package l3;  
import java.io.\*;  
import java.net.\*;  
import javax.swing.\*;  
  
public class Client{  
 static final String *options*[]={"send","disconnect"};  
 public static void main(String args[]){  
 int a=0,b=0;  
 char c=0;  
 char[] res = new char[255];  
 char[] buf1 = new char[255];  
 char[] buf2 = new char[255];  
  
 String sign="";  
 Socket sock;  
 InputStream is;  
 OutputStream os,os2;  
 try{  
 sock=new Socket(InetAddress.*getByName*("localhost"),1024);//создаем сокет клиента и соединяемся с сервером который находится на порту 1024 на этом же компьютере  
 is=sock.getInputStream();//получили входной поток для чтения данных  
 os=sock.getOutputStream();//получили выходной поток для записи данных  
 os2=sock.getOutputStream();//получили выходной поток для записи данных  
  
 boolean cont=true; // если станет = false цикл прервется  
 while(cont){//бесконечный цикл для принятия и отсылки сообщений серверу  
  
 String stringmessage=JOptionPane.*showInputDialog*("Введите 1 строку");//считываем полылаемое сообщение для сервера  
 String stringmessage2=JOptionPane.*showInputDialog*("Введите 2 строку");  
 if(stringmessage==null || stringmessage2==null)  
 {  
 JOptionPane.*showMessageDialog*(null, "Отключение от сервера...");  
 break;}  
 if(stringmessage.compareTo("disconnect")==0 || stringmessage2.compareTo("disconnect")==0) {  
 byte bytemessage1[]=stringmessage.getBytes();//преобразование сообщения из типа String в тип byte[]  
 byte bytemessage2[]=stringmessage2.getBytes();  
 os.write(bytemessage1);//запись в выходной поток преобразованного сообщения  
 os2.write(bytemessage2);  
 JOptionPane.*showMessageDialog*(null, "Отключение от сервера...");  
 break; //выход из цила while  
 }  
 else {  
 byte bytemessage3[]=stringmessage.getBytes();//преобразование сообщения из типа String в тип byte[]  
 byte bytemessage4[]=stringmessage2.getBytes();  
 os.write(bytemessage3);//запись в выходной поток преобразованного сообщения  
 os2.write(bytemessage4);  
 byte readmessage[]=new byte[100];//создаем массив байт для чтения информации от сервера  
 int k=is.read(readmessage);//считываем сообщение посланное от сервера k - количетсво считанных символов  
  
 JOptionPane.*showMessageDialog*(null, "Полученное сообщение от сервера: \n"+new String(readmessage,0,k));//выводим полученное сообщение  
 cont=false;  
 }}  
 is.close();//закрываем входной поток  
 os.close();//закрываем выходной поток  
 os2.close();//закрываем выходной поток  
  
 sock.close();//закрываем сокет клиента  
 }  
 catch(Exception e){// если возникла непредвиденная ошибка  
 System.*out*.println("Error "+ e.toString());  
 }}}

SERVER

package l3;  
import java.net.\*;  
import java.util.StringTokenizer;  
import java.io.\*;  
  
class Server{  
  
 private static class ClientHandler implements Runnable  
 {  
 char[] res = new char[6];  
 char[] b1 = new char[255];  
 char[] b2 = new char[255];  
 char[] res1 = new char[6];  
 char[] res2= new char[6];  
  
 // локальные используемые переменные  
 boolean choose;  
 private int clientid=0;  
 private InputStream is,is2;  
 private OutputStream os;  
 private int a=0,b=0,c=0;  
 private String first,second;  
 private Socket client;  
 public ClientHandler(int id,Socket s)  
  
  
 {  
 clientid=id;  
 client=s;  
 }  
 public void run()  
 {  
 try  
 {  
 os=client.getOutputStream();//получили выходной поток для записи данных  
 is=client.getInputStream();//получили входной поток для чтения данных  
 is2=client.getInputStream();//получили входной поток для чтения данных  
  
 boolean flag=true;  
 while(flag==true){//цикл для работы с одним клиентом  
 int k,m;  
 byte clientMessage[]=new byte[100];//байтовый массив, который будет использоваться при чтении из входного потока  
 byte clientMessage2[]=new byte[100];//байтовый массив, который будет использоваться при чтении из входного потока  
 try  
 {  
  
 k=is.read(clientMessage);//чтение иформации, посланной клиентом, из вхоного потока в массив clientMessage[] k - колличество считанных байт  
 m=is2.read(clientMessage2);  
 }  
 catch(Exception e) //если клиент просто остановил свою работу не отправив disconnect  
 {  
 break;//завершение цикла обработки данного клиента  
 }  
 String tempString=new String(clientMessage,0,k);// формирование строки из того что считанно  
 String tempString2=new String(clientMessage2,0,m);  
 tempString=tempString.trim();// убираем пробелы в конце и в начале  
 tempString2=tempString2.trim();// убираем пробелы в конце и в начале  
  
 if(tempString.compareTo("disconnect")==0 || tempString2.compareTo("disconnect")==0 ) {  
 System.*out*.println("message \"disconnect\" recieved from client");  
 flag=false;// приведет к прекращению повторения цикла  
 }  
  
 else {  
 //подразумеваем что строка передана в виде a знак b  
 try  
 {  
  
 StringTokenizer tok=new StringTokenizer(tempString);// StringTokenizer - класс для разбивки строки на отдельные слова  
 StringTokenizer tok2=new StringTokenizer(tempString2);// StringTokenizer - класс для разбивки строки на отдельные слова  
  
 first=tok.nextToken();//выделяем знак  
 second=tok2.nextToken();//выделяем знак  
  
 // c=count(a,b,sign);// вызываем функцию  
 //String answer=tempString+" = "+c;  
 //os.write(answer.getBytes());//запись в выходной поток вычисленного ответа  
 // System.out.println(answer);//вывод сообщения в консоль на сервере  
  
  
 String answer;  
 choose = first.equals(second);  
 if (choose==true) {answer = "True";}  
 else{ answer = "False";}  
 os.write(answer.getBytes());  
 System.*out*.println("Строка Str1 равна строке Str2? Ответ: " + choose );  
  
  
 } catch(Exception e)// если не получилось что то преобразовать или посчитать  
 {  
 os.write(("Wrong message. "+e.toString()).getBytes());// отправляем клиенту сообщение об ошибке состоящее из wrong message и описания ошибки  
 }}}  
 is.close();//закрытие входного потока  
 is2.close();//закрытие входного потока  
  
 os.close();//закрытие выходного потока  
 client.close();//закрытие сокета, выделенного для работы с подключившимся клиентом  
 }catch(Exception e)  
 {  
  
 }  
 System.*out*.println("Client "+clientid+" disconnected");  
 }}  
 static int *countclients*=0;//счетчик подключившихся клиентов  
  
  
  
 public static void main(String args[]){  
 ServerSocket sock;  
  
 try{  
 sock=new ServerSocket(1024);//создаем серверный сокет работающий локально по порту 1024  
  
 while(true){//бесконечный цикл для возможности подключения последовательно нескольних клиентов  
 Socket client=sock.accept();//сработает, когда клиент подключится, для него выделится отдельный сокет client  
 *countclients*++;//количество подключившихся клиентов увеличивается на 1  
  
 System.*out*.println("Client "+*countclients*+" connected");//вывод сообщения  
 new Thread(new ClientHandler(*countclients*++,client)).start();// стартуем новый поток обработки сообщений нового клиента  
// сама обработка находится в методе run класса ClientHandler  
 }}  
 catch(Exception e){// в случае возникновения других не предвиденных ошибок  
 System.*out*.println("Error "+ e.toString());  
 }}}