Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №6

по курсу «Разработка кроссплатформенных приложений»

на тему «**Сетевое взаимодействие в Java**»

**Выполнили студенты группы 21ввв1:**

Вартанов А.

Федулов Е.

**Приняли**

Юрова О.В.

Карамышева Н.С.

Пенза 2023

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

**Задание на лабораторную работу**

Модифицировать приложение, 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. Используется протокол TCP.

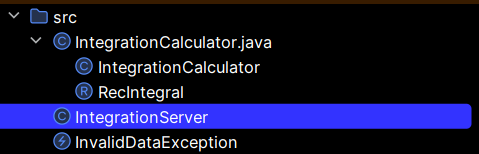
Вариант:

|  |  |
| --- | --- |
| Номер варианта | Функция |
| 2 |  |

**Ход выполнения:**

**Серверная часть.**

**Для начала нам надо реализовать серверную часть.**

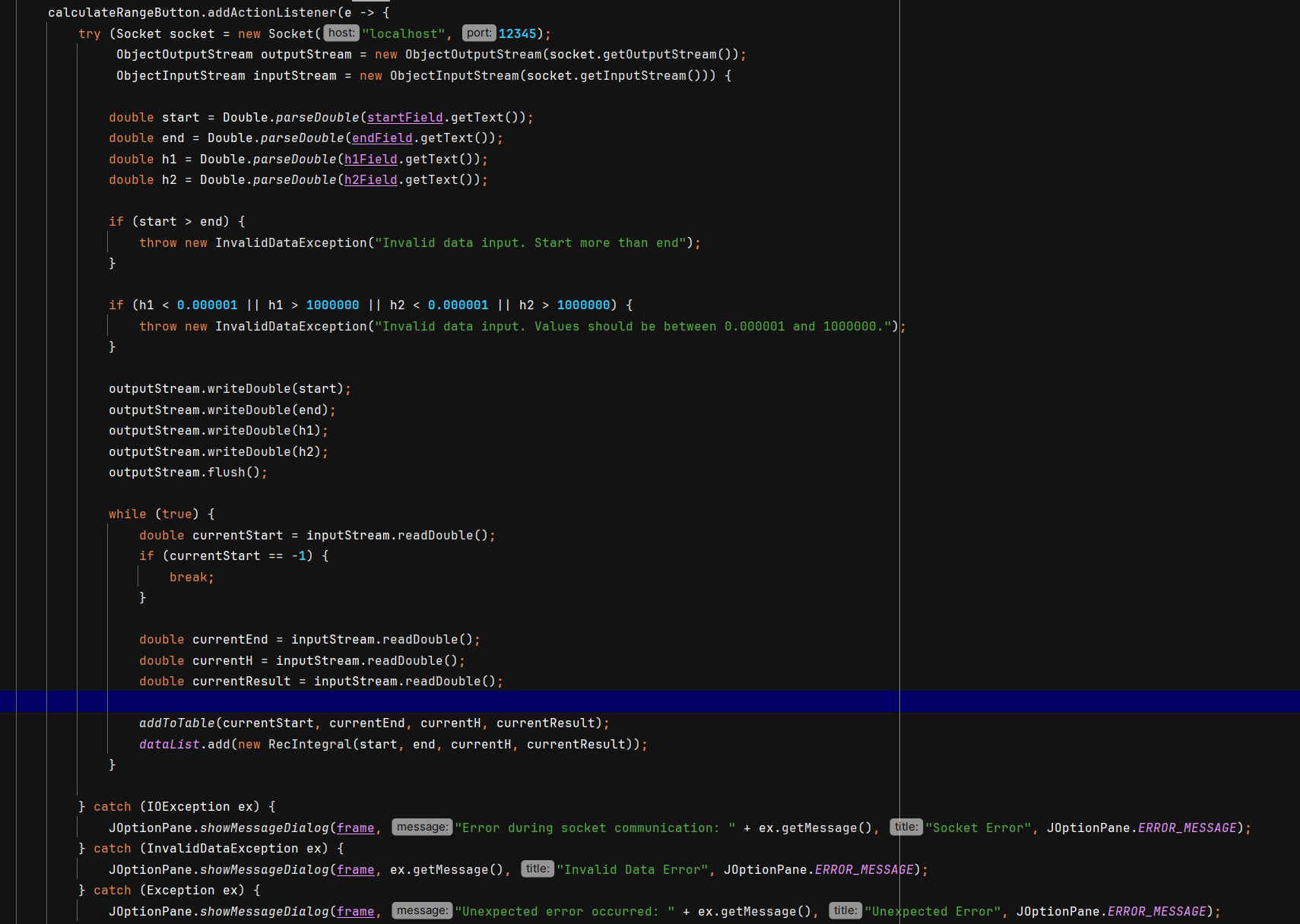
****

**Создадим новый отдельный класс IntegrationServer, который будет принимать входные данные и после вычисления отправлять обратно на клиент, тоесть в общую таблицу программы.**

****

**Клиентская часть.**

**Открывая программу мы присоединяемся к серверу и используем сокеты. При нажатии calculateRange мы отправляем вводные данные(start,end,h1,h2) для дальнейшего вычисления с помощью потоков и отправления обратно результаты, которые потом отправляются на клиент:**

****

**Листинг:**

**СЕРВЕРНАЯ ЧАСТЬ:**

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.function.Function;

public class IntegrationServer {

public static void main(String[] args) {

ServerSocket serverSocket = null;

try {

serverSocket = new ServerSocket(12345);

System.out.println("Server started. Waiting for clients...");

while (true) {

Socket clientSocket = serverSocket.accept();

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

Thread clientHandler = new Thread(() -> {

try {

ObjectInputStream inputStream = new ObjectInputStream(clientSocket.getInputStream());

ObjectOutputStream outputStream = new ObjectOutputStream(clientSocket.getOutputStream());

double start = inputStream.readDouble();

double end = inputStream.readDouble();

double h1 = inputStream.readDouble();

double h2 = inputStream.readDouble();

if (start > end) {

throw new Exception("Invalid data input. Start more than end");

}

if (h1 > 1000000 || h2 > 1000000) {

throw new Exception("Invalid data input. Values should be between 0.000001 and 1000000.");

}

Function<Double, Double> sinFunction = a -> Math.sin(a);

for (double currentH = h1; currentH <= h2; currentH += 0.1) {

double integral = integralTrapec(start, end, currentH, sinFunction);

outputStream.writeDouble(start);

outputStream.writeDouble(end);

outputStream.writeDouble(currentH);

outputStream.writeDouble(integral);

outputStream.flush();

}

outputStream.writeDouble(-1); // Метка для конца вычислений

outputStream.flush();

clientSocket.close();

} catch (Exception e) {

e.printStackTrace();

}

});

clientHandler.start();

}

} catch (Exception e) {

e.printStackTrace();

} finally {

if (serverSocket != null) {

try {

serverSocket.close();

} catch (Exception e) {

e.printStackTrace();

}

}

}

}

public static double integralTrapec(double a, double b, double h, Function<Double, Double> function) {

double area = 0;

for (int i = 0; a + i \* h < b; i++) {

area += h \* (0.5 \* function.apply(a + i \* h) + function.apply(a + (i + 1) \* h));

}

return area;

}

}

**КЛИЕНТСКАЯ ЧАСТЬ:**

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.io.\*;

import java.net.Socket;

import java.util.LinkedList;

import java.util.function.Function;

import java.util.concurrent.CountDownLatch;

public class IntegrationCalculator {

private static DefaultTableModel tableModel;

private static final LinkedList<RecIntegral> dataList = new LinkedList<>();

public static void main(String[] args) {

JFrame frame = new JFrame("Numerical Integration of sin(x)");

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

frame.setSize(800, 600);

tableModel = new DefaultTableModel();

tableModel.addColumn("Start");

tableModel.addColumn("End");

tableModel.addColumn("h");

tableModel.addColumn("Result");

/\* Заполнение коллекции dataList с предварительно заданными данными

double start1 = 0.0;

double end1 = Math.PI / 2;

double h1 = 0.1;

double result1 = integralTrapec(start1, end1, h1, a3 -> Math.sin((Double) a3));

dataList.add(new RecIntegral(start1, end1, h1, result1));

\*/

JTable table = new JTable(tableModel) {

@Override

public boolean isCellEditable(int i, int i1) {

return false;

}

};

JScrollPane scrollPane = new JScrollPane(table);

JPanel inputPanel = new JPanel(new GridLayout(18, 3));

JLabel startLabel = new JLabel("Start:");

JTextField startField = new JTextField();

JLabel endLabel = new JLabel("End:");

JTextField endField = new JTextField();

JLabel hLabel = new JLabel("h:");

JTextField hField = new JTextField();

JLabel h1Label = new JLabel("h1:");

JTextField h1Field = new JTextField();

JLabel h2Label = new JLabel("h2:");

JTextField h2Field = new JTextField();

JButton calculateButton = new JButton("Calculate");

JButton deleteButton = new JButton("Delete");

JButton clearButton = new JButton("Clear");

JButton fillButton = new JButton("Fill");

JButton addAllDataButton = new JButton("Add for link");

JButton deleteAllDataButton = new JButton("Clear all from link");

JButton saveButton = new JButton("Save to File");

JButton loadButton = new JButton("Load from File");

JButton saveBinaryButton = new JButton("Save Binary");

JButton loadBinaryButton = new JButton("Load Binary");

JButton calculateRangeButton = new JButton("Calculate Range");

inputPanel.add(startLabel);

inputPanel.add(startField);

inputPanel.add(endLabel);

inputPanel.add(endField);

inputPanel.add(hLabel);

inputPanel.add(hField);

inputPanel.add(h1Label);

inputPanel.add(h1Field);

inputPanel.add(h2Label);

inputPanel.add(h2Field);

inputPanel.add(calculateButton);

inputPanel.add(deleteButton);

inputPanel.add(clearButton);

inputPanel.add(fillButton);

inputPanel.add(addAllDataButton);

inputPanel.add(deleteAllDataButton);

inputPanel.add(saveButton);

inputPanel.add(loadButton);

inputPanel.add(saveBinaryButton);

inputPanel.add(loadBinaryButton);

inputPanel.add(calculateRangeButton);

calculateButton.addActionListener(e -> {

try {

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

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

double h = Double.parseDouble(hField.getText());

if (a > b) {

throw new InvalidDataException("Invalid data input. Start more then end");

}

if (h > b - a) {

throw new InvalidDataException("Invalid data input. h shouldn't be more then (end-start)");

}

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

throw new InvalidDataException("Invalid data input. Values should be between 0.000001 and 1000000.");

}

double integral = integralTrapec(a, b, h, a1 -> Math.sin((Double) a1));

addToTable(a, b, h, integral);

} catch (InvalidDataException ex) {

JOptionPane.showMessageDialog(frame, ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

});

deleteButton.addActionListener(e -> {

int selectedRow = table.getSelectedRow();

if (selectedRow != -1) {

tableModel.removeRow(selectedRow);

}

});

clearButton.addActionListener(e -> tableModel.setRowCount(0));

fillButton.addActionListener(e -> {

for (RecIntegral rec : dataList) {

tableModel.addRow(new Object[]{rec.start(), rec.end(), rec.h(), rec.result()});

}

});

addAllDataButton.addActionListener(e -> {

try {

double start = Double.parseDouble(startField.getText());

double end = Double.parseDouble(endField.getText());

double h = Double.parseDouble(hField.getText());

if (start > end) {

throw new InvalidDataException("Invalid data input. Start more then end");

}

if (h > end - start) {

throw new InvalidDataException("Invalid data input. h shouldn't be more then (end-start)");

}

if (start < 0.000001 || start > 1000000 || end < 0.000001 || end > 1000000 || h < 0.000001 || h > 1000000) {

throw new InvalidDataException("Invalid data input. Values should be between 0.000001 and 1000000.");

}

double integral = integralTrapec(start, end, h, a -> Math.sin((Double) a));

dataList.add(new RecIntegral(start, end, h, integral));

} catch (InvalidDataException ex) {

JOptionPane.showMessageDialog(frame, ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

});

deleteAllDataButton.addActionListener(e -> dataList.clear());

saveButton.addActionListener(e -> {

JFileChooser fileChooser = new JFileChooser();

int option = fileChooser.showSaveDialog(null);

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

try {

File file = fileChooser.getSelectedFile();

if (!file.getName().endsWith(".txt")) {

file = new File(file.getPath() + ".txt");

}

FileWriter writer = new FileWriter(file);

for (RecIntegral rec : dataList) {

writer.write(rec.start() + ";" + rec.end() + ";" + rec.h() + ";" + rec.result() + "\n");

}

writer.close();

} catch (IOException ex) {

ex.printStackTrace();

}

}

});

loadButton.addActionListener(e -> {

JFileChooser fileChooser = new JFileChooser();

int option = fileChooser.showOpenDialog(null);

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

try {

File file = fileChooser.getSelectedFile();

BufferedReader reader = new BufferedReader(new FileReader(file));

String line;

// Очищаем текущие данные из dataList

dataList.clear();

tableModel.setRowCount(0);

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

String[] values = line.split(";");

double start = Double.parseDouble(values[0]);

double end = Double.parseDouble(values[1]);

double h = Double.parseDouble(values[2]);

double result = Double.parseDouble(values[3]);

dataList.add(new RecIntegral(start, end, h, result));

tableModel.addRow(new Object[]{start, end, h, result});

}

reader.close();

} catch (IOException ex) {

ex.printStackTrace();

}

}

});

saveBinaryButton.addActionListener(e -> {

JFileChooser fileChooser = new JFileChooser();

int option = fileChooser.showSaveDialog(null);

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

try {

File file = fileChooser.getSelectedFile();

if (!file.getName().endsWith(".txt")) {

file = new File(file.getPath() + ".txt");

}

FileOutputStream fileOutputStream = new FileOutputStream(file);

ObjectOutputStream objectOutputStream = new ObjectOutputStream(fileOutputStream);

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

Double start = (Double) tableModel.getValueAt(i, 0);

Double end = (Double) tableModel.getValueAt(i, 1);

Double h = (Double) tableModel.getValueAt(i, 2);

Double result = (Double) tableModel.getValueAt(i, 3);

RecIntegral rec = new RecIntegral(start, end, h, result);

objectOutputStream.writeObject(rec);

}

objectOutputStream.close();

} catch (IOException ex) {

ex.printStackTrace();

}

}

});

loadBinaryButton.addActionListener(e -> {

JFileChooser fileChooser = new JFileChooser();

int option = fileChooser.showOpenDialog(null);

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

try {

File file = fileChooser.getSelectedFile();

FileInputStream fileInputStream = new FileInputStream(file);

ObjectInputStream objectInputStream = new ObjectInputStream(fileInputStream);

dataList.clear();

tableModel.setRowCount(0);

while (fileInputStream.available() > 0) {

RecIntegral rec = (RecIntegral) objectInputStream.readObject();

dataList.add(rec);

tableModel.addRow(new Object[]{rec.start(), rec.end(), rec.h(), rec.result()});

}

objectInputStream.close();

} catch (IOException | ClassNotFoundException ex) {

ex.printStackTrace();

}

}

});

calculateRangeButton.addActionListener(e -> {

try (Socket socket = new Socket("localhost", 12345);

ObjectOutputStream outputStream = new ObjectOutputStream(socket.getOutputStream());

ObjectInputStream inputStream = new ObjectInputStream(socket.getInputStream())) {

double start = Double.parseDouble(startField.getText());

double end = Double.parseDouble(endField.getText());

double h1 = Double.parseDouble(h1Field.getText());

double h2 = Double.parseDouble(h2Field.getText());

if (start > end) {

throw new InvalidDataException("Invalid data input. Start more than end");

}

if (h1 < 0.000001 || h1 > 1000000 || h2 < 0.000001 || h2 > 1000000) {

throw new InvalidDataException("Invalid data input. Values should be between 0.000001 and 1000000.");

}

outputStream.writeDouble(start);

outputStream.writeDouble(end);

outputStream.writeDouble(h1);

outputStream.writeDouble(h2);

outputStream.flush();

while (true) {

double currentStart = inputStream.readDouble();

if (currentStart == -1) {

break;

}

double currentEnd = inputStream.readDouble();

double currentH = inputStream.readDouble();

double currentResult = inputStream.readDouble();

addToTable(currentStart, currentEnd, currentH, currentResult);

}

} catch (IOException ex) {

JOptionPane.showMessageDialog(frame, "Error during socket communication: " + ex.getMessage(), "Socket Error", JOptionPane.ERROR\_MESSAGE);

} catch (InvalidDataException ex) {

JOptionPane.showMessageDialog(frame, ex.getMessage(), "Invalid Data Error", JOptionPane.ERROR\_MESSAGE);

} catch (Exception ex) {

JOptionPane.showMessageDialog(frame, "Unexpected error occurred: " + ex.getMessage(), "Unexpected Error", JOptionPane.ERROR\_MESSAGE);

}

});

CountDownLatch latch = new CountDownLatch(1); // Создаем объект CountDownLatch с начальным значением 1

Thread calculationThread = new Thread(() -> {

double a = 1.0;

double b = 10.0;

for (double h = 0.1; h <= 0.5; h += 0.1) {

double integral = integralTrapec(a, b, h, a1 -> Math.sin((Double) a1));

try {

Thread.sleep(0); // Задержка в одну секунду

} catch (InterruptedException e) {

e.printStackTrace();

}

double finalH = h;

SwingUtilities.invokeLater(() -> addToTable(a, b, finalH, integral));

}

latch.countDown();

});

Thread secondCalculationThread = new Thread(() -> {

try {

latch.await();

} catch (InterruptedException e) {

e.printStackTrace();

}

double a = 1.0;

double b = 10.0;

for (double h = 0.6; h <= 1.0; h += 0.1) {

double integral = integralTrapec(a, b, h, a1 -> Math.sin((Double) a1));

try {

Thread.sleep(0);

} catch (InterruptedException e) {

e.printStackTrace();

}

double finalH = h;

SwingUtilities.invokeLater(() -> addToTable(a, b, finalH, integral));

}

});

calculationThread.start();

secondCalculationThread.start();

frame.add(inputPanel, BorderLayout.NORTH);

frame.add(scrollPane, BorderLayout.CENTER);

frame.setVisible(true);

}

public static double integralTrapec(double a, double b, double h, Function function) {

double area = 0;

for (int i = 0; a + i \* h < b; i++) {

area += h \* (0.5 \* (double) function.apply(a + i \* h) + (double) function.apply(a + (i + 1) \* h));

}

return area;

}

public static void addToTable(double a, double b, double h, double result) {

tableModel.addRow(new Object[]{a, b, h, result});

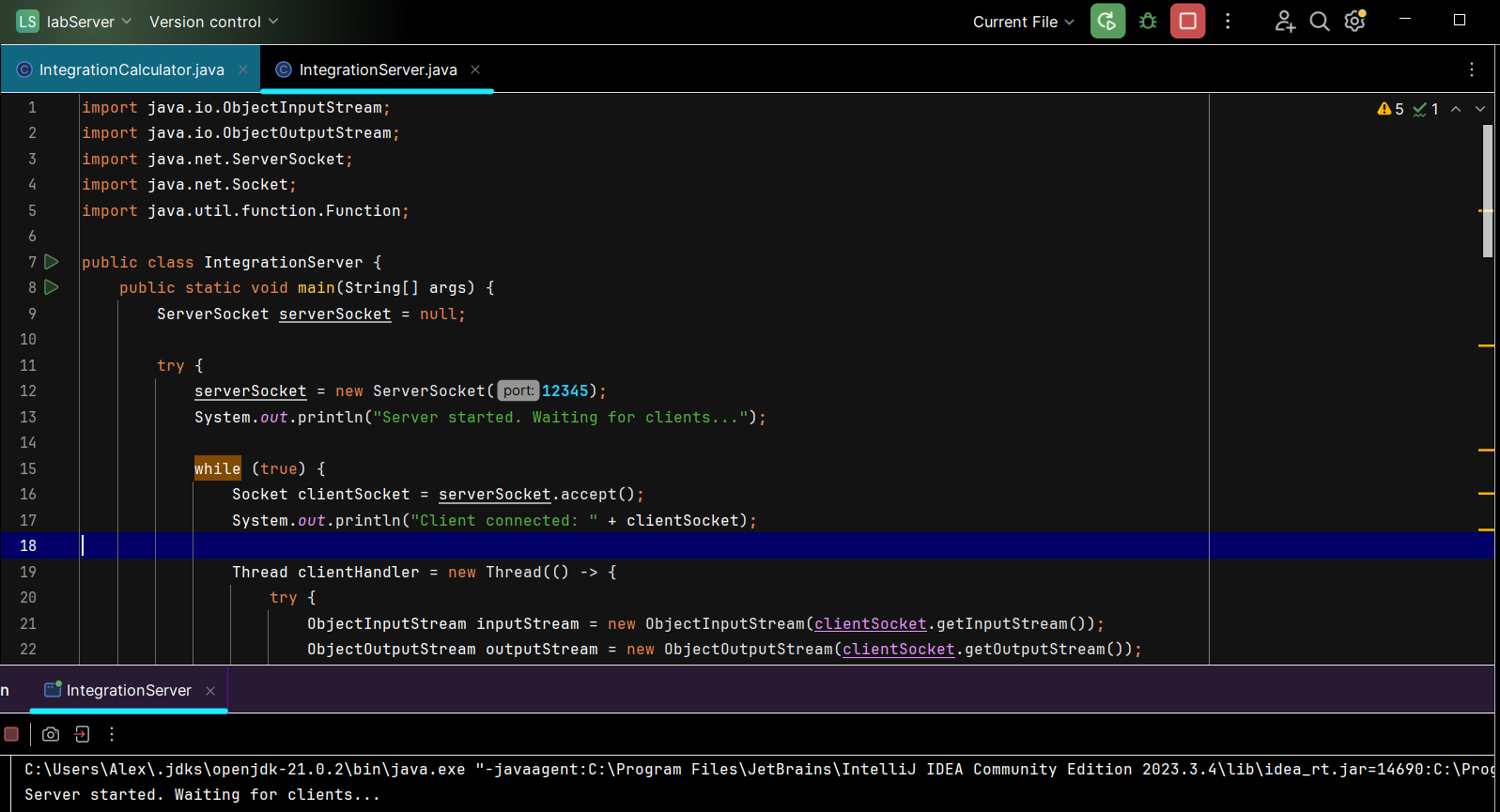
}

}

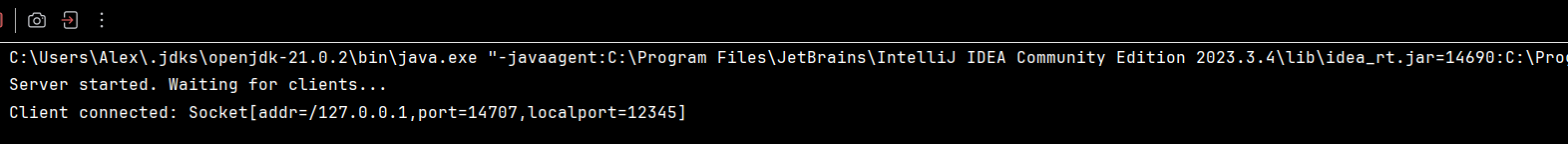
record RecIntegral(double start, double end, double h, double result) implements Serializable {

}

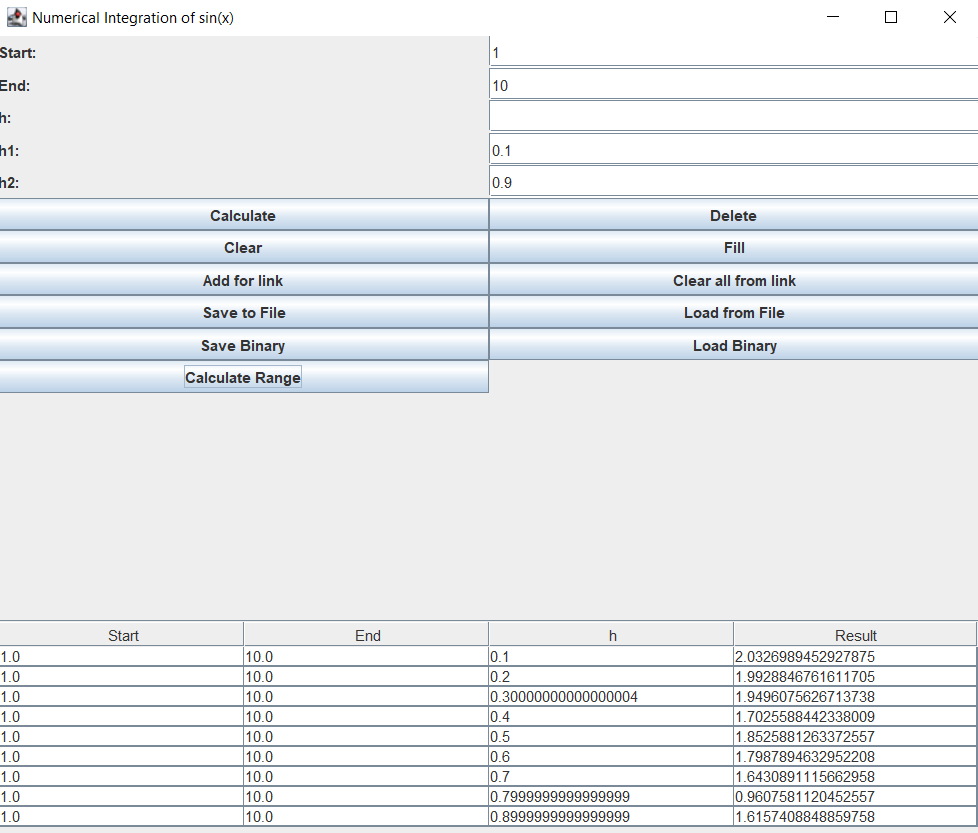
**Результаты работы программы:**

Запуск сервера:  


Проверили подключение:



**Вычисление в заданном диапазоне**

****

**Вывод**: Мы модифицировали приложение, реализовав работу вычисление определённого интеграла в нескольких дополнительных потоках.