Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

Пенза 2022

Выполнили:

студенты группы 19ВВ2

Макаров Д.Д.

Лобанов Д.В.

Приняли:

Карамышева Н.С

Юрова О.В.

ОТЧЕТ

по лабораторной работе №6

по курсу «Программирование на языке Java»

на тему «Сетевое взаимодействие в Java»

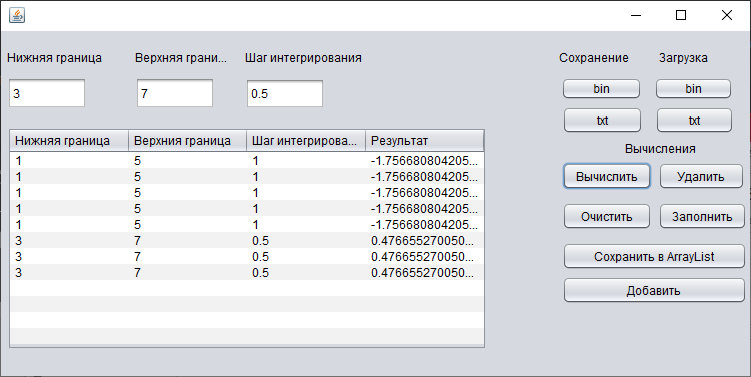
**Цель работы:**

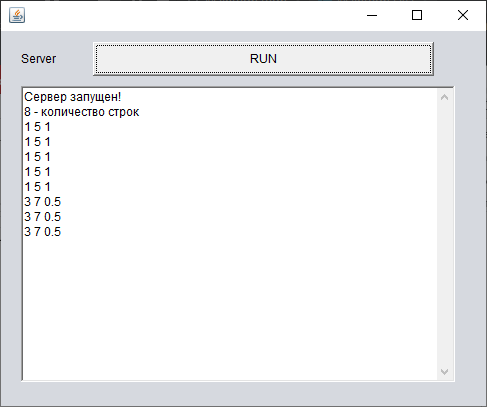
Научиться создавать клиент-серверные приложения c использованием стандартных классов Java.

**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.

**Результат работы:**

****

****

**Вывод:** Мы научились создавать клиент-серверные приложения c использованием стандартных классов Java.

**Листинг:**

ContactEditorUI.java

private void jButton3MouseClicked(java.awt.event.MouseEvent evt) throws InterruptedException  
{//GEN-FIRST:event\_jButton3MouseClicked  
int countOfFilledRows = 0;  
 try {  
 try {  
 // адрес - локальный хост, порт - 4004, такой же как у сервера  
 *clientSocket* = new Socket("localhost", 4004); // этой строкой мы запрашиваем  
 // у сервера доступ на соединение  
  
 // читать соообщения с сервера  
 *in* = new BufferedReader(new InputStreamReader(*clientSocket*.getInputStream()));  
 // писать туда же  
 *out* = new BufferedWriter(new OutputStreamWriter(*clientSocket*.getOutputStream()));  
  
 for(int i = 0; i < jTable2.getRowCount(); i++)  
 {  
 if (jTable2.getValueAt(i, 1)!=null)  
 {  
 countOfFilledRows++;  
 }  
 }  
 *out*.write(String.*valueOf*(countOfFilledRows)+"\n");  
 for (int i = 0; i<countOfFilledRows; i++)  
 {  
 *out*.write(((String) jTable2.getValueAt(i, 0))+"\n");  
 *out*.write(((String) jTable2.getValueAt(i, 1))+"\n");  
 *out*.write(((String) jTable2.getValueAt(i, 2))+"\n");  
 }  
 *out*.flush();  
 for (int i = 0; i<countOfFilledRows; i++)  
 {  
 jTable2.setValueAt(*in*.readLine(), i, 3);  
 }  
  
  
 } finally { // в любом случае необходимо закрыть сокет и потоки  
 System.*out*.println("Клиент был закрыт...");  
 *clientSocket*.close();  
 *in*.close();  
 *out*.close();  
 }  
 } catch (IOException e) {  
 System.*err*.println(e);  
 }  
  
}

LabServer.java

package com.mycompany.labserver;  
import java.io.\*;  
import java.net.ServerSocket;  
import java.net.Socket;  
  
public class Mainform extends javax.swing.JFrame   
{  
   
 public Mainform()   
 {  
 initComponents();  
 }  
   
 private static Socket *clientSocket*; //сокет для общения  
 private static ServerSocket *server*; // серверсокет  
 private static BufferedReader *in*; // поток чтения из сокета  
 private static BufferedWriter *out*; // поток записи в сокет  
   
   
   
 */\*\*  
 \* This method is called from within the constructor to initialize the form.  
 \* WARNING: Do NOT modify this code. The content of this method is always  
 \* regenerated by the Form Editor.  
 \*/* @SuppressWarnings("unchecked")  
 // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents  
 private void initComponents() {  
  
 jLabel1 = new javax.swing.JLabel();  
 RunButton = new java.awt.Button();  
 textArea1 = new java.awt.TextArea();  
  
 setDefaultCloseOperation(javax.swing.WindowConstants.*EXIT\_ON\_CLOSE*);  
  
 jLabel1.setText("Server");  
  
 RunButton.setLabel("RUN");  
 RunButton.setName("RunButton"); // NOI18N  
 RunButton.addMouseListener(new java.awt.event.MouseAdapter() {  
 public void mouseClicked(java.awt.event.MouseEvent evt) {  
 RunButtonMouseClicked(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()  
 .addGap(20, 20, 20)  
 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(layout.createSequentialGroup()  
 .addComponent(textArea1, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 434, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addContainerGap(31, Short.*MAX\_VALUE*))  
 .addGroup(layout.createSequentialGroup()  
 .addComponent(jLabel1)  
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.*RELATED*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*)  
 .addComponent(RunButton, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 341, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addGap(52, 52, 52))))  
 );  
 layout.setVerticalGroup(  
 layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(javax.swing.GroupLayout.Alignment.*TRAILING*, layout.createSequentialGroup()  
 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(layout.createSequentialGroup()  
 .addGap(19, 19, 19)  
 .addComponent(jLabel1))  
 .addGroup(layout.createSequentialGroup()  
 .addGap(11, 11, 11)  
 .addComponent(RunButton, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 34, javax.swing.GroupLayout.*PREFERRED\_SIZE*)))  
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.*RELATED*)  
 .addComponent(textArea1, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 296, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addContainerGap(24, Short.*MAX\_VALUE*))  
 );  
  
 pack();  
 }// </editor-fold>//GEN-END:initComponents  
  
 private void RunButtonMouseClicked(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_RunButtonMouseClicked  
  
  
  
 try   
 {  
 try   
 {  
 *server* = new ServerSocket(4004); // серверсокет прослушивает порт 4004  
 textArea1.append("Сервер запущен!\n");  
 // хорошо бы серверу  
 //объявить о своем запуске  
 *clientSocket* = *server*.accept();   
 // accept() будет ждать пока  
 //кто-нибудь не захочет подключиться  
 try   
 { // установив связь и воссоздав сокет для общения с клиентом можно перейти  
 // к созданию потоков ввода/вывода.  
 // теперь мы можем принимать сообщения  
 *in* = new BufferedReader(new InputStreamReader(*clientSocket*.getInputStream()));  
 // и отправлять  
 *out* = new BufferedWriter(new OutputStreamWriter(*clientSocket*.getOutputStream()));  
  
 String word = *in*.readLine(); // ждём пока клиент что-нибудь нам напишет  
 textArea1.append(word+" - количество строк \n");  
 int countOfFilledRows = 0;  
 countOfFilledRows = Integer.*parseInt*(word);  
  
 ThreadIntegral[] Thread = new ThreadIntegral[countOfFilledRows];  
  
 for (int i = 0; i<countOfFilledRows; i++)  
 {  
 String bottom;  
  
 String high;  
  
 String step;  
  
 String result;  
 Double dwResult;  
  
 bottom = *in*.readLine();  
 textArea1.append(bottom+" ");  
 high = *in*.readLine();  
 textArea1.append(high+" ");  
 step = *in*.readLine();  
 textArea1.append(step+"\n");  
  
  
  
 Thread[i] = new ThreadIntegral(  
 Double.*parseDouble*((String) bottom),  
 Double.*parseDouble*((String) high),  
 Double.*parseDouble*((String) step),  
 i);  
 Thread[i].start();  
 }  
  
 for (int i = 0; i<countOfFilledRows; i++)  
 {  
 Thread[i].join();  
 *out*.write(String.*valueOf*(Thread[i].GetResult())+"\n");  
 }  
 *out*.flush();  
 }  
 catch (InterruptedException e)  
 {  
 throw new RuntimeException(e);  
 }  
 finally  
 { // в любом случае сокет будет закрыт  
 *clientSocket*.close();  
 // потоки тоже хорошо бы закрыть  
 *in*.close();  
 *out*.close();  
 }  
 }   
 finally   
 {  
 System.*out*.println("Сервер закрыт!");  
 *server*.close();  
 }  
 }  
 catch (IOException e)  
 {  
 System.*err*.println(e);  
 }  
   
 }//GEN-LAST:event\_RunButtonMouseClicked  
  
 */\*\*  
 \** ***@param*** *args the command line arguments  
 \*/* public static void main(String args[])   
 {  
 /\* Set the Nimbus look and feel \*/  
 //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">  
 /\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.  
 \* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html   
 \*/  
 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*(Mainform.class.getName()).log(java.util.logging.Level.*SEVERE*, null, ex);  
 } catch (InstantiationException ex) {  
 java.util.logging.Logger.*getLogger*(Mainform.class.getName()).log(java.util.logging.Level.*SEVERE*, null, ex);  
 } catch (IllegalAccessException ex) {  
 java.util.logging.Logger.*getLogger*(Mainform.class.getName()).log(java.util.logging.Level.*SEVERE*, null, ex);  
 } catch (javax.swing.UnsupportedLookAndFeelException ex) {  
 java.util.logging.Logger.*getLogger*(Mainform.class.getName()).log(java.util.logging.Level.*SEVERE*, null, ex);  
 }  
 //</editor-fold>  
  
 /\* Create and display the form \*/  
 java.awt.EventQueue.*invokeLater*(new Runnable()   
 {  
 public void run()   
 {  
 new Mainform().setVisible(true);  
 }  
 });  
   
   
 }  
  
 // Variables declaration - do not modify//GEN-BEGIN:variables  
 private java.awt.Button RunButton;  
 private javax.swing.JLabel jLabel1;  
 private java.awt.TextArea textArea1;  
 // End of variables declaration//GEN-END:variables  
}