Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №6

по курсу «Сетевое взаимодействие в Java»

**Выполнили студенты группы 21ввв1:**

Гришин С.В.

Кузнецов А.С.

**Приняли**

Юрова О.В.

Карамышева Н.С.

Пенза 2024

### 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.

**Листинг:**

package com.mycompany.integratoreserver;

import static com.mycompany.integratoreserver.IntegratorEServer.CalcResults;

import static com.mycompany.integratoreserver.IntegratorEServer.ClientList;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.PrintWriter;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.\*;

public class IntegratorEServer

{

public static ArrayList<ClientProcess> ClientList;

public static ResultsList CalcResults;

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

{

int Port = 9642;

ServerSocket serverSocket;

ClientList = new ArrayList<ClientProcess>();

CalcResults = new ResultsList();

System.out.println("Сервер запущен");

while(true)

{

serverSocket = new ServerSocket(Port);

Socket clientSocket = serverSocket.accept();

System.out.println("Новое подключение");

ClientProcess NewClient = new ClientProcess(clientSocket);

ClientList.add(NewClient);

ClientList.getLast().start();

Port++;

}

}

}

class ClientProcess extends Thread

{

Socket ClientSocket;

BufferedReader in;

PrintWriter out;

String message = null;

int ID;

public ClientProcess(Socket ClientSocket) throws IOException

{

System.out.println("Клиент ИН");

this.ClientSocket = ClientSocket;

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

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

}

public int GetID() {return ID;}

public void SendMessage(String message)

{

out.println(message);

out.flush();

}

public void CloseConnection()

{

System.out.println("Клиент отключился.");

ClientList.remove(this);

try {

ClientSocket.close();

} catch (IOException ex) {

ex.printStackTrace();

}

}

public PrintWriter out(){return out;}

public BufferedReader in(){return in;}

public double NetsCalculate(double Upper\_bound, double Lower\_bound, double Step)

{

double Result = 0;

int PID = (int)Math.random();

int ClientsCount = ClientList.size();

double step = (Upper\_bound - Lower\_bound)/ ClientsCount;

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

{

double lowerBound = i \* step;

double upper = Math.min((i + 1) \* step, Upper\_bound);

ClientList.get(i).SendMessage("CALC\_START " + PID + " " + upper + " " + lowerBound + " " + Step);

}

return Result;

}

public void run()

{

System.out.println("Клиент подключён");

while (true)

{

System.out.println("ожидание сообщения ");

Message message;

try {

message = (new Message(in.readLine()));

} catch (IOException ex) {

ex.printStackTrace();

CloseConnection();

break;

}

if (message == null) {

CloseConnection();

break;

}

System.out.println("получено сообщение " + message.Text);

if(message.Command.equals("CALC"))

{

String[] Tokens = message.Text.split(" ");

System.out.println("получен запрос");

NetCalculate NC = new NetCalculate(Double.parseDouble(Tokens[0]),Double.parseDouble(Tokens[1]),Double.parseDouble(Tokens[2]),out);

NC.start();

}else if(message.Command.equals("CALC\_END"))

{

String[] Tokens = message.Text.split(" ");

System.out.println("получен ответ");

CalcResults.AddResult(Integer.parseInt(Tokens[0]),Double.parseDouble(Tokens[1]));

}

}

//CloseConnection();

}

}

class NetCalculate extends Thread

{

double Upper\_bound;

double Lower\_bound;

double Step;

PrintWriter out;

public NetCalculate(double Upper\_bound, double Lower\_bound, double Step, PrintWriter out)

{

this.Upper\_bound = Upper\_bound;

this.Lower\_bound = Lower\_bound;

this.Step = Step;

this.out = out;

}

public void run()

{

System.out.println("начало вычисления");

double Result = 0;

int PID = 1; //(int)Math.random();

int ClientsCount = ClientList.size();

System.out.println("r " + ClientsCount);

double step = (Upper\_bound - Lower\_bound)/ ClientsCount;

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

{

double lowerBound = (i \* step) + Lower\_bound;

double upper = Math.min(((i+1) \* step) + Lower\_bound, Upper\_bound);

System.out.println("CALC\_START " + PID + " " + upper + " " + lowerBound + " " + Step);

ClientList.get(i).SendMessage("CALC\_START " + PID + " " + upper + " " + lowerBound + " " + Step);

}

System.out.println("запросы разосланны запрос " + ClientsCount + " по " + PID);

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

{

Result += CalcResults.WaitResult(PID);

}

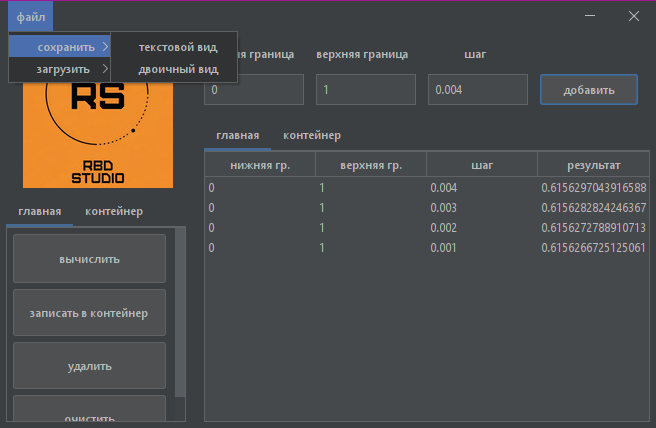
System.out.println("вычисление законченно");

out.println("CALC\_RESULT " + Result);

}

}

**Результаты работы программы:**

****

**Вывод:** в ходе выполнения лабораторной работы мы научились создавать клиент-серверные приложения с использованием стандартных классов Java.