Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

ОТЧЁТ

По лабораторной работе №6

По курсу «Программирование на языке JAVA»

на тему: «Сетевое взаимодействие в Java»

Выполнили:

студенты группы 20ВВП1:

Сурков Максим

Пантелеев Иван

Принял:

Юрова О.В.

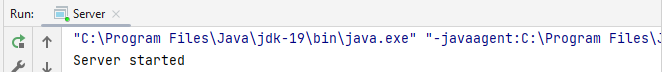
Карамышева Н.С.

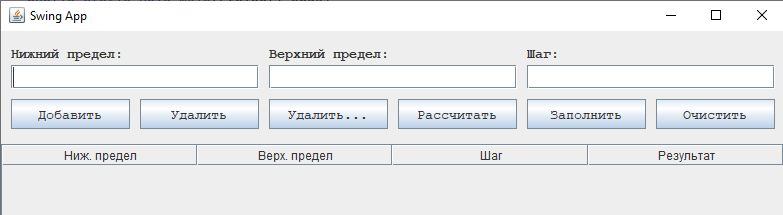
Пенза 2023

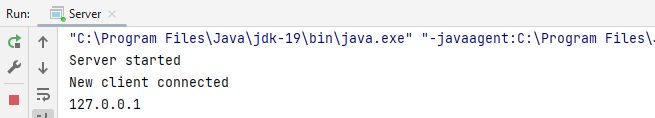
**Цель работы****:** научиться создавать клиент-серверные приложения c использованием стандартных классов Java.

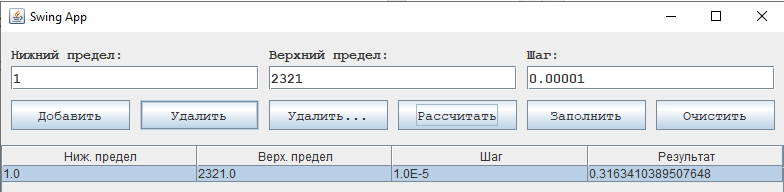
**Ход работы:**

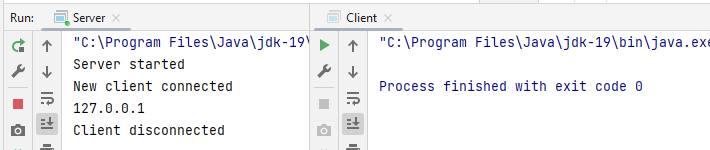
Модифицировали приложение из предыдущей лабораторной работы, реализовав клиент-серверную архитектуру, обеспечивающую распределенное вычисление определенного интеграла на нескольких вычислительных узлах (клиентах)

****

****

****

****

****

**Листинг:**

**Server.java**

package Server;  
  
import java.io.\*;  
import java.net.\*;  
  
class Server {  
 public static void main(String[] args)  
 {  
 ServerSocket server = null;  
  
 try {  
  
 server = new ServerSocket(1234);  
 server.setReuseAddress(true);  
 System.*out*.println("Server started");  
  
 while (true) {  
  
 Socket client = server.accept();  
  
 System.*out*.println("New client connected\n" + client.getInetAddress().getHostAddress());  
  
 ClientHandler clientSock = new ClientHandler(client);  
  
 new Thread(clientSock).start();  
 }  
 }  
 catch (IOException e) {  
 e.printStackTrace();  
 }  
 finally {  
 if (server != null) {  
 try {  
 server.close();  
 }  
 catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 }  
  
 private static class ClientHandler implements Runnable {  
 private final Socket clientSocket;  
  
 public ClientHandler(Socket socket)  
 {  
 this.clientSocket = socket;  
 }  
  
 public void run()  
 {  
 BufferedWriter out = null;  
 BufferedReader in = null;  
 try {  
 out = new BufferedWriter(new OutputStreamWriter(clientSocket.getOutputStream()));  
  
 in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));  
  
 String inputLine;  
 while (!"exit".equals(inputLine = in.readLine())) {  
 String[] splitLine = inputLine.split(" ");  
 Solver solver = new Solver(Double.*parseDouble*(splitLine[0]), Double.*parseDouble*(splitLine[1]), Double.*parseDouble*(splitLine[2]));  
 out.write(solver.solve());  
 out.newLine();  
 out.flush();  
 }  
 }  
 catch (IOException e) {  
 e.printStackTrace();  
 }  
 finally {  
 try {  
 if (out != null) {  
 out.close();  
 }  
 if (in != null) {  
 in.close();  
 }  
 clientSocket.close();  
 System.*out*.println("Client disconnected");  
 }  
 catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 }  
}

**Client.java**

package Client;  
import Client.UI.UserInterface;  
import java.io.\*;  
  
  
class Client {  
 public static void main(String[] args)  
 {  
 try {  
 new UserInterface();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
}

**UserInterface.java только изменения:**

public class UserInterface extends JFrame  
{  
  
 Socket socket;  
 BufferedWriter out;  
 BufferedReader in;  
  
 public UserInterface() throws IOException {  
  
 socket = new Socket("localhost", 1234);  
  
 out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));  
  
 in = new BufferedReader(new InputStreamReader(socket.getInputStream()));  
 super.addWindowListener(new WindowAdapter() {  
 @Override  
 public void windowClosing(WindowEvent e) {  
 try {  
 out.write("exit");  
 out.newLine();  
 out.flush();  
  
 in.close();  
 out.close();  
 socket.close();  
 }  
 catch (IOException a){  
 a.printStackTrace();  
 }  
  
 super.windowClosing(e);  
 }  
 });

**Solve.button ActionListener**

JButton solve = new JButton("Рассчитать");  
solve.addActionListener(e -> {  
 try {  
 int index = table1.getSelectedRow();  
 if (index == -1) {  
 throw new SimpleException("Не выбрана строка");  
 }  
 int rowCount = tableModel.getRowCount();  
 int curIndex = dataList.size() - rowCount + index;  
  
 out.write(dataList.get(curIndex).toString());  
 out.newLine();  
 out.flush();  
  
 dataList.get(curIndex).setResult(in.readLine());  
 tableModel.setValueAt(dataList.get(curIndex).getResult(), index, 3);  
 } catch (IOException a) {  
 a.printStackTrace();  
 }  
 catch (SimpleException ignored){}  
});

**Solver.java**

package Server;  
  
public class Solver {  
 private double downLimit;  
 private double upLimit;  
 private double step;  
 public Solver(double downLimit, double upLimit, double step){  
 this.downLimit = downLimit;  
 this.step = step;  
 this.upLimit = upLimit;  
 }  
 private static double areaFunc(double a, double b,double h){  
 return ((Math.*sin*(a\*a)+Math.*sin*(b\*b))/2)\*h;  
 }  
 private double fff(double a, double b,double c)  
 {  
 double res =0;  
 while(a<b){  
 res+= *areaFunc*(a,a+c,c);  
 a+=c;  
 }  
 return res;  
 }  
 public String solve(){  
  
 double a = this.downLimit;  
 double b = this.upLimit;  
 double c = this.step;  
  
 double ost = (b-a)%c;  
 a+=ost;  
 int steps = (int)((b-a)/c);  
 final double[] result = {0};  
  
 int THREADCOUNT = 8;  
 int cut = steps/THREADCOUNT;  
 int cutOst = steps%THREADCOUNT;  
  
 if(steps<THREADCOUNT) {  
 THREADCOUNT = steps;  
 cut = 1;  
 cutOst = 0;  
 }  
 Thread[] threadArr = new Thread[THREADCOUNT];  
 double finalCutOst = cutOst;  
 double finalA = a+c\*finalCutOst;  
 int finalCut = cut;  
 for (int i = 0;i<THREADCOUNT;i++){  
 int finalI = i;  
 Runnable solve = new Runnable() {  
 @Override  
 public void run() {  
 double localResult = 0;  
 if(finalI==0){  
 if(ost>0) {  
 localResult += fff(finalA - ost, finalA, ost);  
 }  
 localResult += fff(finalA-c\*finalCutOst, finalA, c);  
 }  
 localResult += fff(finalA+(c\*finalCut\*finalI),finalA+(c\*finalCut\*(finalI+1)),c);  
 synchronized(this) {  
 result[0] += localResult;  
 }  
 }  
 };  
 threadArr[i]=new Thread(solve);  
 threadArr[i].start();  
 }  
 for (Thread it : threadArr) {  
 try {  
 it.join();  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 return Double.*toString*(result[0]);  
 }  
  
  
  
 public double notMultiThreadsSolve(){  
 double a = this.downLimit;  
 double b = this.upLimit;  
 double c = this.step;  
  
 double res = 0 ;  
 double ost = (b-a)%c;  
 res+=*areaFunc*(a,a+ost,c);  
 a+=ost;  
 while(a<b){  
 res+= *areaFunc*(a,a+c,c);  
 a+=c;  
 }  
 return res;  
 }  
}