Министерство науки и высшего образования Российской федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №4

по курсу «Программирование на языке Java»

на тему «Сетевое взаимодействие в Java»

Вариант 5

Выполнили

студенты группы 21ВВП2:

Лукина Е.Ю.

Ванюшин И.А.

Тарасов Н.А.

Приняли:

Юрова О.В.

Карамышева Н.С.

Пенза 2024

**Цель работы**

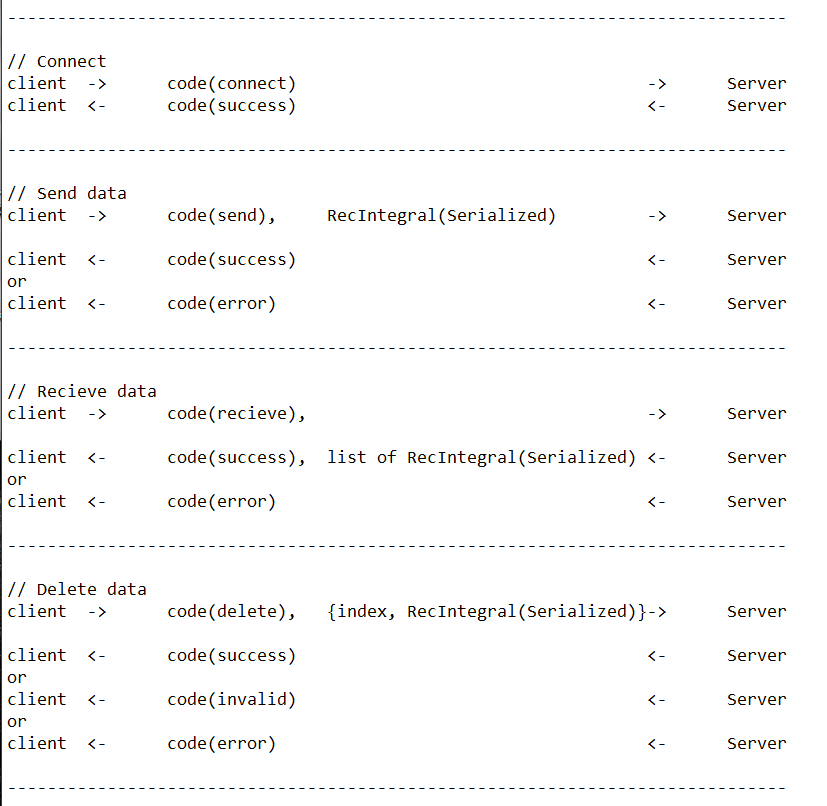
Научиться создавать клиент-серверные приложения c использованием стандартных классов Java.

**Задание на лабораторную работу**

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.

|  |  |  |
| --- | --- | --- |
| Номер варианта | Функция | Протокол |
| 5 |  | UDP |

**Ход работы**

На основе написанных ранее лабораторных работ написали интерфейс для сервера, который будет хранить записи от пользователей, синхронизировать их и удалять. Все функции, описанные в программе, представлены в протоколе ниже.

Для упрощения взаимодействия двух автоматов написали две нумерации, являющиеся кодами запросов клиента и кодами ответа сервера.

Написали класс, отвечающий за мониторинг за дочерними нитями обработки входящих и исходящих пакетов данных. У нитей нет возможности обрабатывать два пакета данных истинно синхронно, поэтому преимущество параллельного программирования в данном случае будет определено промежутком обработки входящей информации и формирования ответа клиенту на стороне сервера.

Для дочерних нитей сервера написали класс, полностью реализующий показанный ранее протокол со стороны сервера.

Реализовали вторую часть протокола со стороны клиента, написав отдельный класс для упрощения использования процедур подключений, а также мониторинга подключения.

**Листинг**

Клиент:

Lab1.MyWin.java

package lab1;

import java.awt.Container;

import java.awt.Dimension;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.BufferedInputStream;

import java.io.BufferedOutputStream;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.util.ArrayList;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.JTextField;

import javax.swing.table.DefaultTableModel;

import javax.swing.JOptionPane;

import javax.swing.JDialog;

import javax.swing.JFileChooser;

import java.net.DatagramSocket;

import java.net.DatagramPacket;

public class MyWin extends JFrame implements ActionListener

{

// Some window's elements

private final JTextField jtfA;

private final JTextField jtfB;

private final JTextField jtfE;

private final DefaultTableModel dtResultsModel;

private final JTable jtResults;

private final JDialog jdFrame;

private final JFileChooser jfc;

private final ClientInstance ci;

private ArrayList alRecs;

private final byte[] buffer;

// Класс для вычисления определенного интеграла

private final IntegralCalculator ic;

// Classes serial number

private static final long serialVersionUID = 1L;

public MyWin()

{

// Initialization block

jdFrame = new JDialog();

alRecs = new ArrayList();

ic = new IntegralCalculator();

jtfA = new JTextField();

jtfB = new JTextField();

jtfE = new JTextField();

jfc = new JFileChooser();

buffer = new byte[32767];

// ErrorDialog configuration

jfc.setDialogTitle("File selection");

jfc.setFileSelectionMode(JFileChooser.FILES\_ONLY);

//----------------==Table Filling==------------------------------

Object[] columnNames = new String[]{"L limit", "R limit", "Step", "Result"};

Object[][] data = new String[][]{};

// Tables initialization

dtResultsModel = new DefaultTableModel(data, columnNames);

dtResultsModel.setColumnIdentifiers(columnNames);

jtResults = new JTable(dtResultsModel);

JScrollPane sp = new JScrollPane(jtResults);

ci = new ClientInstance(dtResultsModel);

Container c = getContentPane(); // Client rect

c.setLayout(null); // НЕНАВИЖУ КОМПАНОВЩИКИ

//----------------==Left Panel's Filling==-----------------------

// Creates a left sided panel

JPanel jpLeftPanel = new JPanel();

jpLeftPanel.setLayout(null); // АГРЕССИЯ И ЗУБЫ СКРИПЯТ

jpLeftPanel.setBounds(0, 0, 200, 480); // Resizes left panel

// Creates buttons.

// The one that Adds new records & the other that Clears all inputs.

JButton jbtnAdd = new JButton("Add");

JButton jbtnClear = new JButton("Clear");

JButton jbtnDelete = new JButton("Delete");

JButton jbtnLoad = new JButton("Load");

JButton jbtnInFileBin = new JButton("Des(bin)");

JButton jbtnInFileTxt = new JButton("Des(txt)");

JButton jbtnOutFileBin = new JButton("Ser(bin)");

JButton jbtnOutFileTxt = new JButton("Ser(txt)");

JButton jbtnConnect = new JButton("Connect");

JButton jbtnSend = new JButton("Send(Srv)");

JButton jbtnRecieve = new JButton("Rcv(Srv)");

JButton jbtnDeleteSrv = new JButton("Del(Srv)");

// Buttons's resize.

jbtnAdd.setBounds(10, 200, 90, 25);

jbtnClear.setBounds(100, 200, 90, 25);

jbtnLoad.setBounds(10, 230, 90, 25);

jbtnDelete.setBounds(100, 230, 90, 25);

jbtnInFileBin.setBounds(10, 260, 90, 25);

jbtnInFileTxt.setBounds(100, 260, 90, 25);

jbtnOutFileBin.setBounds(10, 290, 90, 25);

jbtnOutFileTxt.setBounds(100, 290, 90, 25);

jbtnConnect.setBounds(10, 320, 90, 25);

jbtnSend.setBounds(100, 320, 90, 25);

jbtnRecieve.setBounds(10, 350, 90, 25);

jbtnDeleteSrv.setBounds(100, 350, 90, 25);

// Adds action listeners for the objects.

jbtnAdd.addActionListener(this);

jbtnClear.addActionListener(this);

jbtnDelete.addActionListener(this);

jbtnLoad.addActionListener(this);

jbtnInFileBin.addActionListener(this);

jbtnInFileTxt.addActionListener(this);

jbtnOutFileBin.addActionListener(this);

jbtnOutFileTxt.addActionListener(this);

jbtnConnect.addActionListener(this);

jbtnSend.addActionListener(this);

jbtnRecieve.addActionListener(this);

jbtnDeleteSrv.addActionListener(this);

// Resizes TextFields

jtfA.setBounds(10, 10, 180, 50);

jtfB.setBounds(10, 70, 180, 50);

jtfE.setBounds(10, 130, 180, 50);

// Sets default values to TextFields

jtfA.setText(Double.toString(ic.A()));

jtfB.setText(Double.toString(ic.B()));

jtfE.setText(Double.toString(ic.E()));

sp.setBounds(200, 0, 427, 480);

// Fills panel with early created buttons & txtFields.

jpLeftPanel.add(jtfA);

jpLeftPanel.add(jtfB);

jpLeftPanel.add(jtfE);

jpLeftPanel.add(jbtnAdd);

jpLeftPanel.add(jbtnClear);

jpLeftPanel.add(jbtnDelete);

jpLeftPanel.add(jbtnLoad);

jpLeftPanel.add(jbtnInFileBin);

jpLeftPanel.add(jbtnInFileTxt);

jpLeftPanel.add(jbtnOutFileBin);

jpLeftPanel.add(jbtnOutFileTxt);

jpLeftPanel.add(jbtnConnect);

jpLeftPanel.add(jbtnSend);

jpLeftPanel.add(jbtnRecieve);

jpLeftPanel.add(jbtnDeleteSrv);

//----------------==Client rect's Filling==-----------------------

// Adds left panel to actually left side of client rect.

c.add(jpLeftPanel);

c.add(sp);

//----------------==Setting window's properties==-----------------

// Window's properties.

setTitle("Integral Calculator"); // Window's title. Obviously.

// Sets preffered size to window.

setPreferredSize(new Dimension(640, 480));

// Exit application on Exit button.

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

pack(); // Setting preffered sizes.

setVisible(true); // Makes window visible.

}

@Override

public void actionPerformed(ActionEvent e) {

// "Calculate" case. Calculates the integral with setted parameters.

if (e.getActionCommand().equals("Add"))

{

// Thread here was made to split GUI from calculation stuff

Runnable task = () -> {

int lastI = alRecs.size() - 1; // Saves current row's index to write the result in it later

double result = 0;

if (lastI == -1) return;

try {

double tmpA = ic.A(), tmpB = ic.B(), tmpE = ic.E();

double[] parts = {0, 0, 0, 0, 0};

double integPart = Math.abs(tmpB - tmpA) / 5;

Thread[] thrs = {null, null, null, null, null};

for (int i = 0; i < 5; i++){

int k = i;

// Thread here was made to split Integration on five parts

Runnable part = () -> {

int j = k;

try{

parts[j] = IntegralCalculator.F(tmpA + integPart \* j, tmpA + integPart \* (j + 1), tmpE);

}

catch(IntegralInputException exc){

exc.printStackTrace();

}

};

thrs[i] = new Thread(part);

thrs[i].start();

}

for (int i = 0; i < 5; i++) {

thrs[i].join(); // Syncronization point

result += parts[i]; // Summarizing all parts together

}

((RecIntegral)alRecs.get(lastI)).SetResult(result); // Sets the result to Collection

dtResultsModel.setValueAt(Double.toString(result), lastI, 3); // Sets the result to Table

// SEND REQUEST

}

catch(IntegralInputException exc)

{

dtResultsModel.setValueAt("Error", lastI, 3);

JOptionPane.showMessageDialog(

jdFrame,

exc.getMessage(),

"Error",

JOptionPane.ERROR\_MESSAGE

);

}

catch(InterruptedException exc){

exc.printStackTrace();

}

};

Thread CalcThr = new Thread(task);

try

{

if (Thread.activeCount() > 6){

return;

}

ic.SetArea(Double.parseDouble(jtfA.getText()), Double.parseDouble(jtfB.getText()));

ic.SetE(Double.parseDouble(jtfE.getText()));

alRecs.add(new RecIntegral(ic));

jtfE.setText(((RecIntegral)alRecs.getLast()).Dx());

dtResultsModel.addRow(new String[]

{

String.valueOf(ic.A()),

String.valueOf(ic.B()),

String.valueOf(ic.E()),

"Calculating..."

});

CalcThr.start();

}

catch(IntegralInputException exc)

{

JOptionPane.showMessageDialog(

jdFrame,

exc.getMessage(),

"Error",

JOptionPane.ERROR\_MESSAGE

);

}

catch(NumberFormatException exc)

{

JOptionPane.showMessageDialog(

jdFrame,

exc.getMessage(),

"Error",

JOptionPane.ERROR\_MESSAGE

);

}

}

else if (e.getActionCommand().equals("Clear"))

{

dtResultsModel.setRowCount(0);

}

else if (e.getActionCommand().equals("Load"))

{

dtResultsModel.setRowCount(0);

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

{

dtResultsModel.addRow(new String[]

{

((RecIntegral)alRecs.get(i)).LimitL(),

((RecIntegral)alRecs.get(i)).LimitR(),

((RecIntegral)alRecs.get(i)).Dx(),

((RecIntegral)alRecs.get(i)).Result()

});

}

}

else if (e.getActionCommand().equals("Delete"))

{

int i = jtResults.getSelectedRow();

if (i != -1)

{

dtResultsModel.removeRow(i);

alRecs.remove(i);

}

}

else if (e.getActionCommand().equals("Ser(bin)"))

{

ObjectOutputStream out = null;

File fileOpen = null;

if (jfc.showOpenDialog(this) == JFileChooser.APPROVE\_OPTION)

{

fileOpen = jfc.getSelectedFile();

}

else

{

return;

}

try

{

out = new ObjectOutputStream(new BufferedOutputStream(new FileOutputStream(fileOpen)));

out.writeObject(alRecs);

out.close();

}

catch (IOException ex)

{

ex.printStackTrace();

}

}

else if (e.getActionCommand().equals("Des(bin)"))

{

ObjectInputStream out = null;

File fileOpen = null;

if (jfc.showOpenDialog(this) == JFileChooser.APPROVE\_OPTION)

{

fileOpen = jfc.getSelectedFile();

}

else

{

return;

}

try

{

out = new ObjectInputStream(new BufferedInputStream(new FileInputStream(fileOpen)));

alRecs = (ArrayList)out.readObject();

out.close();

}

catch (IOException ex)

{

ex.printStackTrace();

}

catch (ClassNotFoundException ex)

{

ex.printStackTrace();

}

}

else if (e.getActionCommand().equals("Ser(txt)"))

{

FileWriter out = null;

File fileOpen = null;

if (jfc.showOpenDialog(this) == JFileChooser.APPROVE\_OPTION)

{

fileOpen = jfc.getSelectedFile();

}

else

{

return;

}

try

{

out = new FileWriter(fileOpen);

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

{

RecIntegral tmp = (RecIntegral)alRecs.get(i);

out.write(String.format("%s %s %s %s\n", tmp.LimitL(), tmp.LimitR(), tmp.Dx(), tmp.Result()));

}

out.close();

}

catch (IOException ex)

{

ex.printStackTrace();

}

}

else if (e.getActionCommand().equals("Des(txt)"))

{

BufferedReader out = null;

String params = "";

String[] splitedPars;

File fileOpen = null;

if (jfc.showOpenDialog(this) == JFileChooser.APPROVE\_OPTION)

{

fileOpen = jfc.getSelectedFile();

}

else

{

return;

}

try

{

out = new BufferedReader(new FileReader(fileOpen));

alRecs.clear();

params = out.readLine();

while (params != null)

{

splitedPars = params.split(" ");

alRecs.add(new RecIntegral(Double.valueOf(splitedPars[0]), Double.valueOf(splitedPars[1]), Double.valueOf(splitedPars[2]), Double.valueOf(splitedPars[3])));

params = out.readLine();

}

out.close();

}

catch (IOException ex)

{

ex.printStackTrace();

}

catch (IntegralInputException ex)

{

ex.printStackTrace();

}

}

else if (e.getActionCommand().equals("Connect"))

{

Runnable guiSplit = () -> {

try

{

ci.Connect();

}

catch(Exception exc)

{

return;

}

};

Thread lesGoo = new Thread(guiSplit);

lesGoo.start();

}

else if (e.getActionCommand().equals("Send(Srv)"))

{

Runnable guiSplit = () -> {

ci.Send();

};

Thread lesGoo = new Thread(guiSplit);

lesGoo.start();

}

else if (e.getActionCommand().equals("Rcv(Srv)"))

{

Runnable guiSplit = () -> {

ci.Recieve(alRecs);

};

Thread lesGoo = new Thread(guiSplit);

lesGoo.start();

}

else if (e.getActionCommand().equals("Del(Srv)"))

{

int i = jtResults.getSelectedRow();

Runnable guiSplit = () -> {

if (ci.Delete(i) == 0)

{

dtResultsModel.removeRow(i);

alRecs.remove(i);

}

};

Thread lesGoo = new Thread(guiSplit);

lesGoo.start();

}

else throw new UnsupportedOperationException("Not supported yet."); // Unprocessed action case.

}

// запуск оконного приложения

public static void main(String args[]) {

new MyWin();

}

}

Lab1.ClientInstance.java

package lab1;

import Codes.Answer;

import Codes.Request;

import java.io.ByteArrayInputStream;

import java.io.ByteArrayOutputStream;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

import java.net.SocketException;

import java.net.UnknownHostException;

import javax.swing.JOptionPane;

import javax.swing.table.DefaultTableModel;

/\*\*

\*

\* @author User

\*/

public class ClientInstance {

public ClientInstance(DefaultTableModel \_dtTable)

{

inBuff = new byte[1024];

tableLock = new Object();

dtTable = \_dtTable;

bDisconnect = false;

port = 1234;

try

{

srvIp = InetAddress.getByName("IVDell808");

inSocket = new DatagramSocket(1235);

outSocket = new DatagramSocket();

}

catch(SocketException | UnknownHostException exc)

{

outSocket = null;

inSocket = null;

bDisconnect = true;

}

}

private DatagramSocket inSocket;

private DatagramSocket outSocket;

private final int port;

private final Object tableLock;

private byte[] inBuff;

private final DefaultTableModel dtTable;

private InetAddress srvIp;

boolean bDisconnect;

public int Connect() throws ClassNotFoundException

{

try

{

ByteArrayOutputStream outByteStream = new ByteArrayOutputStream();

ObjectOutputStream outStream = new ObjectOutputStream(outByteStream);

outStream.writeObject(Request.CONNECT);

DatagramPacket outData = new DatagramPacket(outByteStream.toByteArray(), outByteStream.size(), srvIp, port);

outSocket.send(outData);

}

catch(IOException exc)

{

return -1;

}

try

{

DatagramPacket inData = new DatagramPacket(inBuff, inBuff.length);

inSocket.receive(inData);

ObjectInputStream inStream = new ObjectInputStream(new ByteArrayInputStream(inBuff));

if((Answer)inStream.readObject() == Answer.SUCCESS)

{

JOptionPane.showMessageDialog(

null,

"SUCCESS",

"Code",

JOptionPane.INFORMATION\_MESSAGE

);

return 0;

}

}

catch(IOException exc)

{

return -1;

}

return -1;

}

public int Send()

{

int objCount = dtTable.getRowCount();

try

{

ByteArrayOutputStream outByteStream = new ByteArrayOutputStream();

ObjectOutputStream outStream = new ObjectOutputStream(outByteStream);

outStream.writeObject(Request.SEND);

outStream.writeInt(objCount);

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

{

double a = Double.parseDouble((String)dtTable.getValueAt(i, 0));

double b = Double.parseDouble((String)dtTable.getValueAt(i, 1));

double dx = Double.parseDouble((String)dtTable.getValueAt(i, 2));

double result = Double.parseDouble((String)dtTable.getValueAt(i, 3));

outStream.writeObject(new RecIntegral(a, b, dx, result));

}

DatagramPacket outData = new DatagramPacket(outByteStream.toByteArray(), outByteStream.size(), srvIp, port);

outSocket.send(outData);

}

catch(IntegralInputException | IOException exc)

{

return -1;

}

try

{

DatagramPacket inData = new DatagramPacket(inBuff, inBuff.length);

inSocket.receive(inData);

ObjectInputStream inStream = new ObjectInputStream(new ByteArrayInputStream(inBuff));

if((Answer)inStream.readObject() == Answer.SUCCESS)

{

JOptionPane.showMessageDialog(

null,

"SUCCESS",

"Code",

JOptionPane.INFORMATION\_MESSAGE

);

return 0;

}

}

catch(ClassNotFoundException | IOException exc)

{

return -1;

}

return -1;

}

public int Recieve(java.util.ArrayList<RecIntegral> alRecs)

{

try

{

ByteArrayOutputStream outByteStream = new ByteArrayOutputStream();

ObjectOutputStream outStream = new ObjectOutputStream(outByteStream);

outStream.writeObject(Request.RECIEVE);

DatagramPacket outData = new DatagramPacket(outByteStream.toByteArray(), outByteStream.size(), srvIp, port);

outSocket.send(outData);

}

catch(IOException exc)

{

return -1;

}

try

{

DatagramPacket inData = new DatagramPacket(inBuff, inBuff.length);

inSocket.receive(inData);

ObjectInputStream inStream = new ObjectInputStream(new ByteArrayInputStream(inBuff));

if((Answer)inStream.readObject() == Answer.SUCCESS)

{

int objCount = inStream.readInt();

if (objCount <= 0)

{

return -1;

}

dtTable.setRowCount(0);

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

{

RecIntegral tmp = (RecIntegral)inStream.readObject();

dtTable.addRow(new String[]{tmp.LimitL(), tmp.LimitR(), tmp.Dx(), tmp.Result()});

alRecs.add(tmp);

}

JOptionPane.showMessageDialog(

null,

"SUCCESS",

"Code",

JOptionPane.INFORMATION\_MESSAGE

);

return 0;

}

}

catch(ClassNotFoundException | IOException exc)

{

return -1;

}

return -1;

}

public int Delete(int \_index)

{

int i = \_index;

if (i == -1)

{

return -1;

}

try

{

ByteArrayOutputStream outByteStream = new ByteArrayOutputStream();

ObjectOutputStream outStream = new ObjectOutputStream(outByteStream);

outStream.writeObject(Request.DELETE);

outStream.writeInt(i);

outStream.writeObject(Request.DELETE);

DatagramPacket outData = new DatagramPacket(outByteStream.toByteArray(), outByteStream.size(), srvIp, port);

outSocket.send(outData);

}

catch(IOException exc)

{

return -1;

}

try

{

DatagramPacket inData = new DatagramPacket(inBuff, inBuff.length);

inSocket.receive(inData);

ObjectInputStream inStream = new ObjectInputStream(new ByteArrayInputStream(inBuff));

if((Answer)inStream.readObject() == Answer.SUCCESS)

{

JOptionPane.showMessageDialog(

null,

"SUCCESS",

"Code",

JOptionPane.INFORMATION\_MESSAGE

);

return 0;

}

}

catch(ClassNotFoundException | IOException exc)

{

return -1;

}

return -1;

}

}

Сервер:

Server.MyWin.java

package Server;

import java.awt.Container;

import java.awt.Dimension;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.ArrayList;

import javax.swing.JDialog;

import javax.swing.JFrame;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.table.DefaultTableModel;

public class MyWin extends JFrame implements ActionListener

{

// Some window's elements

private final DefaultTableModel dtResultsModel;

private final JTable jtResults;

private final JDialog jdFrame;

private final ServerRoutine thrSrvRoutine;

private final ArrayList alRecs;

// Classes serial number

private static final long serialVersionUID = 1L;

public MyWin()

{

// Initialization block

jdFrame = new JDialog();

alRecs = new ArrayList();

//----------------==Table Filling==------------------------------

Object[] columnNames = new String[]{"L limit", "R limit", "Step", "Result"};

Object[][] data = new String[][]{};

// Tables initialization

dtResultsModel = new DefaultTableModel(data, columnNames);

dtResultsModel.setColumnIdentifiers(columnNames);

jtResults = new JTable(dtResultsModel);

JScrollPane sp = new JScrollPane(jtResults);

thrSrvRoutine = new ServerRoutine(dtResultsModel);

Container c = getContentPane(); // Client rect

c.setLayout(null); // НЕНАВИЖУ КОМПАНОВЩИКИ

sp.setBounds(0, 0, 627, 480);

//----------------==Client rect's Filling==-----------------------

c.add(sp);

//----------------==Setting window's properties==-----------------

// Window's properties.

setTitle("Integral Server"); // Window's title. Obviously.

// Sets preffered size to window.

setPreferredSize(new Dimension(640, 480));

// Exit application on Exit button.

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

pack(); // Setting preffered sizes.

setVisible(true); // Makes window visible.

thrSrvRoutine.start();

}

@Override

public void actionPerformed(ActionEvent e) {

throw new UnsupportedOperationException("Not supported yet."); // Unprocessed action case.

}

// запуск оконного приложения

public static void main(String args[]) {

new MyWin();

}

}Server.ServerRoutine.java

package Server;

import java.io.IOException;

import java.net.DatagramSocket;

import javax.swing.table.DefaultTableModel;

/\*\*

\*

\* @author User

\*/

public class ServerRoutine extends Thread {

public ServerRoutine(DefaultTableModel \_dtTable)

{

try

{

sServer = new DatagramSocket(1234); // Port

for (int i = 0; i < 5; i++)

{

thrs[i] = new ServerInstance(sServer, \_dtTable);

}

}

catch(IOException exc)

{

exc.printStackTrace();

}

}

private DatagramSocket sServer = null;

ServerInstance[] thrs = new ServerInstance[5];

@Override

public void run()

{

for (int i = 0; i < 5; i++)

{

thrs[i].start();

}

for (int i = 0; i < 5; i++)

{

try

{

thrs[i].join();

}

catch(InterruptedException exc){}

}

}

}

Server.ServerInstance.java

package Server;

import Codes.Answer;

import Codes.Request;

import java.io.ByteArrayInputStream;

import java.io.ByteArrayOutputStream;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.net.DatagramSocket;

import java.net.DatagramPacket;

import java.net.InetAddress;

import java.net.SocketException;

import javax.swing.table.DefaultTableModel;

import lab1.RecIntegral;

import lab1.IntegralInputException;

/\*\*

\*

\* @author User

\*/

public class ServerInstance extends Thread {

public ServerInstance(DatagramSocket \_insocket, DefaultTableModel \_dtTable)

{

inSocket = \_insocket;

inBuff = new byte[1024];

inputLock = new Object();

outputLock = new Object();

tableLock = new Object();

dtTable = \_dtTable;

bDisconnect = false;

try

{

outSocket = new DatagramSocket();

}

catch(SocketException exc)

{

outSocket = null;

bDisconnect = true;

}

}

private final DatagramSocket inSocket;

private DatagramSocket outSocket;

private final Object inputLock;

private final Object outputLock;

private final Object tableLock;

private final byte[] inBuff;

private final DefaultTableModel dtTable;

boolean bDisconnect;

@Override

public void run()

{

while (!bDisconnect)

{

DatagramPacket inData = new DatagramPacket(inBuff, inBuff.length);

synchronized(inputLock)

{

try

{

inSocket.receive(inData);

}

catch(IOException exc)

{

bDisconnect = true;

continue;

}

}

try

{

InetAddress ClientIp = inData.getAddress();

int clientPort = inData.getPort();

ObjectInputStream inStream = new ObjectInputStream(new ByteArrayInputStream(inBuff));

Request code = (Request)inStream.readObject();

ByteArrayOutputStream outByteStream = new ByteArrayOutputStream();

ObjectOutputStream outStream = new ObjectOutputStream(outByteStream);

int objCount;

switch (code)

{

// CONNECT REQUEST

case Request.CONNECT:

outStream.writeObject(Answer.SUCCESS);

break;

// SEND REQUEST

case Request.SEND:

objCount = inStream.readInt();

if (objCount <= 0)

{

outStream.writeObject(Answer.ERROR);

break;

}

synchronized(tableLock)

{

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

{

RecIntegral tmp = (RecIntegral)inStream.readObject();

dtTable.addRow(new String[]{tmp.LimitL(), tmp.LimitR(), tmp.Dx(), tmp.Result()});

}

}

outStream.writeObject(Answer.SUCCESS);

break;

// RECIEVE REQUEST

case Request.RECIEVE:

objCount = dtTable.getRowCount();

outStream.writeObject(Answer.SUCCESS);

outStream.writeInt(objCount);

synchronized(tableLock){

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

{

double a = Double.parseDouble((String)dtTable.getValueAt(i, 0));

double b = Double.parseDouble((String)dtTable.getValueAt(i, 1));

double dx = Double.parseDouble((String)dtTable.getValueAt(i, 2));

double result = Double.parseDouble((String)dtTable.getValueAt(i, 3));

try

{

outStream.writeObject(new RecIntegral(a, b, dx, result));

}

catch(IntegralInputException exc)

{

outStream.flush();

outStream.writeObject(Answer.ERROR);

break;

}

}

}

break;

// DELETE REQUEST

case Request.DELETE:

int index = inStream.readInt();

if (index >= 0)

{

dtTable.removeRow(index);

outStream.writeObject(Answer.SUCCESS);

}

else

{

outStream.writeObject(Answer.INVALID);

}

break;

// ERROR CASE

default:

outStream.writeObject(Answer.ERROR);

break;

}

DatagramPacket outData = new DatagramPacket(outByteStream.toByteArray(), outByteStream.size(), ClientIp, 1235);

synchronized(outputLock)

{

outSocket.send(outData);

}

inStream.close();

outStream.flush();

}

catch(ClassNotFoundException | IOException exc)

{

exc.printStackTrace();

continue;

}

}

}

}

Codes.Request.java

package Codes;

import java.io.Serializable;

/\*\*

\*

\* @author User

\*/

public enum Request implements Serializable

{

CONNECT,

SEND,

RECIEVE,

DELETE

}

Codes.Answer.java

package Codes;

import java.io.Serializable;

/\*\*

\*

\* @author User

\*/

public enum Answer implements Serializable

{

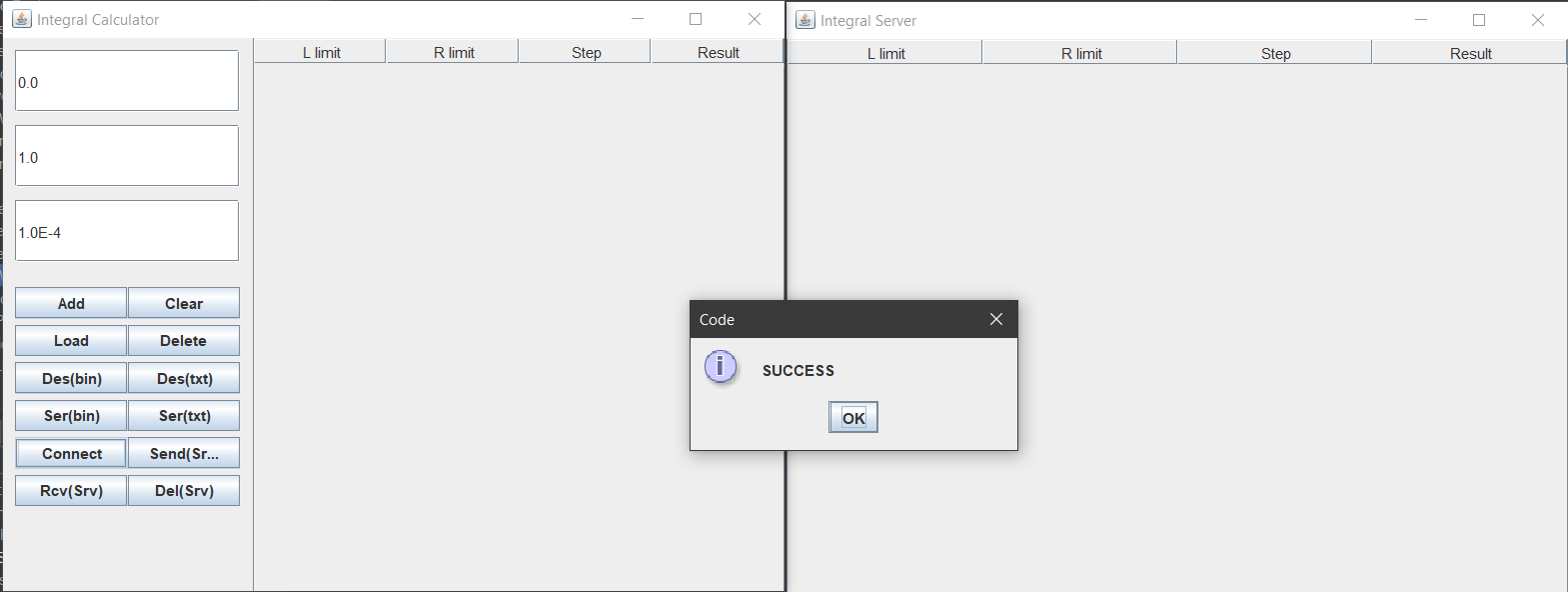
SUCCESS,

ERROR,

INVALID

}

**Результат работы программы**

Результаты работы программы представлены на рисунках 1-4. 

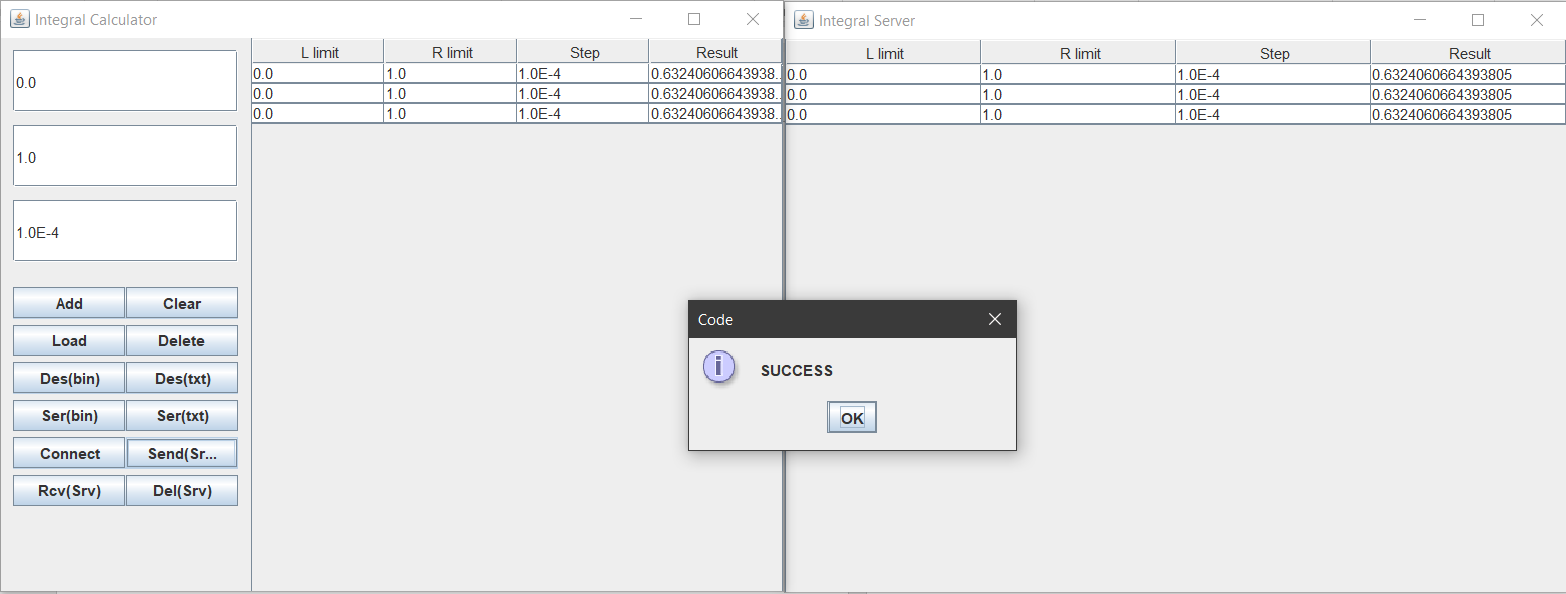
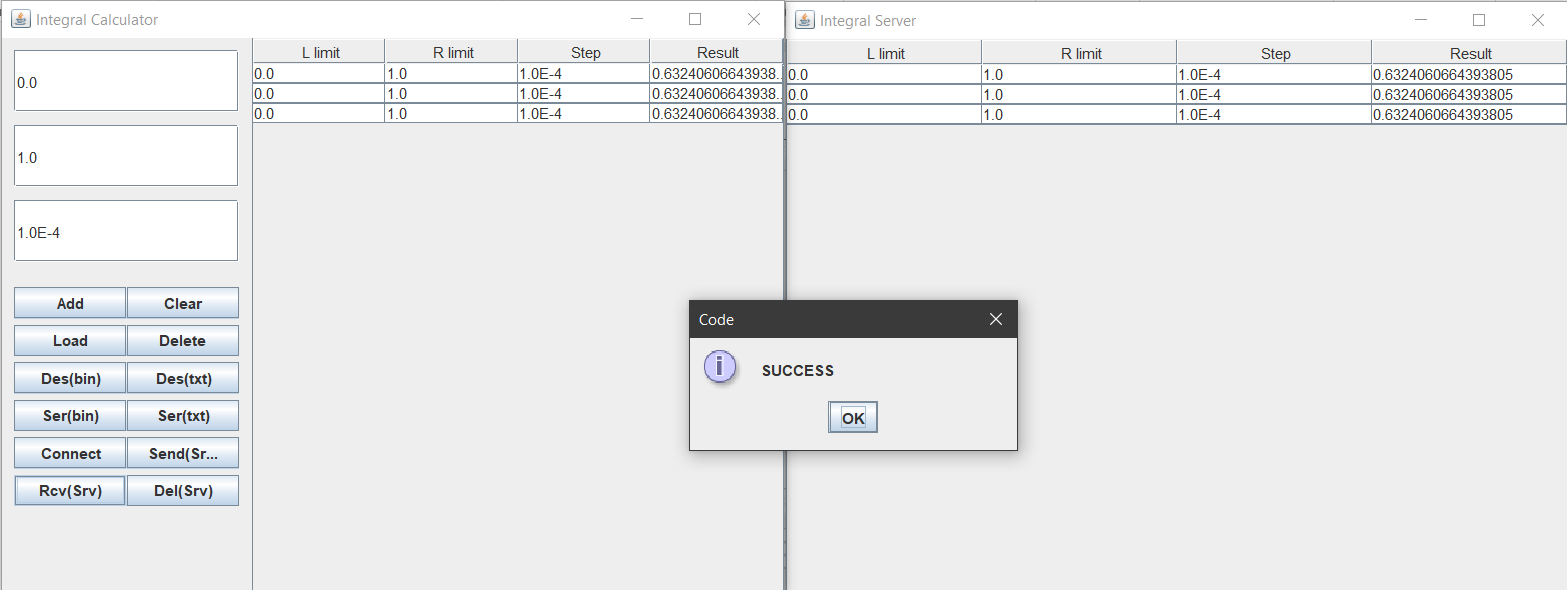
Рис.1 – Состояние программы при подключении

Рис.2 – Состояние программы после пересылки данных от клиента к серверу



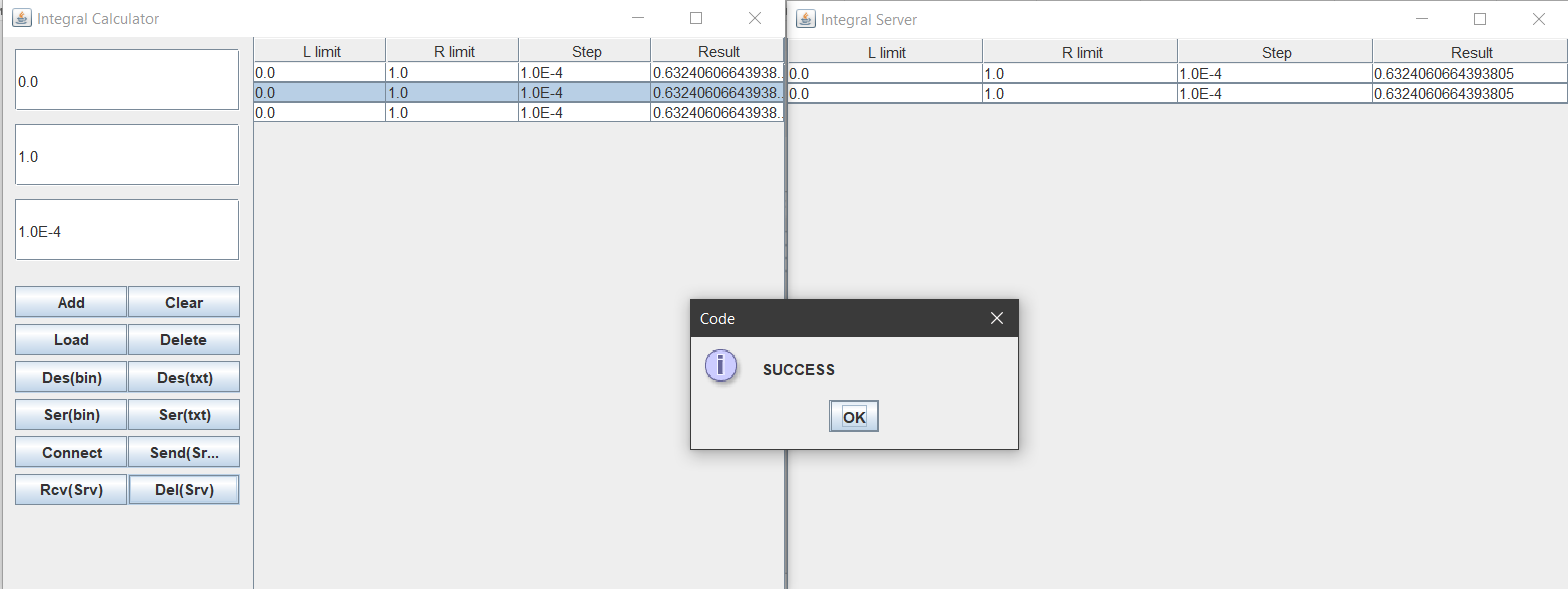
Рис.3 – Состояние программы после синхронизации данных клиента с сервером

Рис.4 – Состояние программы после удаления одной из строк таблицы на сервере через клиента

Результат программы совпал с ожидаемым, следовательно, программа составлена верно.

**Вывод**

В результате выполнения методических указаний к лабораторной работе были получены навыки построения клиент-серверного взаимодействия. Получены навыки синхронизации параллельных нитей в приложении, реализующем параллельное программирование.