Пензенский государственный университет

Факультет вычислительной техники

Кафедра «Вычислительная техника»

Сетевое взаимодействие в Java

Отчет о лабораторной работе № 6

по дисциплине «Разработка кроссплатформенных приложений»

Вариант № 2

Выполнили: ст-ты гр. 21ВОЭ1

Кокорин А. А.

Тишаев М. Р.

Приняли: к.т.н., доцент каф. ВТ

Юрова О. В.

к.т.н., доцент каф. ВТ

Карамышева Н. С.

2024

1 Цель работы

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

2 Формулировка задания

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. Использовать TCP.

3 Описание программы

Программа построена на основе лабораторной работы № 5.

Текст программы приведен в приложении А.

В программу был добавлен класс *Server* для вычисления интеграла с помощью разбиения потоков в клиент-серверном приложении с использованием стандартных классов Java.

В классе *Server* были описаны:

ServerSocket(9999)- Создаем серверный сокет, который прослушивает порт 9999

serverSocket.accept(); Принимаем соединение от клиента

new BufferedReader - Создаем потоки для обмена данными с клиентом

Читаем и отправляем данные клиенту в бесконечном цикле.

clientSocket.close(); - Закрываем соединение с клиентом

Модифицирована часть NewFrame :

private Socket socket; - Добавлена глобальная переменная сокета

private PrintWriter out; - Добавлена глобальная переменная вывода

private BufferedReader in; - Добавлена глобальная переменная ввода

connectToServer()- Создаем функцию , которая будет подключаться к серверу с указанным портом и создавать потоки для обмена данными с сервером

jButtonFindActionPerformed – Основная часть изменений встроена в кнопку вычислить

out.println(Double.toString(res));- Отправляем данные на сервер

String response = in.readLine(); - Ждем ответа от сервера

System.out.println(response); - Ждем ответа от сервера

jTable1.setValueAt(response, row, 3); - Ждем ответа от сервера

Создание серверного сокета:

Серверный сокет создается и начинает прослушивать соединения на порту 9999. Этот сокет будет ожидать входящих подключений от клиентов. Сервер входит в бесконечный цикл, ожидая подключения клиентов. Когда клиент подключается, сервер принимает это подключение с помощью метода *accept()*, который создает новый сокет *clientSocket* для взаимодействия с клиентом. При каждом подключении выводится сообщение о подключении клиента. Создаются потоки для обмена данными. *BufferedReader* для чтения данных, полученных от клиента. *PrintWriter* для отправки данных клиенту. С помощью внутреннего цикла сервер продолжает читать данные от клиента до тех пор, пока клиент не прекратит соединение. Когда клиент отправляет сообщение, сервер читает это сообщение через *in.readLine()*. Проверяет, не является ли сообщение *null* (если *null*, это означает, что клиент закрыл соединение, и цикл прерывается). Выводит полученные данные на консоль. Отправляет полученные данные обратно клиенту через *out.println(dataFromClient)*.После завершения общения с клиентом (например, когда клиент закрыл соединение), сервер закрывает сокет *clientSocket*. Вся логика сервера обернута в блок *try-catch* для обработки возможных исключений, таких как проблемы с созданием сокетов или потока ввода/вывода.

4 Программа и методика испытаний

Для проверки правильности работы программы подготовлен вручную тестовый набор данных в таблице 1.

Таблица 1 – Тестовый набор данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a | b | h | Результат работы программы | Результат ручного просчета |
| 2 | 4 | 0,2 | 0,1047 | 0,1047 |
| 1 | 4 | 0,3 | 0,1526 | 0,1526 |
| 100 | 100000 | 0,01 | -19,9084 | -19,9084 |
| 24 | 200 | 0,02 | 106,2013 | 106,2013 |

При работе программы с тестовыми данными получены результаты, приведенные в Приложении Б. Результаты, приведенные в таблице 1, совпадают с результатами работы программы на рисунке Б.1. Таким образом, можно сделать вывод, что программа работает правильно.

Вывод

Результатом выполнения данной лабораторной работы является изучение создания клиент-серверного приложения c использованием стандартных классов Java.

ТЕКСТ ПРОГРАММЫ

Приложение А

(обязательное)

Класс NewJFrame

package my.javaapplication1;

import javax.swing.table.DefaultTableModel;

import java.util.LinkedList;

import java.io.\*;

import java.net.Socket;

import java.util.Scanner;

import java.util.logging.Level;

import java.util.logging.Logger;

public class NewJFrame extends javax.swing.JFrame {

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

private Socket socket;

private PrintWriter out;

private BufferedReader in;

public NewJFrame() {

initComponents();

connectToServer();

}

@SuppressWarnings("unchecked")

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

private void initComponents() {

jDialog1 = new javax.swing.JDialog();

jLabelError = new javax.swing.JLabel();

jFileChooser1 = new javax.swing.JFileChooser();

jButtonAdd = new javax.swing.JButton();

jButtonDel = new javax.swing.JButton();

jButtonFind = new javax.swing.JButton();

jTextFieldLow = new javax.swing.JTextField();

jTextFieldTop = new javax.swing.JTextField();

jTextFieldStep = new javax.swing.JTextField();

jScrollPane1 = new javax.swing.JScrollPane();

jTable1 = new javax.swing.JTable();

jLabel1 = new javax.swing.JLabel();

jLabel2 = new javax.swing.JLabel();

jLabel3 = new javax.swing.JLabel();

jButtonClear = new javax.swing.JButton();

jButtonFill = new javax.swing.JButton();

jButtonFileWriterText = new javax.swing.JButton();

jButtonFileReaderText = new javax.swing.JButton();

jButtonFileWriterBin = new javax.swing.JButton();

jButtonFileReaderBin = new javax.swing.JButton();

jDialog1.setTitle("Ошибка ввода");

jLabelError.setText("jLabel4");

javax.swing.GroupLayout jDialog1Layout = new javax.swing.GroupLayout(jDialog1.getContentPane());

jDialog1.getContentPane().setLayout(jDialog1Layout);

jDialog1Layout.setHorizontalGroup(

jDialog1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

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

.addGroup(jDialog1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jDialog1Layout.createSequentialGroup()

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

.addComponent(jLabelError)

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

);

jDialog1Layout.setVerticalGroup(

jDialog1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

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

.addGroup(jDialog1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jDialog1Layout.createSequentialGroup()

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

.addComponent(jLabelError)

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

);

jFileChooser1.setDialogType(javax.swing.JFileChooser.CUSTOM\_DIALOG);

jFileChooser1.setApproveButtonToolTipText("");

jFileChooser1.setDialogTitle("");

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

setTitle("Lab5");

jButtonAdd.setText("Добавить");

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

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

jButtonAddActionPerformed(evt);

}

});

jButtonDel.setText("Удалить");

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

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

jButtonDelActionPerformed(evt);

}

});

jButtonFind.setText("Вычислить");

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

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

jButtonFindActionPerformed(evt);

}

});

jTable1.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];

}

});

jTable1.getTableHeader().setReorderingAllowed(false);

jScrollPane1.setViewportView(jTable1);

if (jTable1.getColumnModel().getColumnCount() > 0) {

jTable1.getColumnModel().getColumn(0).setResizable(false);

jTable1.getColumnModel().getColumn(1).setResizable(false);

jTable1.getColumnModel().getColumn(2).setResizable(false);

jTable1.getColumnModel().getColumn(3).setResizable(false);

}

jLabel1.setText("Нижняя граница");

jLabel2.setText("Верхняя граница");

jLabel3.setText("Шаг");

jButtonClear.setText("Очистить");

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

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

jButtonClearActionPerformed(evt);

}

});

jButtonFill.setText("Заполнить");

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

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

jButtonFillActionPerformed(evt);

}

});

jButtonFileWriterText.setText("Сохранение в текстовом виде");

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

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

jButtonFileWriterTextActionPerformed(evt);

}

});

jButtonFileReaderText.setText("Загрузка в текстовом виде");

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

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

jButtonFileReaderTextActionPerformed(evt);

}

});

jButtonFileWriterBin.setText("Сохранение в двоичном виде");

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

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

jButtonFileWriterBinActionPerformed(evt);

}

});

jButtonFileReaderBin.setText("Загрузка в доичном виде");

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

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

jButtonFileReaderBinActionPerformed(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()

.addContainerGap()

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

.addComponent(jScrollPane1)

.addGroup(layout.createSequentialGroup()

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

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

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

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

.addGap(30, 30, 30)

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

.addGroup(layout.createSequentialGroup()

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

.addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT\_SIZE, 266, Short.MAX\_VALUE)

.addComponent(jTextFieldTop)

.addComponent(jTextFieldStep)

.addGroup(layout.createSequentialGroup()

.addComponent(jLabel3)

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

.addComponent(jTextFieldLow))

.addGap(32, 32, 32)

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

.addComponent(jButtonFill, javax.swing.GroupLayout.DEFAULT\_SIZE, 100, Short.MAX\_VALUE)

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

.addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 108, javax.swing.GroupLayout.PREFERRED\_SIZE))

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

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

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

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

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

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

.addContainerGap())

);

layout.setVerticalGroup(

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

.addGroup(layout.createSequentialGroup()

.addContainerGap()

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

.addGroup(layout.createSequentialGroup()

.addComponent(jLabel1)

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

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

.addComponent(jButtonAdd)

.addComponent(jTextFieldLow, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jButtonFill))

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

.addComponent(jLabel2)

.addGap(6, 6, 6)

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

.addComponent(jButtonDel)

.addComponent(jTextFieldTop, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jButtonClear))

.addGap(17, 17, 17)

.addComponent(jLabel3)

.addGap(8, 8, 8)

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

.addComponent(jButtonFind)

.addComponent(jTextFieldStep, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)))

.addGroup(layout.createSequentialGroup()

.addComponent(jButtonFileWriterText)

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

.addComponent(jButtonFileReaderText)

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

.addComponent(jButtonFileWriterBin)

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

.addComponent(jButtonFileReaderBin)))

.addGap(18, 18, 18)

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 176, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

);

pack();

setLocationRelativeTo(null);

}// </editor-fold>

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

try {

DefaultTableModel TModel = (DefaultTableModel) jTable1.getModel();

if ((RecIntegral.isNumeric(jTextFieldLow.getText()) && RecIntegral.isNumeric(jTextFieldTop.getText()) && RecIntegral.isNumeric(jTextFieldStep.getText())) == false)

throw new ExceptionRecIntegral("Ошибка при вводе исходных данных");

else if (Double.parseDouble(jTextFieldLow.getText()) >= Double.parseDouble(jTextFieldTop.getText())) {

throw new ExceptionRecIntegral("Ошибка при вводе исходных данных");

}

else {

TModel.addRow(new Object[] {jTextFieldLow.getText(), jTextFieldTop.getText(), jTextFieldStep.getText()});

List.addLast(new RecIntegral(Double.valueOf(jTextFieldLow.getText()), Double.valueOf(jTextFieldTop.getText()), Double.valueOf(jTextFieldStep.getText())));

}

}

catch (ExceptionRecIntegral ExRec) {

jDialog1.setSize(250, 100);

jDialog1.setLocationRelativeTo(null);

jDialog1.setModal(true);

jLabelError.setText(ExRec.getMessage());

jDialog1.setVisible(true);

}

}

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

int row = jTable1.getSelectedRow();

if (row != -1) {

DefaultTableModel TModel = (DefaultTableModel) jTable1.getModel();

TModel.removeRow(row);

List.remove(row);

}

}

private void connectToServer() {

try {

socket = new Socket("localhost", 9999);

// Создаем потоки для обмена данными с сервером

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

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

}

catch(IOException ex) {

ex.printStackTrace();

}

}

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

int row = jTable1.getSelectedRow();

if (row != -1) {

double h, a, b, res = 0;

a = Double.parseDouble(jTable1.getValueAt(row, 0).toString());

b = Double.parseDouble(jTable1.getValueAt(row, 1).toString());

h = Double.parseDouble(jTable1.getValueAt(row, 2).toString());

if ((b - a) % h < 0.0001 || (b - a) % h > h - 0.0001) {

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

LinkedList<ThreadIntegral> ThreadList = new LinkedList<ThreadIntegral>();

int n = 2;

double i = 0, hs, ha, hb;

hs = (b - a) / n;

while (hs % h >= 0.00001 && hs % h <= h - 0.00001) {

n = n - 1;

hs = (b - a) / n;

}

for (ha = a; ha < b; ha += hs) {

hb = ha + hs;

RecIntList.addLast(new RecIntegral(ha, hb, h));

}

for (RecIntegral Rec: RecIntList) {

i += 1;

ThreadList.addLast(new ThreadIntegral("ThreadIntegral " + i, Rec));

}

for (ThreadIntegral Thread: ThreadList) {

Thread.start();

}

try {

for (ThreadIntegral Thread: ThreadList) {

Thread.join();

res += Thread.getRes();

}

// Отправляем данные на сервер

out.println(Double.toString(res));

// Ждем ответа от сервера

String response = in.readLine();

System.out.println(response);

jTable1.setValueAt(response, row, 3);

}

catch(InterruptedException ie) {

} catch (IOException ex) {

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

}

}

}

}

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

if (List.size() != -1) {

DefaultTableModel TModel = (DefaultTableModel) jTable1.getModel();

for (RecIntegral Rec: List)

TModel.addRow(new Object[] {Rec.getA(), Rec.getB(), Rec.getH()});

}

}

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

DefaultTableModel TModel = (DefaultTableModel) jTable1.getModel();

TModel.setRowCount(0);

}

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

try {

jFileChooser1.showDialog(null, "Выбор файла для сохранения");

DefaultTableModel TModel = (DefaultTableModel) jTable1.getModel();

int row = TModel.getRowCount();

if (row != 0) {

File ft = jFileChooser1.getSelectedFile();

FileWriter fwt = new FileWriter(ft);

ft.delete();

for (RecIntegral Rec: List) {

fwt.write(Rec.getA() + "\n");

fwt.write(Rec.getB() + "\n");

fwt.write(Rec.getH() + "\n");

}

fwt.close();

}

}

catch(Exception ex) {

}

}

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

try {

jFileChooser1.showDialog(null, "Выбор файла для загрузки");

DefaultTableModel TModel = (DefaultTableModel) jTable1.getModel();

TModel.setRowCount(0);

File ft = jFileChooser1.getSelectedFile();

FileReader frt = new FileReader(ft);

Scanner scan = new Scanner(frt);

while (scan.hasNextLine())

TModel.addRow(new Object[] {scan.nextLine(), scan.nextLine(), scan.nextLine()});

frt.close();

frt.close();

}

catch(Exception ex) {

}

}

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

ObjectOutputStream fwb = null;

try {

jFileChooser1.showDialog(null, "Выбор файла для сохранения");

File fb = jFileChooser1.getSelectedFile();

fwb = new ObjectOutputStream(new BufferedOutputStream(new FileOutputStream(fb)));

fb.delete();

DefaultTableModel TModel = (DefaultTableModel) jTable1.getModel();

int row = TModel.getRowCount();

for (RecIntegral Rec: List) {

fwb.writeObject(Rec);

}

fwb.close();

}

catch (Exception ex ) {

}

}

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

ObjectInputStream frb = null;

try {

jFileChooser1.showDialog(null, "Выбор файла для загрузки");

File fb = jFileChooser1.getSelectedFile();

DefaultTableModel TModel = (DefaultTableModel) jTable1.getModel();

TModel.setRowCount(0);

frb = new ObjectInputStream(new BufferedInputStream(new FileInputStream(fb)));

fb.delete();

RecIntegral RecInt = null;

while (frb.available() != -1) {

RecInt = (RecIntegral)frb.readObject();

TModel.addRow(new Object[] {Double.toString(RecInt.getA()), Double.toString(RecInt.getB()), Double.toString(RecInt.getH())});

}

frb.close();

}

catch (Exception ex ) {

}

}

public static void main(String args[]) {

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(NewJFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

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

} catch (IllegalAccessException ex) {

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

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

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

}

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

public void run() {

new NewJFrame().setVisible(true);

}

});

}

// Variables declaration - do not modify

private javax.swing.JButton jButtonAdd;

private javax.swing.JButton jButtonClear;

private javax.swing.JButton jButtonDel;

private javax.swing.JButton jButtonFileReaderBin;

private javax.swing.JButton jButtonFileReaderText;

private javax.swing.JButton jButtonFileWriterBin;

private javax.swing.JButton jButtonFileWriterText;

private javax.swing.JButton jButtonFill;

private javax.swing.JButton jButtonFind;

private javax.swing.JDialog jDialog1;

private javax.swing.JFileChooser jFileChooser1;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JLabel jLabelError;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JTable jTable1;

private javax.swing.JTextField jTextFieldLow;

private javax.swing.JTextField jTextFieldStep;

private javax.swing.JTextField jTextFieldTop;

// End of variables declaration

}

Класс RecIntegral

package my.javaapplication1;

import java.io.\*;

public class RecIntegral implements Serializable {

private double a, b, h;

public RecIntegral(double a, double b, double h) {

this.a = a;

this.b = b;

this.h = h;

}

public double geta(){

return this.a;

}

public double getb(){

return this.b;

}

public double geth(){

return this.h;

}

public static boolean isNumeric(String num) {

try {

if((Double.parseDouble(num) > 0.000001 && Double.parseDouble(num) < 1000000))

return true;

else

return false;

}

catch(NumberFormatException e) {

return false;

}

}

}

Класс ExceptionRecIntegral

package my.javaapplication1;

public class ExceptionRecIntegral extends Exception {

public ExceptionRecIntegral(String message){

super(message);

}

}

Класс Server

package my.javaapplication1;

import java.io.\*;

import java.net.\*;

public class Server {

public static void main(String[] args) {

try {

ServerSocket serverSocket = new ServerSocket(9999); // Создаем серверный сокет, который прослушивает порт 9999

System.out.println("Server is running. Wait client ...");

while (true) {

Socket clientSocket = serverSocket.accept(); // Принимаем соединение от клиента

System.out.println("Client connected: " + clientSocket);

// Создаем потоки для обмена данными с клиентом

BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);

// Читаем и отправляем данные клиенту в бесконечном цикле

while (true) {

// Читаем данные от клиента

String dataFromClient = in.readLine();

if (dataFromClient == null) break;

// Выводим принятые данные

System.out.println("DATA from client: " + dataFromClient);

// Отправляем данные обратно клиенту

out.println(dataFromClient);

}

// Закрываем соединение с клиентом

clientSocket.close();

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

РЕЗУЛЬТАТЫ ИСПЫТАНИЙ

Приложение Б

(обязательное)

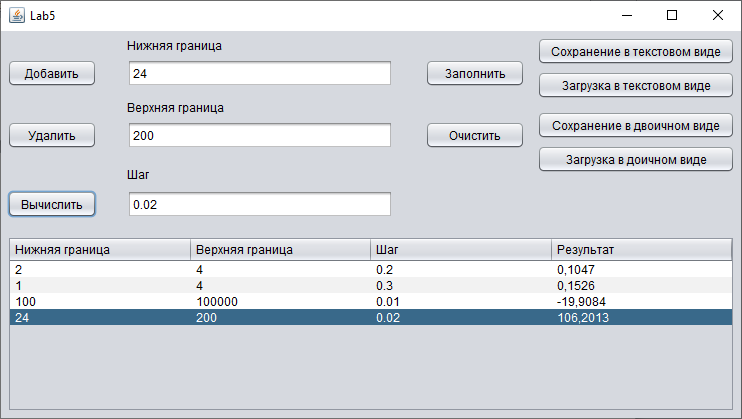


Рисунок Б.1 – Вычисление определенного интеграла

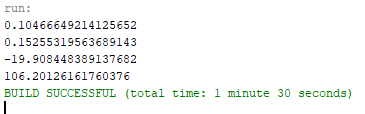


Рисунок Б.2 – Результаты выполнения программы Клиента

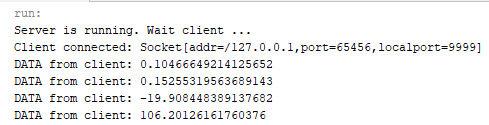


Рисунок Б.3 – Результаты выполнения программы Сервера