Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе № 6

по дисциплине: «Программирование на языке Java»

на тему: «Сетевое взаимодействие в Java»

Выполнили:

студенты группы 20ВВП1

Макарова А. Ю.

Тельнова А. Д.

Приняли:

к.т.н., доцент

Юрова О. В.

к.т.н., доцент  
Карамышева Н.С.

Пенза, 2023

# Цель работы

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

# Лабораторное 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. Необходимо использовать протокол TCP.

# Описание работы программы

Вычисление определенного интеграла происходит на двух клиентах.

В методе CalcButtonActionPerformed() открыты два потока — на ввод и вывод, которые позволяют отправлять данные для вычисления интеграла и принимать результат вычислений. На сервере данные из таблицы преобразуются в строку для дальнейшей передачи клиентам. А также на сервер приходит результат вычислений и записывается в таблицу.

Для работы сервера используется класс серверного сокета — ServerSocket. Серверный сокет открыт на локальном компьютере и занимает порт 1777. Сервер входит в бесконечный цикл, в рамках которого происходит обработка результатов от клиентов. Первым шагом внутри цикла сервер переходит в режим ожидания соединения — вызов accept(). При приходе результатов от клиентского приложения метод возвращает объект Socket, который используется так же, как и клиентский сокет.

В клиенте устанавливается соединение с сервером. Аналогично серверу, на клиенте открыты два потока — на ввод и вывод, которые позволяют отправлять результат вычисления интеграла и принимать данные для вычислений. В цикле while() происходит чтение данных с сервера. В методе run() происходит расчет определенного интеграла. В этом же методе результаты вычислений отправляются на сервер.

# Листинг программы

Сервер

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.InputStreamReader;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.io.PrintWriter;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.Arrays;

import javax.swing.JOptionPane;

import javax.swing.table.DefaultTableModel;

import java.util.LinkedList;

import java.util.Vector;

import javax.swing.JFileChooser;

import javax.swing.filechooser.FileNameExtensionFilter;

public class JFrame extends javax.swing.JFrame {

public JFrame() {

initComponents();

socket = new MySocket();

socket.start();

}

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

jScrollPane1 = new javax.swing.JScrollPane();

Table = new javax.swing.JTable();

AddButton = new javax.swing.JButton();

DeleteButton = new javax.swing.JButton();

CalcButton = new javax.swing.JButton();

LowTextField = new javax.swing.JTextField();

UpTextField = new javax.swing.JTextField();

StepTextField = new javax.swing.JTextField();

LowLabel = new javax.swing.JLabel();

UpLabel = new javax.swing.JLabel();

StepLabel = new javax.swing.JLabel();

ClearButton = new javax.swing.JButton();

FillButton = new javax.swing.JButton();

SaveTxt = new javax.swing.JButton();

SaveBinary = new javax.swing.JButton();

LoadTxt = new javax.swing.JButton();

LoadBinary = new javax.swing.JButton();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

jScrollPane1.setBackground(new java.awt.Color(255, 255, 255));

Table.setModel(new javax.swing.table.DefaultTableModel(

new Object [][] {

},

new String [] {

"Нижняя граница", "Верхняя граница", "Шаг", "Результат"

}

) {

boolean[] canEdit = new boolean [] {

true, true, true, false

};

public boolean isCellEditable(int rowIndex, int columnIndex) {

return canEdit [columnIndex];

}

});

Table.setSelectionBackground(new java.awt.Color(255, 20, 147));

Table.setShowGrid(false);

Table.getTableHeader().setReorderingAllowed(false);

jScrollPane1.setViewportView(Table);

AddButton.setBackground(new java.awt.Color(255, 20, 147));

AddButton.setFont(new java.awt.Font("Segoe UI", 1, 18)); // NOI18N

AddButton.setForeground(new java.awt.Color(255, 255, 255));

AddButton.setText("Добавить");

AddButton.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

AddButtonActionPerformed(evt);

}

});

DeleteButton.setBackground(new java.awt.Color(255, 20, 147));

DeleteButton.setFont(new java.awt.Font("Segoe UI", 1, 18)); // NOI18N

DeleteButton.setForeground(new java.awt.Color(255, 255, 255));

DeleteButton.setText("Удалить");

DeleteButton.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

DeleteButtonActionPerformed(evt);

}

});

CalcButton.setBackground(new java.awt.Color(255, 20, 147));

CalcButton.setFont(new java.awt.Font("Segoe UI", 1, 18)); // NOI18N

CalcButton.setForeground(new java.awt.Color(255, 255, 255));

CalcButton.setText("Вычислить");

CalcButton.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

CalcButtonActionPerformed(evt);

}

});

LowTextField.setFont(new java.awt.Font("Segoe UI", 0, 18)); // NOI18N

UpTextField.setFont(new java.awt.Font("Segoe UI", 0, 18)); // NOI18N

StepTextField.setFont(new java.awt.Font("Segoe UI", 0, 18)); // NOI18N

LowLabel.setFont(new java.awt.Font("sansserif", 0, 24)); // NOI18N

LowLabel.setForeground(new java.awt.Color(204, 0, 102));

LowLabel.setText("Нижняя граница");

UpLabel.setFont(new java.awt.Font("sansserif", 0, 24)); // NOI18N

UpLabel.setForeground(new java.awt.Color(204, 0, 102));

UpLabel.setText("Верхняя граница");

StepLabel.setFont(new java.awt.Font("sansserif", 0, 24)); // NOI18N

StepLabel.setForeground(new java.awt.Color(204, 0, 102));

StepLabel.setText("Шаг");

ClearButton.setBackground(new java.awt.Color(255, 20, 147));

ClearButton.setFont(new java.awt.Font("Segoe UI", 1, 18)); // NOI18N

ClearButton.setForeground(new java.awt.Color(255, 255, 255));

ClearButton.setText("Очистить таблицу");

ClearButton.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

ClearButtonActionPerformed(evt);

}

});

FillButton.setBackground(new java.awt.Color(255, 20, 147));

FillButton.setFont(new java.awt.Font("Segoe UI", 1, 18)); // NOI18N

FillButton.setForeground(new java.awt.Color(255, 255, 255));

FillButton.setText("Заполнить таблицу");

FillButton.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

FillButtonActionPerformed(evt);

}

});

SaveTxt.setBackground(new java.awt.Color(255, 20, 147));

SaveTxt.setFont(new java.awt.Font("Segoe UI", 1, 18)); // NOI18N

SaveTxt.setForeground(new java.awt.Color(255, 255, 255));

SaveTxt.setText("Сохранить в текстовом виде");

SaveTxt.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

SaveTxtActionPerformed(evt);

}

});

SaveBinary.setBackground(new java.awt.Color(255, 20, 147));

SaveBinary.setFont(new java.awt.Font("Segoe UI", 1, 18)); // NOI18N

SaveBinary.setForeground(new java.awt.Color(255, 255, 255));

SaveBinary.setText("Сохранить в двоичном виде");

SaveBinary.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

SaveBinaryActionPerformed(evt);

}

});

LoadTxt.setBackground(new java.awt.Color(255, 20, 147));

LoadTxt.setFont(new java.awt.Font("Segoe UI", 1, 18)); // NOI18N

LoadTxt.setForeground(new java.awt.Color(255, 255, 255));

LoadTxt.setText("Загрузить в текстовом виде");

LoadTxt.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

LoadTxtActionPerformed(evt);

}

});

LoadBinary.setBackground(new java.awt.Color(255, 20, 147));

LoadBinary.setFont(new java.awt.Font("Segoe UI", 1, 18)); // NOI18N

LoadBinary.setForeground(new java.awt.Color(255, 255, 255));

LoadBinary.setText("Загрузить в двоичном виде");

LoadBinary.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

LoadBinaryActionPerformed(evt);

}

});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

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

.addGroup(layout.createSequentialGroup()

.addGap(0, 43, Short.MAX\_VALUE)

.addComponent(LowLabel)

.addGap(18, 18, 18)

.addComponent(LowTextField, javax.swing.GroupLayout.PREFERRED\_SIZE, 350, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(layout.createSequentialGroup()

.addGap(26, 26, 26)

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

.addComponent(jScrollPane1)

.addGroup(layout.createSequentialGroup()

.addComponent(AddButton, javax.swing.GroupLayout.PREFERRED\_SIZE, 150, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(58, 58, 58)

.addComponent(DeleteButton, javax.swing.GroupLayout.PREFERRED\_SIZE, 150, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(CalcButton, javax.swing.GroupLayout.PREFERRED\_SIZE, 150, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()

.addGap(0, 9, Short.MAX\_VALUE)

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

.addComponent(UpLabel, javax.swing.GroupLayout.Alignment.TRAILING)

.addComponent(StepLabel, javax.swing.GroupLayout.Alignment.TRAILING))

.addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)

.addComponent(UpTextField, javax.swing.GroupLayout.DEFAULT\_SIZE, 350, Short.MAX\_VALUE)

.addComponent(StepTextField)))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(SaveTxt, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(SaveBinary, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(ClearButton, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)

.addComponent(LoadTxt, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(LoadBinary, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(FillButton, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))))))

.addGap(24, 24, 24))

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(14, 14, 14)

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

.addComponent(LowTextField, javax.swing.GroupLayout.DEFAULT\_SIZE, 40, Short.MAX\_VALUE)

.addComponent(LowLabel))

.addGap(18, 18, 18)

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

.addComponent(UpTextField, javax.swing.GroupLayout.PREFERRED\_SIZE, 40, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(UpLabel))

.addGap(18, 18, 18)

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

.addComponent(StepTextField, javax.swing.GroupLayout.PREFERRED\_SIZE, 40, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(StepLabel))

.addGap(33, 33, 33)

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

.addComponent(AddButton, javax.swing.GroupLayout.PREFERRED\_SIZE, 50, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(DeleteButton, javax.swing.GroupLayout.PREFERRED\_SIZE, 50, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(CalcButton, javax.swing.GroupLayout.PREFERRED\_SIZE, 50, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(37, 37, 37)

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 230, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(18, 18, 18)

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

.addComponent(ClearButton, javax.swing.GroupLayout.PREFERRED\_SIZE, 50, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(FillButton, javax.swing.GroupLayout.PREFERRED\_SIZE, 50, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(LoadTxt, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(SaveTxt, javax.swing.GroupLayout.DEFAULT\_SIZE, 50, Short.MAX\_VALUE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(SaveBinary, javax.swing.GroupLayout.DEFAULT\_SIZE, 50, Short.MAX\_VALUE)

.addComponent(LoadBinary, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

.addGap(26, 26, 26))

);

pack();

}// </editor-fold>

LinkedList<RecIntegral> data = new LinkedList<>();

MySocket socket;

private void AddButtonActionPerformed(java.awt.event.ActionEvent evt) {

if (LowTextField.getText().equals("") || UpTextField.getText().equals("") || StepTextField.getText().equals("")) {

JOptionPane.showMessageDialog(this, "Заполните все поля данными!", "Ошибочка вышла!", JOptionPane.PLAIN\_MESSAGE);

}

String l = LowTextField.getText();

String u = UpTextField.getText();

String s = StepTextField.getText();

String r = null;

try {

if (Double.valueOf(l) > 1000000 || Double.valueOf(l) < 0.000001) {

throw new SomeProblems("Неверное значение нижнего предела!");

} else if (Double.valueOf(u) > 1000000 || Double.valueOf(u) < 0.000001) {

throw new SomeProblems("Неверное значение верхнего предела!");

} else if (Double.valueOf(s) > 1000000 || Double.valueOf(s) < 0.000001) {

throw new SomeProblems("Неверное значение шага интегрирования!");

} else if (Double.valueOf(l) > Double.valueOf(u) || Double.valueOf(s) == 0 || Double.valueOf(s) > Double.valueOf(u) - Double.valueOf(l)) {

throw new SomeProblems("Что-то не то с данными!");

}

} catch (SomeProblems e) {

LowTextField.setText("");

UpTextField.setText("");

StepTextField.setText("");

return;

}

DefaultTableModel model = (DefaultTableModel) Table.getModel();

model.addRow(new Object[]{LowTextField.getText(), UpTextField.getText(), StepTextField.getText()});

RecIntegral object = new RecIntegral();

object.NewNode(l, u, s, r);

data.add(object);

LowTextField.setText("");

UpTextField.setText("");

StepTextField.setText("");

}

private void DeleteButtonActionPerformed(java.awt.event.ActionEvent evt) {

DefaultTableModel model = (DefaultTableModel) Table.getModel();

int SelectedRow = Table.getSelectedRow();

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

model.removeRow(Table.getSelectedRow());

data.remove(SelectedRow);

} else

JOptionPane.showMessageDialog(this, "Выберите строку, которую нужно удалить!", "Ошибочка вышла!", JOptionPane.PLAIN\_MESSAGE);

}

private void CalcButtonActionPerformed(java.awt.event.ActionEvent evt) {

DefaultTableModel model = (DefaultTableModel) Table.getModel();

Vector data = model.getDataVector();

int countRows = model.getRowCount();

//'этот цикл нужен для того, чтобы оценить данные на корректность

for (int j = 0; j < data.size(); j++) {

double low = Double.parseDouble((String) Table.getValueAt(j, 0));

double up = Double.parseDouble((String) Table.getValueAt(j, 1));

double step = Double.parseDouble((String) Table.getValueAt(j, 2));

//оценка данных на корректность

try {

if (low > up || step == 0 || low > 1000000 || low < 0.000001 || up > 1000000 || up < 0.000001 || step > 1000000 || step < 0.000001 || step > up - low) {

throw new SomeProblems("Вы некорректно изменили данные в таблице!");

}

} catch (SomeProblems e) {

return;

}

}

String message[] = new String[data.size()];

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

Vector CurrentData = (Vector) data.get(i);

message[i] = String.valueOf(CurrentData.get(0)) + " " + String.valueOf(CurrentData.get(1)) + " " + String.valueOf(CurrentData.get(2)) + " " + String.valueOf(i);

System.out.println("Отправляем на клиент " + message[i]);

}

Socket fromClientSocket[] = new Socket[]{socket.getFromClientSocket(0), socket.getFromClientSocket(1)};

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

try {

Socket localSocket = fromClientSocket[i % 2];

PrintWriter pw = new PrintWriter(localSocket.getOutputStream(), true);

BufferedReader br = new BufferedReader(new InputStreamReader(localSocket.getInputStream()));

pw.println(message [i]);

String str = br.readLine();

Double[] num = Arrays.stream(str.split(" ")).map(Double::parseDouble).toArray(Double[]::new);

model.setValueAt(num[0], num[1].intValue(), 3);

} catch (IOException ex) {

ex.printStackTrace(System.out);

}

}

}

private void FillButtonActionPerformed(java.awt.event.ActionEvent evt) {

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

if (!data.isEmpty()) {

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

RecIntegral object = data.get(i);

module.addRow(new Object[]{object.low, object.up, object.step, object.result});

}

} else

JOptionPane.showMessageDialog(this, "Linked List пуст!", "Ошибочка вышла!", JOptionPane.PLAIN\_MESSAGE);

}

private void ClearButtonActionPerformed(java.awt.event.ActionEvent evt) {

DefaultTableModel model = (DefaultTableModel) Table.getModel();

if (Table.getRowCount() != 0) {

model.setRowCount(0);

} else

JOptionPane.showMessageDialog(this, "Таблица и так пустая!", "Ошибочка вышла!", JOptionPane.PLAIN\_MESSAGE);

}

private void SaveTxtActionPerformed(java.awt.event.ActionEvent evt) {

DefaultTableModel model = (DefaultTableModel) Table.getModel();

JFileChooser choosedir = new JFileChooser();

choosedir.setDialogTitle("Сохранить файл в текстовом виде");

choosedir.setFileFilter(new FileNameExtensionFilter("Текстовые документы (\*.txt)", "txt"));

int result = choosedir.showSaveDialog(this);

//File txtfile = choosedir.getSelectedFile();

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

File txtfile = new File(choosedir.getSelectedFile() + ".txt");

try {

int columns = 4;

int rows = Table.getRowCount();

FileWriter fwrite = new FileWriter(txtfile);

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

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

fwrite.write(model.getValueAt(i, j).toString());

fwrite.write(" ");

}

fwrite.write("\n");

}

fwrite.close();

JOptionPane.showMessageDialog(this, "Файл сохранен!", "Уведомление", JOptionPane.PLAIN\_MESSAGE);

} catch (IOException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(this, "Произошла ошибка при сохранении файла!", "Уведомление", JOptionPane.PLAIN\_MESSAGE);

}

} else if (result == JFileChooser.CANCEL\_OPTION)

System.out.println("Сохранение файла отменено");

}

private void SaveBinaryActionPerformed(java.awt.event.ActionEvent evt) {

DefaultTableModel model = (DefaultTableModel) Table.getModel();

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

JFileChooser choosedir = new JFileChooser();

choosedir.setDialogTitle("Сохранить файл в двоичном виде");

choosedir.setFileFilter(new FileNameExtensionFilter("Двоичные файлы (\*.bin)", "bin"));

int result = choosedir.showSaveDialog(this);

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

File txtfile = new File(choosedir.getSelectedFile() + ".bin");

int columns = 4;

int rows = Table.getRowCount();

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

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

bindata.add(model.getValueAt(i, j).toString());

}

}

try (ObjectOutputStream obj = new ObjectOutputStream(new BufferedOutputStream(new FileOutputStream(txtfile)))) {

obj.writeObject(bindata);

obj.close();

JOptionPane.showMessageDialog(this, "Файл сохранен!", "Уведомление", JOptionPane.PLAIN\_MESSAGE);

} catch (IOException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(this, "Произошла ошибка при сохранении файла!", "Уведомление", JOptionPane.PLAIN\_MESSAGE);

}

} else if (result == JFileChooser.CANCEL\_OPTION)

System.out.println("Сохранение файла отменено");

}

private void LoadTxtActionPerformed(java.awt.event.ActionEvent evt) {

DefaultTableModel model = (DefaultTableModel) Table.getModel();

JFileChooser choosedir = new JFileChooser();

choosedir.setDialogTitle("Загрузить файл в текстовом виде");

choosedir.setFileFilter(new FileNameExtensionFilter("Текстовые документы (\*.txt)", "txt"));

int result = choosedir.showOpenDialog(this);

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

File txtfile = choosedir.getSelectedFile();

try {

FileReader fread = new FileReader(txtfile);

BufferedReader buff = new BufferedReader(fread);

String line;

String[] split;

data.clear();

if (model.getRowCount() != 0) {

model.setRowCount(0);

}

while ((line = buff.readLine()) != null) {

split = line.split(" ");

model.addRow(new Object[]{split[0], split[1], split[2], split[3]});

RecIntegral val = new RecIntegral();

val.NewNode(split[0], split[1], split[2], split[3]);

data.add(val);

}

buff.close();

fread.close();

JOptionPane.showMessageDialog(this, "Файл загружен!", "Уведомление", JOptionPane.PLAIN\_MESSAGE);

} catch (IOException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(this, "Произошла ошибка при загрузке файла!", "Уведомление", JOptionPane.PLAIN\_MESSAGE);

}

} else if (result == JFileChooser.CANCEL\_OPTION)

System.out.println("Открытие файла отменено");

}

private void LoadBinaryActionPerformed(java.awt.event.ActionEvent evt) {

DefaultTableModel model = (DefaultTableModel) Table.getModel();

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

JFileChooser choosedir = new JFileChooser();

choosedir.setDialogTitle("Загрузить файл в двоичном виде");

choosedir.setFileFilter(new FileNameExtensionFilter("Двоичные файлы (\*.bin)", "bin"));

int result = choosedir.showOpenDialog(this);

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

File txtfile = choosedir.getSelectedFile();

try (ObjectInputStream obj = new ObjectInputStream(new BufferedInputStream(new FileInputStream(txtfile)))) {

bindata = (LinkedList<String>) obj.readObject();

obj.close();

data.clear();

if (model.getRowCount() != 0) {

model.setRowCount(0);

}

for (int i = 0; i < bindata.size(); i += 4) {

RecIntegral integral = new RecIntegral();

integral.NewNode(bindata.get(i), bindata.get(i + 1), bindata.get(i + 2), bindata.get(i + 3));

model.addRow(new Object[]{bindata.get(i), bindata.get(i + 1), bindata.get(i + 2), bindata.get(i + 3)});

data.add(integral);

}

JOptionPane.showMessageDialog(this, "Файл загружен!", "Уведомление", JOptionPane.PLAIN\_MESSAGE);

} catch (IOException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(this, "Произошла ошибка при загрузке файла!", "Уведомление", JOptionPane.PLAIN\_MESSAGE);

} catch (ClassNotFoundException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(this, "Произошла ошибка при загрузке файла!", "Уведомление", JOptionPane.PLAIN\_MESSAGE);

}

} else if (result == JFileChooser.CANCEL\_OPTION)

System.out.println("Открытие файла отменено");

}

public class RecIntegral {

public String low;

public String up;

public String step;

public String result;

public void NewNode(String low, String up, String step, String result) {

this.low = low;

this.up = up;

this.step = step;

this.result = result;

}

}

static class SomeProblems extends Exception {

public SomeProblems(String message) {

JOptionPane.showMessageDialog(null, message, "Ошибочка вышла!", JOptionPane.PLAIN\_MESSAGE);

}

}

public class TwoThreads extends Thread {

public DefaultTableModel model;

public String name;

public TwoThreads(DefaultTableModel model, String name) {

this.model = model;

this.name = name;

}

}

public class MySocket extends Thread {

private ServerSocket servSocket;

private Socket fromClientSocket [] = new Socket [2];

public Socket getFromClientSocket(int num) {

return fromClientSocket [num];

}

public MySocket() {

}

public void run() {

int port = 1777;

try {

// Открыть серверный сокет (ServerSocket)

// Это специальный класс для сетевого взаимодействия с серверной стороны

servSocket = new ServerSocket(port);

// Входим в бесконечный цикл - ожидаем соединения

while (true) {

System.out.println("Ожидаем подключение от порта " + port);

// Получив соединение начинаем работать с сокетом

fromClientSocket [0] = servSocket.accept();

fromClientSocket [1] = servSocket.accept();

}

} catch (IOException ex) {

ex.printStackTrace(System.out);

}

}

}

public static void main(String args[]) {

//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

/\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.

\* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

\*/

try {

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break;

}

}

} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(JFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(JFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(JFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(JFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

//</editor-fold>

java.awt.EventQueue.invokeLater(new Runnable() {

@Override

public void run() {

JFrame frame = new JFrame();

frame.setVisible(true);

frame.setLocationRelativeTo(null);

frame.setTitle("Вычисляем интегральчик");

frame.getContentPane().setBackground(new java.awt.Color(255, 228, 225));

}

});

}

// Variables declaration - do not modify

private javax.swing.JButton AddButton;

private javax.swing.JButton CalcButton;

private javax.swing.JButton ClearButton;

private javax.swing.JButton DeleteButton;

private javax.swing.JButton FillButton;

private javax.swing.JButton LoadBinary;

private javax.swing.JButton LoadTxt;

private javax.swing.JLabel LowLabel;

private javax.swing.JTextField LowTextField;

private javax.swing.JButton SaveBinary;

private javax.swing.JButton SaveTxt;

private javax.swing.JLabel StepLabel;

private javax.swing.JTextField StepTextField;

private javax.swing.JTable Table;

private javax.swing.JLabel UpLabel;

private javax.swing.JTextField UpTextField;

private javax.swing.JScrollPane jScrollPane1;

// End of variables declaration

}

Клиент

import java.io.\*;

import java.net.\*;

import java.util.Arrays;

class TwoThreads extends Thread {

double Low;

double Up;

double Step;

int i;

PrintWriter pw;

//double result;

public TwoThreads(double Low, double Up, double Step, int i, PrintWriter pw) {

this.Up = Up;

this.Low = Low;

this.Step = Step;

this.i = i;

this.pw = pw;

}

@Override

public void run() {

double IntegralFunc = 0;

for (double i = Low; i < Up - Step; i += Step) {

if (Up - Low > Step) {

IntegralFunc += ((Math.sin(i) + Math.sin(i + Step)) / 2) \* Step;

} else {

IntegralFunc += ((Math.sin(Up) + Math.sin(i - Step)) / 2) \* Step;

}

}

String message = String.valueOf(IntegralFunc) + " " + i;

pw.println(message);

System.out.println("Отправляем на сервер " + message);

}

}

public class Client {

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

// Определяем номер порта, на котором нас ожидает сервер для ответа

int portNumber = 1777;

System.out.println("Клиент запущен");

// Открыть сокет (Socket) для обращения к локальному компьютеру

// Сервер мы будем запускать на этом же компьютере

// Это специальный класс для сетевого взаимодействия c клиентской стороны

Socket socket = new Socket("127.0.0.1", portNumber);

// Создать поток для чтения символов из сокета

// Для этого надо открыть поток сокета - socket.getInputStream()

// Потом преобразовать его в поток символов - new InputStreamReader

// И уже потом сделать его читателем строк - BufferedReader

BufferedReader br = new BufferedReader(new InputStreamReader(socket.getInputStream()));

// Создать поток для записи символов в сокет

PrintWriter pw = new PrintWriter(socket.getOutputStream(), true);

String str = " ";

//!"bye".equals(str = br.readLine())

// Входим в цикл чтения, что нам ответил сервер

while ((str = br.readLine()) != null) {

Double[] num = Arrays.stream(str.split(" ")).map(Double::parseDouble).toArray(Double[]::new);

TwoThreads threadthird = new TwoThreads(num[0], num[1], num[2], num[3].intValue(), pw);

threadthird.start();

}

br.close();

pw.close();

socket.close();

}

}

# Результат работы программы

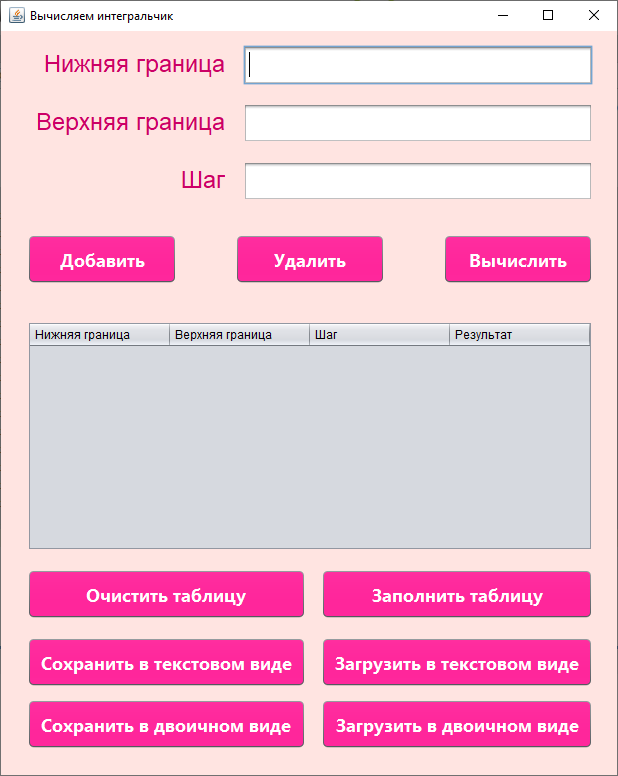


Рисунок – Результат запуска программы

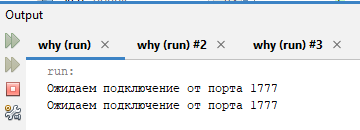


Рисунок – Вывод сообщений в консоль на сервере

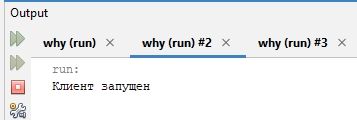


Рисунок – Вывод сообщений в консоль на первом клиенте

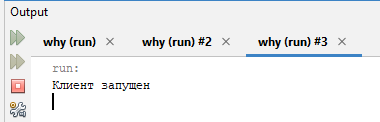


Рисунок – Вывод сообщений в консоль на втором клиенте

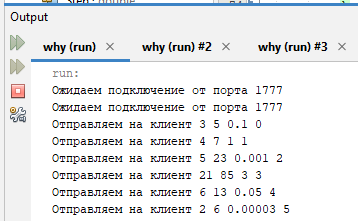


Рисунок – Вывод сообщений в консоль об отправке данных на клиенты

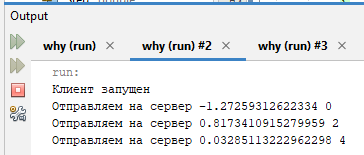


Рисунок – Вывод сообщений в консоль об отправке данных на сервер с первого клиента

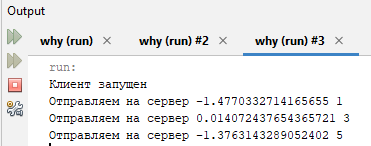


Рисунок – Вывод сообщений в консоль об отправке данных на сервер со второго клиента

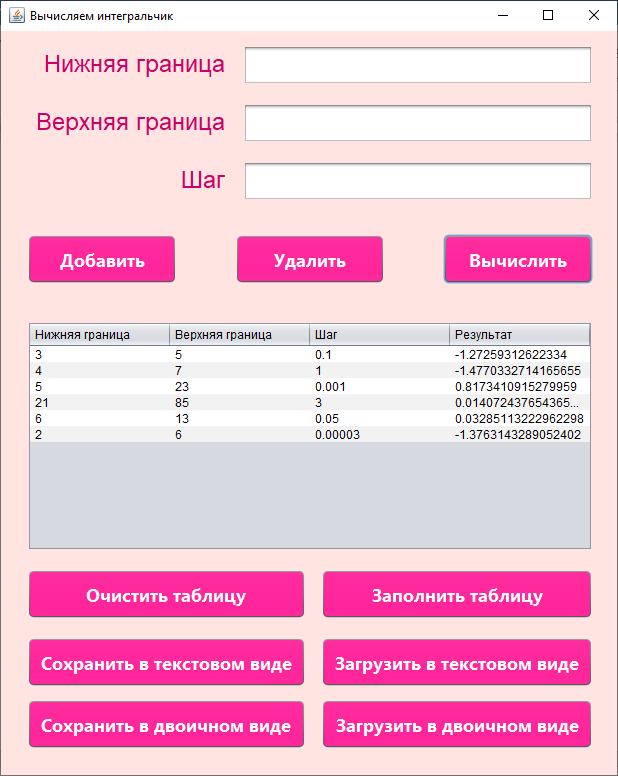


Рисунок – Результат расчета определенных интегралов

# Вывод

Получен опыт в создании клиент-серверных приложений c использованием стандартных классов Java.