Пензенский Государственный университет

Кафедра «Вычислительная техника»

**Отчёт**

По лабораторной работе №6

По дисциплине: «Программирование на языке JAVA»

По теме: «Сетевое взаимодействие в Java»

Вариант 2

Выполнили:

Студенты группы 20ВВП2

Мирясов Н.А.

Культиясов Я.К.

Приняли:

Юрова О.В.

Карамышева Н.С.

Пенза 2023

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

**Задание**: 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. Оформление лабораторной работы должно быть выполнено в соответствии с требованиями, приведенными в Приложении 2.

**Ход выполнения работы**

Расширили функционал программы из предыдущей лабораторной работы, создав класс «Client», на котором проводим вычисление интеграла, которое происходит с использованием нескольких нитей. При реализации соединения типа «Клиент-Сервер» использовали сокеты, для передачи которых использовали протокол TCP.

**Листинг**

**Файл Form\_Integral.java**

import javax.swing.\*;

import javax.swing.filechooser.FileNameExtensionFilter;

import javax.swing.plaf.multi.MultiTableHeaderUI;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import java.io.\*;

import java.net.\*;

import java.nio.charset.StandardCharsets;

import java.util.LinkedList;

import java.util.Vector;

import java.util.function.Function;

import java.util.logging.Level;

import java.util.logging.Logger;

public class Form\_Integral extends JFrame{

private JButton addbtn = new JButton("Добавить"),

delbtn = new JButton("Удалить"),

countbtn = new JButton("Вычислить"),

cleanbtn = new JButton("Очистить"),

fillbtn = new JButton(("Заполнить")),

serDec =new JButton("СериализацияDec"),

serBin =new JButton("СериализацияBin"),

deserDec =new JButton("ДесериализацияDec"),

deserBin =new JButton("ДесериализацияBin");

private JTextField highText = new JTextField("",10),

lowText = new JTextField("",10),

StepText = new JTextField("",10);

private JLabel highlabel = new JLabel("Введите верхнюю границу: "),

lowlabel = new JLabel("Введите нижнюю границу: "),

steplabel = new JLabel("Введите шаг интегрирования: ");

private JTable table;

// LinkedList

public class List

{

public LinkedList<RecIntegral> dataList = new LinkedList<RecIntegral>();

public LinkedList<RecIntegral> getDataList()

{

return dataList;

}

}

List list =new List();

// JFrame

static JFrame f;

DatagramSocket socet;

InetAddress address;

//Создаём отоброжаемую на экране форму

public Form\_Integral() throws IOException, SocketException

{

socet = new DatagramSocket();

address = InetAddress.getByName("localhost");

setTitle("Вычисление интеграла");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setBounds(100,100,550,450);

Container c = getContentPane();

JPanel panel = new JPanel(),

btnpanel = new JPanel();

JScrollPane tablePanel = new JScrollPane();

GroupLayout layout = new GroupLayout(panel);

panel.setLayout(layout);

layout.setAutoCreateGaps(true);

layout.setAutoCreateContainerGaps(true);

layout.setHorizontalGroup(

layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)

.addComponent(highlabel)

.addComponent(lowlabel)

.addComponent(steplabel)

)

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)

.addComponent(highText)

.addComponent(lowText)

.addComponent(StepText)

)

);

layout.setVerticalGroup(

layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)

.addComponent(highlabel)

.addComponent(highText))

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)

.addComponent(lowlabel)

.addComponent(lowText))

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)

.addComponent(steplabel)

.addComponent(StepText))

);

btnpanel.setLayout(new GridLayout(3,4,4,4));

//Панель с кнопками

countbtn.addActionListener(new BtnEventListener());

addbtn.addActionListener(new AddDataBtn());

delbtn.addActionListener(new DeleteDataBtn());

cleanbtn.addActionListener(new CleanDataBtn());

fillbtn.addActionListener(new FillDataBtn());

serDec.addActionListener(new MyFileWriterDec());

deserDec.addActionListener(new MyFileReaderDec());

serBin.addActionListener(new MyFileWriterBin());

deserBin.addActionListener(new MyFileReaderBin());

btnpanel.add(addbtn);

btnpanel.add(delbtn);

btnpanel.add(countbtn);

btnpanel.add(cleanbtn);

btnpanel.add(fillbtn);

btnpanel.add(serDec);

btnpanel.add(serBin);

btnpanel.add(deserDec);

btnpanel.add(deserBin);

btnpanel.setBorder(BorderFactory.createEmptyBorder(10,10,10,10));

//Панель с табличкой

String[] columnNames = { "Верхняя граница", "Нижняя граница", "Шаг интегрирования", "Результат" };

int numRows = 0;

DefaultTableModel model = new DefaultTableModel(numRows,columnNames.length);

model.setColumnIdentifiers(columnNames);

table = new JTable(model)

{

public boolean isCellEditable(int row, int column)

{

return column !=3;

}

};

table.setSize(530,250);

table.setBackground(new Color(164,191,220));

table.addMouseListener(new TableMouseClicked());

panel.setBackground(new Color(164,191,220));

//btnpanel.setBackground(Color.blue);

//tablePanel.setBackground(Color.green);

c.setLayout(new BoxLayout(c, BoxLayout.Y\_AXIS));

c.add(panel);

c.add(btnpanel);

c.add(new JScrollPane(table));

}

//Эта функция выводит на текстовые поля данные из строки таблицы,на которую мы нажали

class TableMouseClicked implements MouseListener{

public void mouseClicked(MouseEvent e) {

DefaultTableModel tblModel = (DefaultTableModel)table.getModel();

String tblHigh = tblModel.getValueAt(table.getSelectedRow(),0).toString();

String tblLow = tblModel.getValueAt(table.getSelectedRow(),1).toString();

String tblStep = tblModel.getValueAt(table.getSelectedRow(),2).toString();

highText.setText(tblHigh);

lowText.setText(tblLow);

StepText.setText(tblStep);

}

@Override

public void mousePressed(MouseEvent e) {

}

@Override

public void mouseReleased(MouseEvent e) {

}

@Override

public void mouseEntered(MouseEvent e) {

}

@Override

public void mouseExited(MouseEvent e) {

}

}

class JThread extends Thread

{

public double top;

public double low;

public double step;

public double result;

int size;

public JThread()

{

top = 0;

low = 0;

step = 0;

result = 0;

}

public void setData(double newTop, double newLow, double newStep)

{

top=newTop;

low=newLow;

step=newStep;

}

public double getResult()

{

return this.result;

}

public void run()

{

DefaultTableModel module = (DefaultTableModel) table.getModel();

DatagramSocket socket = null;

try {

socket = new DatagramSocket(26);

} catch (SocketException ex) {

Logger.getLogger(Form\_Integral.class.getName()).log(Level.SEVERE, null, ex);

}

for(int i = 0; i < size; i++)

{

byte[] buffer = new byte[256];

DatagramPacket request = new DatagramPacket(buffer, buffer.length);

try {

socket.receive(request);

String Message = new String(request.getData(), 0, request.getLength());

String Resoult = "",

Num = "";

int j = 0;

while (Message.charAt(j) != ' ')

{

Resoult += Message.charAt(j);

j++;

}

j++;

while (j != Message.length())

{

Num += Message.charAt(j);

j++;

}

module.setValueAt(Float.parseFloat(Resoult), Integer.parseInt(Num), 3);

} catch (IOException ex) {

Logger.getLogger(Form\_Integral.class.getName()).log(Level.SEVERE, null, ex);

}

}

socket.close();

}

}

//В этой функции совершается подсчёт интеграла

class BtnEventListener extends Component implements ActionListener{

//Здесь будет выполняться вычисление интеграла

public void actionPerformed (ActionEvent e)

{

int CountOfThreads=4;

double Interval;

double Top = Double.valueOf(highText.getText()),

Low = Double.valueOf(lowText.getText()),

Step = Double.valueOf(StepText.getText()),

integral = 0;

Interval =(Top-Low)/CountOfThreads;

byte[] buf;

String message = String.valueOf(Top) + " " + String.valueOf(Low) + " " + String.valueOf(Step) + " " + String.valueOf(Interval)+ " " + String.valueOf(CountOfThreads);

buf = message.getBytes();

DatagramPacket packet = new DatagramPacket(buf, buf.length, address, 17);

try

{

socet.send(packet);

System.out.println(buf);

}

catch (IOException ex)

{

Logger.getLogger(Form\_Integral.class.getName()).log(Level.SEVERE, null, ex);

}

DatagramSocket newSocket = null;

DatagramSocket socketSend = null;

InetAddress address = null;

try

{

newSocket = new DatagramSocket(26);

socketSend = new DatagramSocket();

address = InetAddress.getByName("localhost");

}

catch (SocketException ex)

{

throw new RuntimeException(ex);

}

catch (UnknownHostException ex)

{

throw new RuntimeException(ex);

}

byte[] buffer = new byte[256];

DatagramPacket request = new DatagramPacket(buffer, buffer.length);

try

{

newSocket.receive(request);

String Message = new String(request.getData(), 0, request.getLength());

String Resoult = "";

int size =Message.length();

int j = 0;

while (j != size)

{

Resoult += Message.charAt(j);

j++;

}

integral=Double.valueOf(Resoult);

}

catch (IOException ex)

{

Logger.getLogger(Form\_Integral.class.getName()).log(Level.SEVERE, null, ex);

}

newSocket.close();

//Записываем данные в таблицу

DefaultTableModel tblModel = (DefaultTableModel) table.getModel();

if (table.getSelectedColumnCount() == 1)

{

String high = highText.getText(),

low = lowText.getText(),

step = StepText.getText();

tblModel.setValueAt(high, table.getSelectedRow(), 0);

tblModel.setValueAt(low, table.getSelectedRow(), 1);

tblModel.setValueAt(step, table.getSelectedRow(), 2);

tblModel.setValueAt(String.valueOf(integral), table.getSelectedRow(), 3);

}

else

{

if (table.getRowCount() == 0)

{

JOptionPane.showMessageDialog(null, "Таблица пустая!");

}

else

{

JOptionPane.showMessageDialog(null, "Выберите строку,которую надо вычислить!");

}

}

}

}

//Добавление информации в таблицу

class AddDataBtn extends Component implements ActionListener

{

public void actionPerformed (ActionEvent e)

{

String data[] = {highText.getText(), lowText.getText(), StepText.getText()};

try

{

if(Double.valueOf(StepText.getText()) > Double.valueOf(highText.getText()))

{

throw new IntegralException("Шаг больше верхнего предела!");

}

else if (Double.valueOf(StepText.getText()) < 0.0)

{

throw new IntegralException("Шаг меньше 0!");

}

else if (Double.valueOf(lowText.getText()) > Double.valueOf(highText.getText()))

{

throw new IntegralException("Нижний предел больше верхнего!");

}

else if (Double.valueOf(highText.getText())<0.000001 || Double.valueOf(highText.getText())>1000000)

{

throw new IntegralException("Верхний предел имеет неправильное значение!");

}

else if (Double.valueOf(lowText.getText())<0.000001 || Double.valueOf(lowText.getText())>1000000)

{

throw new IntegralException("Нижний предел имеет неправильное значение!");

}

else if (Double.valueOf(StepText.getText())<0.000001 || Double.valueOf(StepText.getText())>1000000)

{

throw new IntegralException("Шаг имеет неправильное значение!");

}

}

catch (Exception ex)

{

return;

}

RecIntegral recIntegral = new RecIntegral();

recIntegral.setData(data);

//List list = new List();

list.getDataList().add(recIntegral);

//dataList.add(recIntegral);

DefaultTableModel tblModel = (DefaultTableModel) table.getModel();

tblModel.addRow(data);

//очищаем поле для новых данных

highText.setText("");

lowText.setText("");

StepText.setText("");

}

}

//Функция очищающая таблицу

class CleanDataBtn extends Component implements ActionListener

{

public void actionPerformed(ActionEvent e){

DefaultTableModel tblModel = (DefaultTableModel)table.getModel();

tblModel.setRowCount(0);

}

}

//Функция заполняющая таблицу данными из LinkedList`а

class FillDataBtn extends Component implements ActionListener

{

public void actionPerformed(ActionEvent e){

DefaultTableModel tblModel = (DefaultTableModel) table.getModel();

tblModel.setRowCount(0);

for(int i = 0; i < list.getDataList().size(); i++)

{

tblModel.addRow(list.getDataList().get(i).dataFromList);

}

}

}

//Удаление данных из таблицы

class DeleteDataBtn extends Component implements ActionListener{

public void actionPerformed(ActionEvent e){

DefaultTableModel tblModel = (DefaultTableModel)table.getModel();

if (table.getSelectedColumnCount() == 1){

tblModel.removeRow(table.getSelectedRow());

}else{

if (table.getRowCount() == 0){

JOptionPane.showMessageDialog(null, "Таблица пустая!");

}

else{

//если таблица не пустая, но ничего не выбрано

JOptionPane.showMessageDialog(null, "Вы ничего не выбрали!");

}

}

}

}

//Промежуточный класс для хранение одной строки данных

class RecIntegral implements Serializable

{

public String[] dataFromList = new String[3];

public void setData(String new\_data[])

{

dataFromList=new\_data;

}

public String[] getData()

{

return dataFromList;

}

}

//Класс обработки исключений значений класса RecIntegral

class IntegralException extends Exception

{

public IntegralException(String message)

{

JOptionPane.showMessageDialog(null, message);

}

}

//Класс сериализации

class MyFileWriterDec implements ActionListener

{

public void actionPerformed (ActionEvent e)

{

OpenDialogBox odb = new OpenDialogBox();

String FileName = odb.DialogSave("txt");

try(FileWriter writer = new FileWriter(FileName, true))

{

String text[]=null;

for(int i = 0; i < list.getDataList().size(); i++)

{

text = list.getDataList().get(i).dataFromList;

writer.write(text[0]+" " + text[1]+" " +text[2]+"\n");

writer.flush();

}

DefaultTableModel tblModel = (DefaultTableModel) table.getModel();

tblModel.setRowCount(0);

}

catch(IOException ex)

{

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

}

list.getDataList().clear();

}

}

//Класс бинарной сериализации

class MyFileWriterBin implements ActionListener

{

public void actionPerformed(ActionEvent e)

{

OpenDialogBox odb = new OpenDialogBox();

String FileName = odb.DialogSave("bin");

LinkedList<String> arr = new LinkedList<String>();

try

{

FileOutputStream fos = new FileOutputStream(FileName);

BufferedOutputStream bis = new BufferedOutputStream(fos);

ObjectOutputStream oos = new ObjectOutputStream(bis);

for (int i = 0; i < list.getDataList().size(); i++)

{

Object Step = null, Lower = null, Top = null;

RecIntegral Node = list.getDataList().get(i);

Top = Node.dataFromList[0];

Lower = Node.dataFromList[1];

Step = Node.dataFromList[2];

arr.add(Top.toString() + ' ' + Lower.toString() + ' ' + Step.toString());

}

oos.writeObject(arr);

oos.close();

} catch (FileNotFoundException ex) {

throw new RuntimeException(ex);

} catch (IOException ex) {

throw new RuntimeException(ex);

}

DefaultTableModel tblModel = (DefaultTableModel) table.getModel();

tblModel.setRowCount(0);

list.getDataList().clear();

}

}

//Класс бинарной десериализации

class MyFileReaderBin implements ActionListener

{

public void actionPerformed(ActionEvent e)

{

OpenDialogBox odb = new OpenDialogBox();

String FileName = odb.DialogOpen("bin");

LinkedList<String> arr = new LinkedList<String>();

DefaultTableModel tblModel = (DefaultTableModel)table.getModel();

try

{

FileInputStream fis = new FileInputStream(FileName);

BufferedInputStream bis = new BufferedInputStream(fis);

ObjectInputStream ois = new ObjectInputStream(bis);

arr = (LinkedList<String>)ois.readObject();

if(arr==null)

{

return;

}

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

{

String str = arr.get(i),

strTop = "",

strLower = "",

strStep = "";

int size = str.length();

int j = 0;

while (str.charAt(j) != ' ')

{

strTop += str.charAt(j);

j++;

}

j++;

while (str.charAt(j) != ' ')

{

strLower += str.charAt(j);

j++;

}

j++;

while (j != size)

{

strStep += str.charAt(j);

j++;

}

RecIntegral Node = new RecIntegral();

String data[] = {strTop, strLower, strStep};

Node.setData(data);

list.getDataList().add(Node);

tblModel.addRow(list.getDataList().get(i).dataFromList);

}

} catch (FileNotFoundException ex) {

throw new RuntimeException(ex);

} catch (IOException ex) {

//throw new RuntimeException(ex);

} catch (ClassNotFoundException ex) {

throw new RuntimeException(ex);

}

PrintWriter writer = null;

try {

writer = new PrintWriter(FileName);

} catch (FileNotFoundException ex) {

throw new RuntimeException(ex);

}

writer.print("");

writer.close();

}

}

//Класс десериализации

class MyFileReaderDec implements ActionListener

{

public void actionPerformed(ActionEvent e){

DefaultTableModel tblModel = (DefaultTableModel) table.getModel();

try

{

OpenDialogBox odb = new OpenDialogBox();

String FileName = odb.DialogOpen("txt");

FileReader fReader = new FileReader(FileName);

BufferedReader reader = new BufferedReader(fReader);

String readStr = reader.readLine();

if(readStr==null)

{

return;

}

while(readStr != null)

{

String[] words =readStr.split(" ");

String data[] = {words[0], words[1], words[2]};

RecIntegral recIntegral = new RecIntegral();

recIntegral.setData(data);

list.getDataList().add(recIntegral);

readStr=reader.readLine();

}

FileWriter writer = new FileWriter(FileName, false);

PrintWriter pWriter = null;

try {

pWriter = new PrintWriter(FileName);

} catch (FileNotFoundException ex) {

throw new RuntimeException(ex);

}

}

catch(IOException ex)

{

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

}

for(int i = 0; i < list.getDataList().size(); i++)

{

tblModel.addRow(list.getDataList().get(i).dataFromList);

}

}

}

class OpenDialogBox

{

String FileName;

public String DialogSave(String NameFilter)

{

JFileChooser chooser = new JFileChooser();

FileNameExtensionFilter filter = new FileNameExtensionFilter(

"txt & bin", NameFilter);

chooser.setFileFilter(filter);

int returnVal = chooser.showSaveDialog(null);

if(returnVal == JFileChooser.APPROVE\_OPTION)

{

FileName = chooser.getSelectedFile().getAbsolutePath();

}

return FileName;

}

public String DialogOpen(String NameFilter)

{

JFileChooser chooser = new JFileChooser();

FileNameExtensionFilter filter = new FileNameExtensionFilter(

"txt & bin", NameFilter);

chooser.setFileFilter(filter);

int returnVal = chooser.showOpenDialog(null);

if(returnVal == JFileChooser.APPROVE\_OPTION)

{

FileName = chooser.getSelectedFile().getAbsolutePath();

}

return FileName;

}

}

}

**Файл Main.java**

import java.io.IOException;

public class Main {

public static void main(String[] args) throws IOException {

Form\_Integral app = new Form\_Integral();

app.setVisible(true);

}

}

**Файл Client.java**

package client;

import java.io.IOException;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

import java.net.SocketException;

import java.util.Vector;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.swing.table.DefaultTableModel;

class JThread extends Thread

{

public double top;

public double low;

public double step;

public double result;

public int strNumber;

public JThread()

{

top = 0;

low = 0;

step = 0;

strNumber = 0;

result = 0;

}

public void setData(double newTop, double newLow, double newStep)

{

top=newTop;

low=newLow;

step=newStep;

}

public double getResult()

{

return this.result;

}

public void run()

{

for (double i = low; i < top; i += step) {

if(top - i > step)

{

result += step \* (0.5 \* (Math.sin(i) + Math.sin(i + step)));

}

else

{

result += (top - i) \* (0.5 \* (Math.sin(i) + Math.sin(i + top)));

}

}

}

}

public class Client

{

public static void main(String[] args) throws SocketException, IOException {

DatagramSocket socket = new DatagramSocket(17);

DatagramSocket socketSend = new DatagramSocket();

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

String messageSend;

double Integral=0;

while(true) {

byte[] buffer = new byte[256];

DatagramPacket request = new DatagramPacket(buffer, buffer.length);

socket.receive(request);

if (request.getLength() != 0) {

String Message = new String(request.getData(), 0, request.getLength());

String strTop = "",

strLower = "",

strStep = "",

strInterval = "",

strCountOfThreads = "";

int size = Message.length();

int j = 0;

while (Message.charAt(j) != ' ') {

strTop += Message.charAt(j);

j++;

}

j++;

while (Message.charAt(j) != ' ') {

strLower += Message.charAt(j);

j++;

}

j++;

while (Message.charAt(j) != ' ') {

strStep += Message.charAt(j);

j++;

}

j++;

while (Message.charAt(j) != ' ') {

strInterval += Message.charAt(j);

j++;

}

j++;

while (j != size) {

strCountOfThreads += Message.charAt(j);

j++;

}

int CountOfThreads = Integer.valueOf(strCountOfThreads);

JThread[] MyThread = new JThread[CountOfThreads];

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

MyThread[i] = new JThread();

MyThread[i].setData((Double.valueOf(strLower) + (Double.valueOf(strInterval) \* (i + 1))), (Double.valueOf(strLower) + (Double.valueOf(strInterval) \* i)), Double.valueOf(strStep));

MyThread[i].start();

try

{

MyThread[i].join();

}

catch (InterruptedException ex)

{

throw new RuntimeException(ex);

}

System.out.println(1432);

Integral += MyThread[i].getResult();

}

}

messageSend = String.valueOf(Integral);

Integral=0;

byte[] buf = messageSend.getBytes();

DatagramPacket packet = new DatagramPacket(buf, buf.length, address, 26);

try {

socketSend.send(packet);

} catch (IOException ex) {

Logger.getLogger(JThread.class.getName()).log(Level.SEVERE, null, ex);

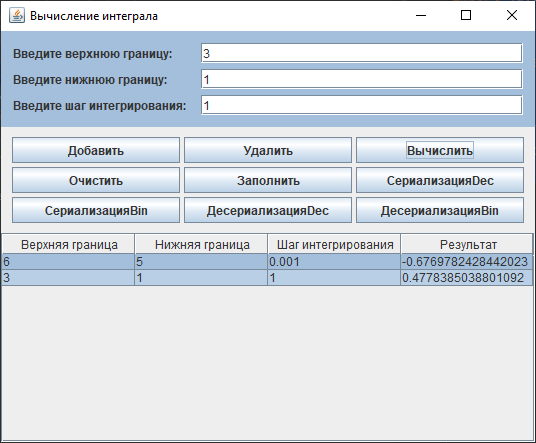
}

}

}

}

**Результат работы**



**Вывод**: научились создавать клиент-серверные приложения c использованием стандартных классов Java.