel.add(serializeTxt);  
 downPanel.add(deserializeBit);  
 downPanel.add(deserializeTxt);  
 downPanel.setBorder(BorderFactory.*createEmptyBorder*(5, 0, 5, 0));  
  
*// Изменение шрифта* {  
 label1.setFont(new Font(Font.*MONOSPACED*, Font.*BOLD*, 14));  
 label2.setFont(new Font(Font.*MONOSPACED*, Font.*BOLD*, 14));  
 label3.setFont(new Font(Font.*MONOSPACED*, Font.*BOLD*, 14));  
 solve.setFont(new Font(Font.*MONOSPACED*, Font.*PLAIN*, 14));  
 add.setFont(new Font(Font.*MONOSPACED*, Font.*PLAIN*, 14));  
 remove.setFont(new Font(Font.*MONOSPACED*, Font.*PLAIN*, 14));  
 removeAll.setFont(new Font(Font.*MONOSPACED*, Font.*PLAIN*, 14));  
 clear.setFont(new Font(Font.*MONOSPACED*, Font.*PLAIN*, 14));  
 fill.setFont(new Font(Font.*MONOSPACED*, Font.*PLAIN*, 14));  
 serializeBit.setFont(new Font(Font.*MONOSPACED*, Font.*PLAIN*, 14));  
 serializeTxt.setFont(new Font(Font.*MONOSPACED*, Font.*PLAIN*, 14));  
 deserializeBit.setFont(new Font(Font.*MONOSPACED*, Font.*PLAIN*, 14));  
 deserializeTxt.setFont(new Font(Font.*MONOSPACED*, Font.*PLAIN*, 14));  
 upLimitField.setFont(new Font(Font.*MONOSPACED*, Font.*BOLD*, 14));  
 downLimitField.setFont(new Font(Font.*MONOSPACED*, Font.*BOLD*, 14));  
 stepField.setFont(new Font(Font.*MONOSPACED*, Font.*BOLD*, 14));  
 }  
  
 mainPanel.add(upPanel,BorderLayout.*NORTH*);  
 mainPanel.add(centerPanel,BorderLayout.*CENTER*);  
 mainPanel.add(downPanel,BorderLayout.*SOUTH*);  
 getContentPane().add(mainPanel);  
 setSize(800,500);  
 setLocationRelativeTo(null);  
 setVisible(true);  
 }  
 public static void main(String[] args) {  
 new LabApp();  
 }  
}