Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЁТ**

По лабораторной работе №6

По дисциплине: «Программирование на языке Java»

На тему: «Сетевое взаимодействие в Java»

Выполнили:

Студент группы 20ВВП1

Кочергин Владислав

Папшев Никита

Принял:

Юрова О.В.

Карамышева Н.С.

***Пенза 2022***

**Цель работы:** научиться создавать клиент-серверные приложения c использованием стандартных классов Java.

**Задание**

|  |  |
| --- | --- |
| Номер варианта | Протокол |
| 9 | UDP |

Модифицировать приложение из предыдущей лабораторной работы, реализовав клиент-серверную архитектуру, обеспечивающую распределенное вычисление определенного интеграла на нескольких вычислительных узлах (клиентах) при этом каждый узел использует несколько нитей, как в предыдущей работе. Сервер не занимается вычислениями, а лишь реализует взаимодействие с пользователем и агрегацию результатов вычислений от клиентов. Нечетные варианты используют протокол UDP, а четные TCP.

**Листинг №1**

**Файл Form.java**

import javax.swing.\*;

import javax.swing.border.TitledBorder;

import javax.swing.plaf.FontUIResource;

import javax.swing.table.DefaultTableModel;

import javax.swing.text.StyleContext;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.\*;

import java.net.\*;

import java.util.ArrayList;

import java.util.Locale;

import static java.lang.Math.cos;

public class Form extends JFrame {

private JPanel MainPanel;

private JTextField textField1;

private JButton enterButton;

private JTextField textField2;

private JTextField textField3;

public JTable table1;

private JButton removeButton;

private JButton calculateButton;

private JButton FillButton;

private JButton ClearButton;

private JButton WriteButton;

private JButton WriteButtonBin;

private JButton OpenButton;

private JButton OpenButtonBin;

private JFileChooser fileChooser;

public ArrayList<RecIntegral> Stroka = new ArrayList<>();

public Form() {

table1.setDefaultEditor(Object.class, null);

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

fileChooser = new JFileChooser("D:\\Work2\\Work\\Java\_Proj\\FirstLab");

MyModel.addColumn("Верх");

MyModel.addColumn("Низ");

MyModel.addColumn("Шаг");

MyModel.addColumn("Результат");

enterButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

double a = Double.parseDouble(textField1.getText());

double b = Double.parseDouble(textField2.getText());

double c = Double.parseDouble(textField3.getText());

try {

if ((a < 0.000001) || (a > 1000000) || (b < 0.000001) || (b > 1000000) || (c < 0.000001) || (c > 1000000)) {

throw new MyException("Присутствует слишком малое либо слишком большое число.\n Укажите пожалуйста число в диапазоне от 0.000001 до 1000000");

}

if (a < b) throw new MyException("Нижний не может быть больше верхнего");

MyModel.addRow(new Object[]{Double.parseDouble(textField1.getText()),

Double.parseDouble(textField2.getText()),

Double.parseDouble(textField3.getText()), 0.0});

} catch (MyException ex) {

JOptionPane.showMessageDialog(MainPanel, ex.getMessage());

}

}

});

removeButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

if (table1.getSelectedRowCount() == 1) {

MyModel.removeRow(table1.getSelectedRow());

} else

JOptionPane.showMessageDialog(calculateButton, "Пожалуйста выберите 1 любую строку");

}

});

calculateButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

try {

DatagramSocket clientSocket = new DatagramSocket();

InetAddress IPAddress = InetAddress.getByName("localhost");

byte[] sendingDataBuffer = new byte[1024];

String message;

DatagramPacket sendingPacket;

byte[] receivingDataBuffer = new byte[1024];

DatagramPacket receivingPacket;

for (int i = 0; i < 9; i++) {

for (int j = 0; j < 3; j++) {

message = new String(MyModel.getValueAt(i, j).toString());

sendingDataBuffer = message.getBytes();

sendingPacket = new DatagramPacket(sendingDataBuffer, sendingDataBuffer.length, IPAddress, 8080);

clientSocket.send(sendingPacket);

}

}

for (int i = 0; i < 9; i++) {

receivingPacket = new DatagramPacket(receivingDataBuffer, receivingDataBuffer.length);

clientSocket.receive(receivingPacket);

String receivedData = new String(receivingPacket.getData());

table1.setValueAt(Double.parseDouble(receivedData), i, 3);

}

clientSocket.close();

} catch (UnknownHostException ex) {

throw new RuntimeException(ex);

} catch (SocketException ex) {

throw new RuntimeException(ex);

} catch (IOException ex) {

throw new RuntimeException(ex);

}

// Thread[] thread = new Thread[9];

// for (int i = 0; i < 9; i++) {

// thread[i] = new Thread(new MyThread(Double.parseDouble(MyModel.getValueAt(i, 0).toString()),

// Double.parseDouble(MyModel.getValueAt(i, 1).toString()),

// Double.parseDouble(MyModel.getValueAt(i, 2).toString()),

// i, table1));

// thread[i].start();

// }

}

});

ClearButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

int count = table1.getRowCount();

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

Stroka.add(new RecIntegral(Double.parseDouble(table1.getValueAt(0, 0).toString()),

Double.parseDouble(table1.getValueAt(0, 1).toString()),

Double.parseDouble(table1.getValueAt(0, 2).toString()),

Double.parseDouble(table1.getValueAt(0, 3).toString())));

MyModel.removeRow(0);

}

}

});

FillButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

for (int i = 0; i < Stroka.size(); i++) {

double high = Stroka.get(i).high;

double low = Stroka.get(i).low;

double height = Stroka.get(i).height;

double result = Stroka.get(i).result;

//MyModel.removeRow(0);

MyModel.addRow(new Object[]{high, low, height, result});

}

Stroka.clear();

}

});

WriteButtonBin.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

ObjectOutputStream out = null;

ArrayList<SaveData> save = new ArrayList<SaveData>();

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

save.add(new SaveData(Double.parseDouble(table1.getValueAt(i, 0).toString()),

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

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

Double.parseDouble(table1.getValueAt(i, 3).toString())));

}

try {

out = new ObjectOutputStream(new BufferedOutputStream(

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

out.writeObject(save);

out.close();

} catch (IOException ex) {

ex.printStackTrace();

}

}

});

OpenButtonBin.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

fileChooser.setDialogTitle("Выбор файла");

//fileChooser.setFileSelectionMode(JFileChooser.DIRECTORIES\_ONLY);

int result = fileChooser.showOpenDialog(Form.this);

if (result == JFileChooser.APPROVE\_OPTION) {

while (MyModel.getRowCount() > 0) {

MyModel.removeRow(0);

}

ObjectInputStream in = null;

ArrayList<SaveData> open = new ArrayList<SaveData>();

try {

in = new ObjectInputStream(new BufferedInputStream(

new FileInputStream(fileChooser.getSelectedFile())));

open = (ArrayList<SaveData>) in.readObject();

for (int i = 0; i < open.size(); i++) {

double first = open.get(i).getFirstColumn();

double second = open.get(i).getSecondColumn();

double third = open.get(i).getThirdColumn();

double fourth = open.get(i).getFourthColumn();

MyModel.addRow(new Object[]{first, second, third, fourth});

}

} catch (IOException ex) {

ex.printStackTrace();

} catch (ClassNotFoundException ex) {

throw new RuntimeException(ex);

}

}

}

});

WriteButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

SaveData save = new SaveData(Double.parseDouble(table1.getValueAt(0, 0).toString()),

Double.parseDouble(table1.getValueAt(0, 1).toString()),

Double.parseDouble(table1.getValueAt(0, 2).toString()),

Double.parseDouble(table1.getValueAt(0, 3).toString()));

try {

FileWriter writer = new FileWriter("A.txt", false);

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

writer.write(table1.getValueAt(i, 0).toString());

writer.write(' ');

writer.write(table1.getValueAt(i, 1).toString());

writer.write(' ');

writer.write(table1.getValueAt(i, 2).toString());

writer.write(' ');

writer.write(table1.getValueAt(i, 3).toString());

writer.write('\n');

writer.flush();

}

} catch (IOException ex) {

throw new RuntimeException(ex);

}

}

});

OpenButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

fileChooser.setDialogTitle("Выбор файла");

//fileChooser.setFileSelectionMode(JFileChooser.DIRECTORIES\_ONLY);

int result = fileChooser.showOpenDialog(Form.this);

if (result == JFileChooser.APPROVE\_OPTION) {

while (MyModel.getRowCount() > 0) {

MyModel.removeRow(0);

}

try {

FileReader open = new FileReader(fileChooser.getSelectedFile());

char[] buf = new char[256];

int a;

String readData = null;

while ((a = open.read(buf)) != -1) {

readData = String.valueOf(buf, 0, a);

}

open.close();

int numberColumn = 0;

StringBuilder first = new StringBuilder();

StringBuilder second = new StringBuilder();

StringBuilder third = new StringBuilder();

StringBuilder fourth = new StringBuilder();

for (int i = 0; i < readData.length(); i++) {

if (readData.charAt(i) != ' ') {

if (readData.charAt(i) != '\n') {

switch (numberColumn) {

case 0:

first.append(readData.charAt(i));

break;

case 1:

second.append(readData.charAt(i));

break;

case 2:

third.append(readData.charAt(i));

break;

case 3:

fourth.append(readData.charAt(i));

break;

}

} else {

MyModel.addRow(new Object[]{Double.parseDouble(first.toString()),

Double.parseDouble(second.toString()),

Double.parseDouble(third.toString()),

Double.parseDouble(fourth.toString())});

first.delete(0, 255);

second.delete(0, 255);

third.delete(0, 255);

fourth.delete(0, 255);

numberColumn = 0;

}

} else numberColumn++;

}

} catch (IOException ex) {

throw new RuntimeException(ex);

}

}

}

});

}

public static void main(String[] args) {

Form f = new Form();

f.setContentPane(f.MainPanel);

f.setTitle("Shedevr");

f.setSize(600, 300);

f.setVisible(true);

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

}

**Файл RecIntegral.java**

public class RecIntegral{  
  
 public double high;  
 public double low;  
 public double height;  
 public double result;  
 public RecIntegral(double high1, double low1, double height1, double result1) {  
 this.high = high1;  
 this.low = low1;  
 this.height = height1;  
 this.result = result1;  
 }  
}

**Файл MyException.java**

public class MyException extends Exception{  
 public MyException(String message){  
 super(message);  
 }  
}

**Файл SaveData.java**

import java.io.Serializable;  
  
public class SaveData implements Serializable {  
 private double FirstColumn;  
 private double SecondColumn;  
 private double ThirdColumn;  
 private double FourthColumn;  
 public SaveData(double FirstColumn, double SecondColumn, double ThirdColumn, double FourthColumn){  
 this.FirstColumn = FirstColumn;  
 this.SecondColumn = SecondColumn;  
 this.ThirdColumn = ThirdColumn;  
 this.FourthColumn = FourthColumn;  
 }  
  
 public double getFirstColumn() {  
 return FirstColumn;  
 }  
  
 public double getSecondColumn() {  
 return SecondColumn;  
 }  
  
 public double getThirdColumn() {  
 return ThirdColumn;  
 }  
  
 public double getFourthColumn() {  
 return FourthColumn;  
 }  
}

**MyThread.java**

import javax.swing.\*;

import static java.lang.Math.cos;

public class MyThread implements Runnable{

private double FirstColumn;

private double SecondColumn;

private double ThirdColumn;

public double s;

private JTable table;

private int i;

public MyThread(double FirstColumn, double SecondColumn, double ThirdColumn, int i, JTable table){

this.FirstColumn = FirstColumn;

this.SecondColumn = SecondColumn;

this.ThirdColumn = ThirdColumn;

this.i = i;

this.table = table;

}

public void run(){

double a = FirstColumn;

double b = SecondColumn;

double h = ThirdColumn;

s = 0;

int j = 0;

for (double i = 0; i <= (a - b) / h; i++) {

if (b + (i + 1) \* h <= a)

s += (cos(b \* b + j \* h) + cos(b \* b + (j + 1) \* h)) \* h / 2;

else

s += (cos(b \* b + j \* h) + cos(a \* a)) \* h / 2;

j++;

}

table.setValueAt(s, i, 3);

}

}

**Листинг №2**

**Файл Main.java**

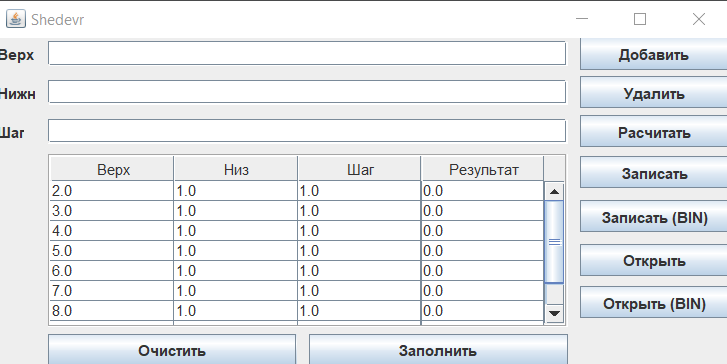
import java.io.IOException;  
import java.net.DatagramPacket;  
import java.net.DatagramSocket;  
import java.net.InetAddress;  
import java.net.SocketException;  
  
public class Main {  
 public static double [] res = new double[9];  
 public static void main(String[] args) throws IOException {  
 try {  
 DatagramSocket serverSocket = new DatagramSocket(8080);  
 byte [] buffer = new byte[1024];  
 DatagramPacket datagramPacket = new DatagramPacket(buffer, buffer.length);  
 String receivedMessage = null;  
 double [] func = new double[4];  
 Thread[] thread = new Thread[9];  
  
 byte [] sendingDataBuffer = new byte[1024];  
 String SendingMessage;  
 InetAddress senderAddress;  
 int senderPort;  
 DatagramPacket outputPacket;  
  
 //clientSocket.setSoTimeout(3000);  
 for (int i = 0; i < 9; i++) {  
 for (int j = 0; j < 3; j++) {  
 serverSocket.receive(datagramPacket);  
 receivedMessage = new String(datagramPacket.getData());  
 func[j] = Double.parseDouble(receivedMessage);  
 System.out.println(func[j]);  
 }  
 thread[i] = new Thread(new MyThread(func[0],func[1],func[2],i));  
 thread[i].start();  
 }  
  
 for(int i = 0; i < 9; i++){  
 thread[i].join();  
 SendingMessage = Double.toString(res[i]);  
 sendingDataBuffer = SendingMessage.getBytes();  
 senderAddress = datagramPacket.getAddress();  
 senderPort = datagramPacket.getPort();  
 outputPacket = new DatagramPacket(sendingDataBuffer, sendingDataBuffer.length, senderAddress, senderPort);  
 serverSocket.send(outputPacket);  
 }  
 serverSocket.close();  
 } catch (InterruptedException e) {  
 throw new RuntimeException(e);  
 } catch (IOException ex) {  
 throw new RuntimeException(ex);  
 }  
 }  
}

**Файл MyThread.java**

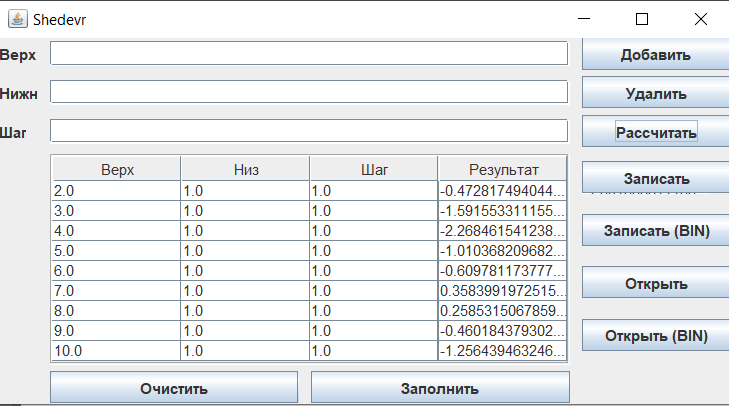
import javax.swing.\*;  
  
import static java.lang.Math.cos;  
  
public class MyThread implements Runnable{  
 private double FirstColumn;  
 private double SecondColumn;  
 private double ThirdColumn;  
 public double s;  
 private JTable table;  
 private int i;  
 public MyThread(double FirstColumn, double SecondColumn, double ThirdColumn, int i){  
 this.FirstColumn = FirstColumn;  
 this.SecondColumn = SecondColumn;  
 this.ThirdColumn = ThirdColumn;  
 this.i = i;  
 }  
 public void run(){  
 double a = FirstColumn;  
 double b = SecondColumn;  
 double h = ThirdColumn;  
 s = 0;  
 int j = 0;  
 for (double i = 0; i <= (a - b) / h; i++) {  
 if (b + (i + 1) \* h <= a)  
 s += (cos(b \* b + j \* h) + cos(b \* b + (j + 1) \* h)) \* h / 2;  
 else  
 s += (cos(b \* b + j \* h) + cos(a \* a)) \* h / 2;  
 j++;  
 }  
 System.out.println(s);  
 Main.res[i] = s;  
 }  
}

**Результат работы программы**

Заполнили таблицу



Нажали на кнопку «Рассчитать» и отправили строки в виде пакета данных на сервер. Сервер обработал данные и вернул результат вычислений.



**Вывод:** Сегодня на лабораторной работе мы научились создавать клиент-серверные приложения c использованием стандартных классов Java.