Министерство науки и образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №6

по курсу «Программирование на Java»

на тему «Сетевое взаимодействие»

Вариант 3

Выполнили студенты группы 19ВВ3:

Ланцов Андрей

Трубаненко Александр

Приняли:

Юрова О.В.

**2022**

**Цель работы**

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

**Задание на лабораторную работу**

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. Нечетные варианты используют протокол UDP, а четные TCP. Оформление лабораторной работы должно быть выполнено в соответствии с требованиями, приведенными в Приложении 2.

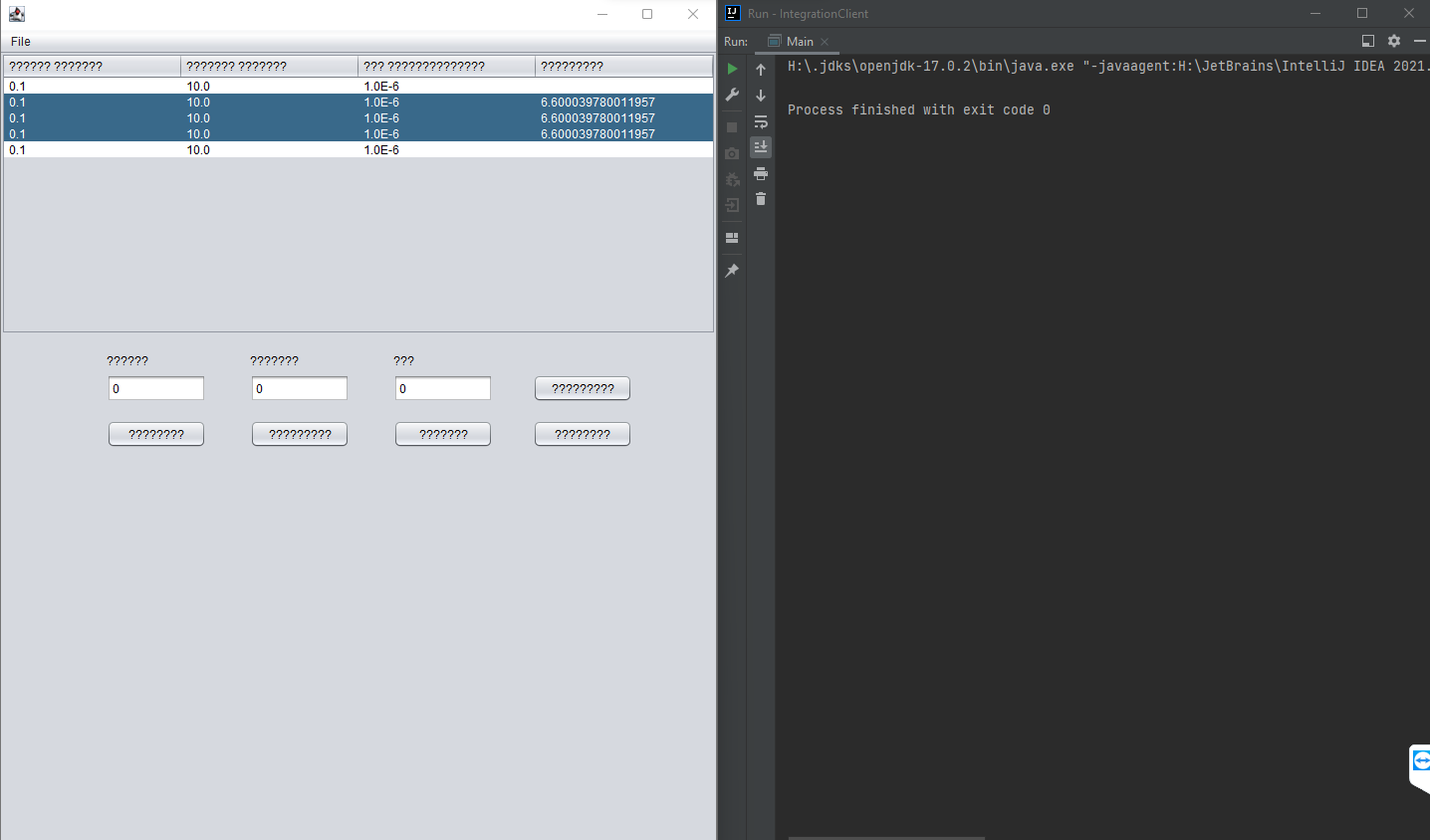
**Ход работы**

Листинг

**package** Lab1;  
  
**public interface** IntegrationServerResultGetter {  
 **void** getResult(Object[] result);  
}

**public class** IntegrationServer **extends** Thread{  
 **public static final int *port*** = 2437;  
 **public static final int *timeout*** = 30000;  
  
 **private boolean clientConnected** = **false**;  
 **private final** ServerSocket **serverSocket**;  
 IntegrationServerResultGetter **getter**;  
  
 **public** IntegrationServer(**int** port,IntegrationServerResultGetter getter) **throws** IOException {  
 **this**.**getter** = getter;  
 **serverSocket** = **new** ServerSocket(port);  
 **serverSocket**.setSoTimeout(***timeout***);  
 }  
 **public synchronized boolean** getConnectionStatus(){  
 **return clientConnected**;  
 }  
 **public void** run() {  
 **while**(**true**) {  
 **try** {  
 **clientConnected** = **false**;  
 **serverSocket**.getLocalPort();  
 Socket server = **serverSocket**.accept();  
 *//server.getRemoteSocketAddress();* DataOutputStream out = **new** DataOutputStream(server.getOutputStream());  
 DataInputStream in = **new** DataInputStream(server.getInputStream());  
  
 **clientConnected** = **true**;  
 **while** (!Globals.*getRecsStatus*()){  
 *sleep*(20);  
 }  
 Globals.*setRecsStatus*(**false**);  
  
 StringBuilder outData = **new** StringBuilder();  
 **for** (**int** i : Globals.*selectedRecs*){  
 outData.append(Globals.*tableRecs*.get(i).toString());  
 }  
 out.writeUTF(outData.toString());  
  
 **var** results = **new** ArrayList<Double>();  
 String inData = in.readUTF();  
  
 **for** (**var** line : inData.lines().toList()){  
 results.add(Double.*parseDouble*(line));  
 }  
 **getter**.getResult(results.toArray());  
 } **catch** (SocketTimeoutException s) {  
 JOptionPane.*showMessageDialog*(**null**,**"Время сокета истекло!"**);  
 **break**;  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 **break**;  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
}

**class** Client{  
 **private static final** String ***serverName*** = **"localhost"**;  
 **private static final int *port*** = 2437;  
  
 **private final int threadsCount** = 3;  
  
 **public void** start(){  
 **try** {  
 Socket client = **new** Socket(***serverName***, ***port***);  
 InputStream inFromServer = client.getInputStream();  
 DataInputStream in = **new** DataInputStream(inFromServer);  
 OutputStream outToServer = client.getOutputStream();  
 DataOutputStream out = **new** DataOutputStream(outToServer);  
  
 String inData = in.readUTF();  
ArrayList<RecIntegral> recs = **new** ArrayList<>();  
 **for** (**var** line : inData.lines().toList()){  
 recs.add(RecIntegral.*fromString*(line));  
 }  
  
 IntegrationThread[] threads = **new** IntegrationThread[**threadsCount**];  
  
 **var** results = **new** ArrayList<Double>();  
  
 **int** processedRowsCount = 0;  
 **while** (processedRowsCount != recs.size()){  
 **int** iterationsCount = Integer.*min*(**threadsCount**,recs.size()-processedRowsCount);  
 **try**{  
 **for** (**int** i = 0; i < iterationsCount; i++){  
 RecIntegral rec = recs.get(i);  
 threads[i] = **new** IntegrationThread(rec.**lowerEdge**,rec.**upperEdge**,rec.**step**);  
 threads[i].start();  
 }  
 **for** (**int** i = 0; i < iterationsCount; i++){  
 threads[i].join();  
 }  
 }**catch** (Exception e){  
 e.printStackTrace();  
 **return**;  
 }  
 **for** (**int** i = 0; i < iterationsCount; i++){  
 results.add(threads[i].getResult());  
 }  
 processedRowsCount+=iterationsCount;  
 }  
 StringBuilder outData = **new** StringBuilder();  
 **for** (**double** result:results){  
 outData.append(result).append(**"\n"**);  
 }  
 out.writeUTF(outData.toString());  
  
 } **catch** (IOException e) {  
 System.***out***.println(**"Сервер недоступен"**);  
 } **catch** (RecIntegralException | NumberFormatException e) {  
 e.printStackTrace();  
 }  
 }  
}



**Вывод**

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