Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №5

по курсу «Программирование на языке Java»

на тему «Многопоточность в Java»

Вариант №5

Выполнили студенты группы 20ВВП2:

Горожанин Я. А.

Скирдова В. М.

Приняли:

Юрова О. В.

Карамышева Н. С.

Пенза 2023

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

**Лабораторное задание:**

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:**

package com.company;  
  
import javax.swing.\*;  
import javax.swing.filechooser.FileNameExtensionFilter;  
import javax.swing.table.DefaultTableModel;  
import java.awt.event.\*;  
import java.io.\*;  
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;  
 private JButton saveBButton;  
 private JButton saveTButton;  
 private JButton loadTButton;  
 private JButton loadBButton;  
  
 class MyThread extends Thread {  
 private String Upper;  
 private String Lower;  
 private String Step;  
 private int numb;  
 private RecIntegral temp;  
 private DefaultTableModel model;  
  
 MyThread(){}  
  
 public void setFields(String Upper, String Lower, String Step, int numb, RecIntegral temp, DefaultTableModel model){  
 this.Upper = Upper;  
 this.Lower = Lower;  
 this.Step = Step;  
 this.numb = numb;  
 this.temp = temp;  
 this.model = model;  
 }  
  
 public void run() *//Этот метод будет выполняться в побочном потоке* {  
 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;  
 temp.setResult(Double.*toString*(sum));  
 listA.set(numb, temp);  
 model.setValueAt(sum, numb, 4);  
 }  
 }  
  
 class RecIntegral implements Serializable {  
 String Upper, Lower, Step, Result;  
  
 String getUpper() {  
 return Upper;  
 }  
  
 String getLower() {  
 return Lower;  
 }  
  
 String getStep() {  
 return Step;  
 }  
  
 String getResult() {  
 return Result;  
 }  
  
 void setUpper(String Temp) {  
 this.Upper = Temp;  
 }  
  
 void setLower(String Temp) {  
 this.Lower = Temp;  
 }  
  
 void setStep(String Temp) {  
 this.Step = Temp;  
 }  
  
 void setResult(String Temp) {  
 this.Result = Temp;  
 }  
  
 void setAll(String limUp, String limDown, String step, String result) {  
 this.setUpper(limUp);  
 this.setLower(limDown);  
 this.setStep(step);  
 this.setResult(result);  
 }  
 }  
  
 static class MyException extends Exception {  
 String msg;  
  
 MyException(String code) {  
 msg = code;  
 }  
 }  
  
 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);  
 if (limUp < 0.000001 || limUp > 100000)  
 throw new MyException("Неверное значение верхнего предела");  
  
 else if (limDown < 0.000001 || limDown > 100000)  
 throw new MyException("Неверное значение нижнего предела");  
 else if (limDown > limUp)  
 throw new MyException("Нижний предел должен быть меньше верхнего");  
 else if ((limUp - limDown) < step)  
 throw new MyException("Шаг должен быть меньше интервала интегрирования");  
  
 } catch (MyException e) {  
 ShowMsg(e.msg);  
 return;  
 } catch (Exception e) {  
 ShowMsg("Некорректно введены данные");  
 return;  
 }  
 DefaultTableModel model = (DefaultTableModel) table1.getModel();  
 RecIntegral temp = new RecIntegral();  
 temp.setUpper(str\_limUp);  
 temp.setLower(str\_limDown);  
 temp.setStep(str\_step);  
  
 model.addRow(new Object[]{model.getRowCount() + 1, str\_limUp, str\_limDown, str\_step});  
  
 listA.add(temp);  
 input1.setText("");  
 input2.setText("");  
 input3.setText("");  
 UpdateWindow();  
 }  
 });  
  
 delButton.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 int SelectedRow;  
 try {  
 SelectedRow = table1.getSelectedRow();  
 if (SelectedRow == -1)  
 throw new Exception();  
 } catch (Exception e1) {  
 ShowMsg("Не выбрана строка в таблице ");  
 return;  
 }  
 int RowCount = table1.getRowCount();  
  
 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) {  
 DefaultTableModel model = (DefaultTableModel) table1.getModel();  
 MyThread[] thread = new MyThread[5];  
 for (int i = 0; i < 5; i++) {  
 thread[i] = new MyThread();  
 }  
  
 for (int i = 0; i < model.getRowCount(); i++) {  
 RecIntegral temp = listA.get(i);  
 thread[i % 5].setFields(temp.getUpper(), temp.getLower(), temp.getStep(), i, temp, model);  
 thread[i % 5].start();  
  
 }  
 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(), recIntegral.getResult()});  
 }  
 listA.addAll(listA);  
 UpdateWindow();  
 }  
 });  
  
 clearButton.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 try {  
 if (table1.getRowCount() == 0)  
 throw new Exception();  
 } catch (Exception e1) {  
 ShowMsg("Таблица пуста");  
 return;  
 }  
 DefaultTableModel model = (DefaultTableModel) table1.getModel();  
 while (model.getRowCount() != 0) {  
 model.removeRow(0);  
 }  
 UpdateWindow();  
 }  
 });  
  
 saveTButton.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 JFileChooser fc = new JFileChooser();  
 fc.setDialogTitle("Сохранение в текстовом виде");  
 fc.setFileFilter(new FileNameExtensionFilter("Text Files", "txt"));  
 fc.showSaveDialog(null);  
 File f = fc.getSelectedFile();  
  
 try {  
 DefaultTableModel model = (DefaultTableModel) table1.getModel();  
 int row = model.getRowCount();  
 int col = model.getColumnCount();  
  
 FileWriter fw = new FileWriter(f);  
 for (int i = 0; i < row; i++) {  
 for (int j = 0; j < col; j++) {  
 fw.write(model.getValueAt(i, j).toString());  
 fw.write(" ");  
 }  
 fw.write("\n");  
 }  
 fw.close();  
 ShowMsg("Сохранение прошло успешно");  
 } catch (IOException ioException) {  
 ioException.printStackTrace();  
 ShowMsg("Ошибка при сохранении");  
 }  
 }  
 });  
  
 saveBButton.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 JFileChooser fc = new JFileChooser();  
 fc.setDialogTitle("Сохранение в двоичном виде");  
 fc.setFileFilter(new FileNameExtensionFilter("Binary Files", "bin"));  
 fc.showSaveDialog(null);  
 File f = fc.getSelectedFile();  
 ArrayList<String> arr = new ArrayList<String>();  
 DefaultTableModel model = (DefaultTableModel) table1.getModel();  
 int row = model.getRowCount();  
 int col = model.getColumnCount();  
  
 for (int i = 0; i < row; i++) {  
 for (int j = 0; j < col; j++) {  
 arr.add(model.getValueAt(i, j).toString());  
 }  
 }  
  
 try (ObjectOutputStream oos = new ObjectOutputStream(new BufferedOutputStream(new FileOutputStream(f)))) {  
 oos.writeObject(arr);  
 oos.close();  
 ShowMsg("Сохранение прошло успешно");  
 } catch (IOException ioException) {  
 ioException.printStackTrace();  
 ShowMsg("Ошибка при сохранении");  
 }  
 }  
 });  
  
 loadTButton.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 JFileChooser fc = new JFileChooser();  
 fc.setFileFilter(new FileNameExtensionFilter("Text Files", "txt"));  
 fc.showOpenDialog(null);  
 File f = fc.getSelectedFile();  
 try {  
 DefaultTableModel model = (DefaultTableModel) table1.getModel();  
 FileReader fr = new FileReader(f);  
 BufferedReader reader = new BufferedReader(fr);  
 String line;  
 String[] split;  
 RecIntegral temp = new RecIntegral();  
 listA.clear();  
 while (model.getRowCount() != 0)  
 model.removeRow(0);  
 while ((line = reader.readLine()) != null) {  
 split = line.split(" ");  
 model.addRow(new Object[]{model.getRowCount() + 1, split[1], split[2], split[3], split[4]});  
 temp.setAll(split[1], split[2], split[3], split[4]);  
 listA.add(temp);  
 }  
 reader.close();  
 fr.close();  
 ShowMsg("Загрузка прошла успешно");  
 UpdateWindow();  
 } catch (IOException ioException) {  
 ioException.printStackTrace();  
 ShowMsg("Ошибка при загрузке");  
 }  
 }  
 });  
  
 loadBButton.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 JFileChooser fc = new JFileChooser();  
 fc.setFileFilter(new FileNameExtensionFilter("Binary Files", "bin"));  
 fc.showOpenDialog(null);  
 File f = fc.getSelectedFile();  
 try {  
 DefaultTableModel model = (DefaultTableModel) table1.getModel();  
 ObjectInputStream ois = new ObjectInputStream(new BufferedInputStream(new FileInputStream(f)));  
 ArrayList<String> arr = (ArrayList<String>) ois.readObject();  
 ois.close();  
 listA.clear();  
 while (model.getRowCount() != 0)  
 model.removeRow(0);  
 for (int i = 0; i < arr.size(); i += 5) {  
 RecIntegral recint = new RecIntegral();  
 recint.setAll(arr.get(i + 1), arr.get(i + 2), arr.get(i + 3), arr.get(i + 4));  
 model.addRow(new Object[]{model.getRowCount() + 1, arr.get(i + 1), arr.get(i + 2), arr.get(i + 3), arr.get(i + 4)});  
 listA.add(recint);  
 }  
 ShowMsg("Загрузка прошла успешно");  
 UpdateWindow();  
 } catch (IOException | ClassNotFoundException ioException) {  
 ioException.printStackTrace();  
 ShowMsg("Ошибка при загрузке");  
 }  
 }  
 });  
  
 buttonOK.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 onOK();  
 }  
 });  
  
 buttonCancel.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 onCancel();  
 }  
 });  
  
 setDefaultCloseOperation(*DO\_NOTHING\_ON\_CLOSE*);  
  
 addWindowListener(new WindowAdapter() {  
 public void windowClosing(WindowEvent e) {  
 onCancel();  
 }  
 });  
  
 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() {  
 dispose();  
 }  
  
 private void onCancel() {  
 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:**

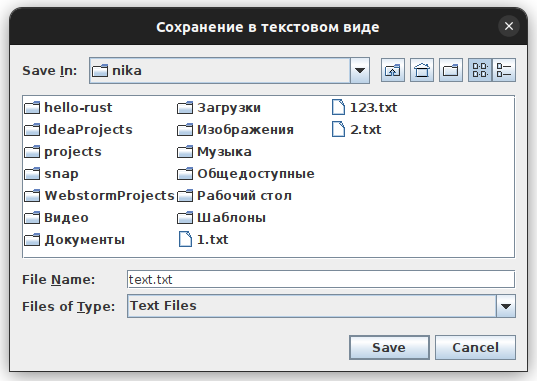


Рисунок 1 — Сохранение в текстовом виде

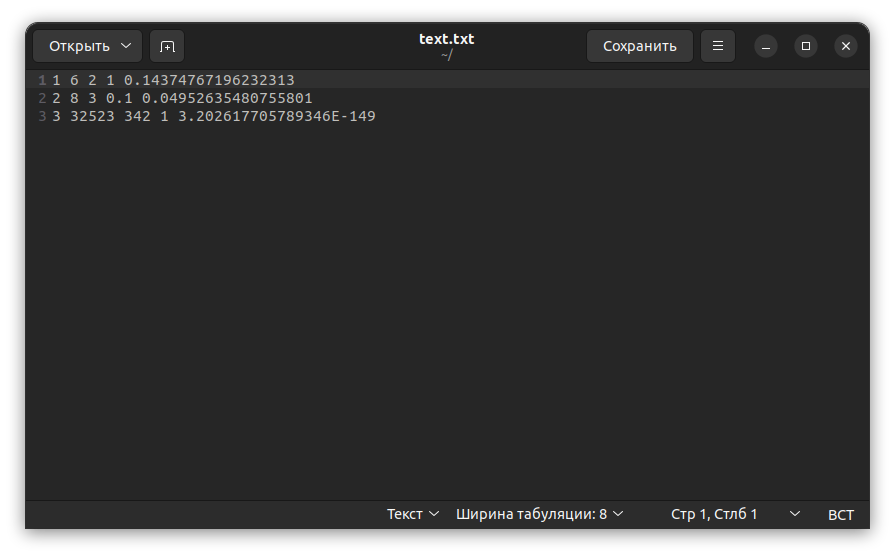


Рисунок 2 — Текстовый документ

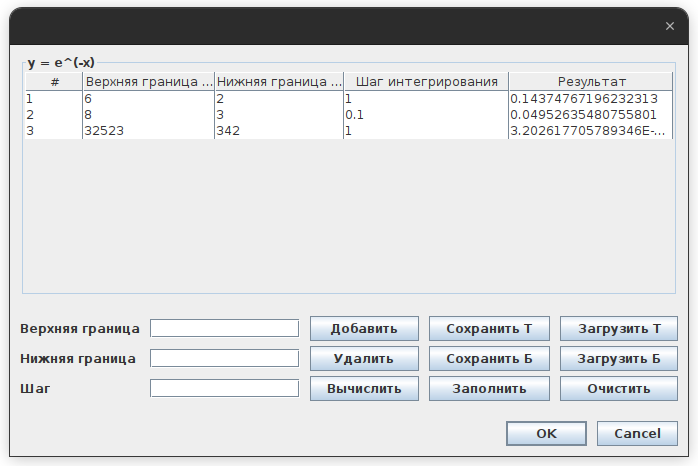


Рисунок 3 — Успешная загрузка из текстового документа

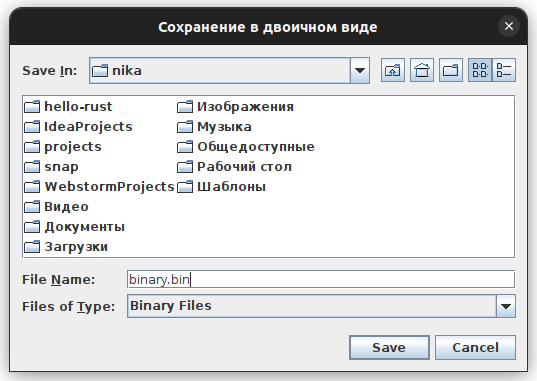


Рисунок 4 — Сохранение в двоичном виде

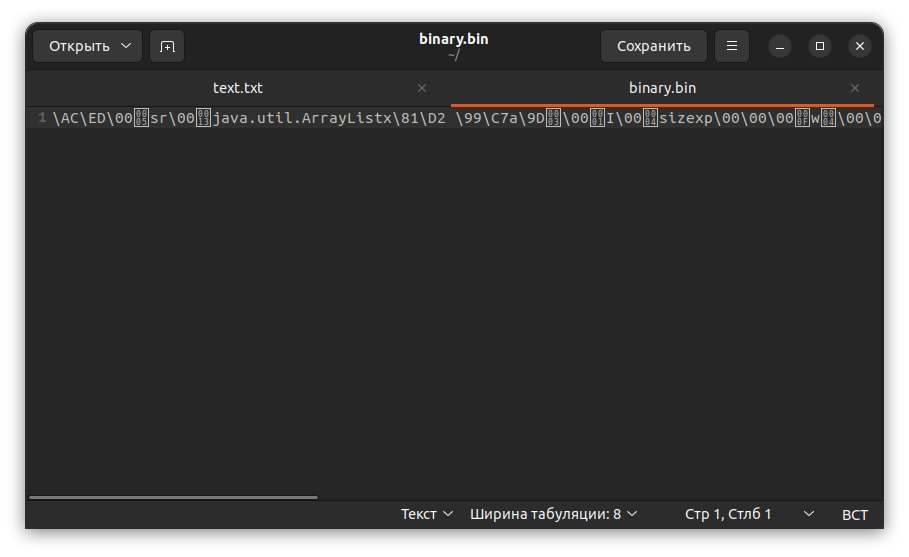


Рисунок 5 — Двоичный файл

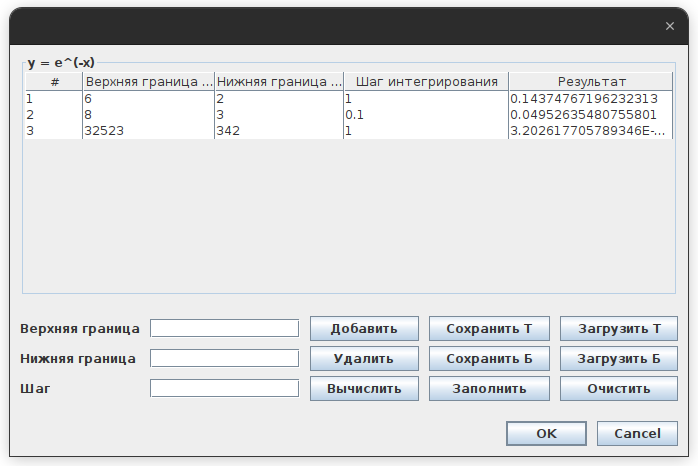


Рисунок 6 — Успешная загрузка из двоичного файла

**Вывод:** в ходе выполнения данной лабораторной работы были получены навыки создания многопоточных приложений c использованием стандартных средств языка Java.