package l3;  
import java.net.\*;  
import java.io.\*;  
class Server {  
 static int *countclients* = 0;//счетчик подключившихся клиентов  
 public static void main(String args[]) throws IOException {  
 ServerSocket sock = null;  
 InputStream is = null;  
 OutputStream os = null;  
 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("=======================================");  
 System.*out*.println("Client " + *countclients* + " connected");  
 is = client.getInputStream(); //получили входной поток для чтения данных  
 os = client.getOutputStream();//получили выходной поток для записи данных  
 boolean flag = true;  
 while (flag) {  
 byte[] bytes = new byte[1024];  
 is.read(bytes); //чтение иформации, посланной клиентом, из вхоного потока в массив bytes[]  
 String str = new String(bytes, "UTF-8"); // переводим тип byte в тип String  
 String[] words = str.split(" "); // разбиваем строку на подстроки пробелами  
 String m = ""; //переменнная,в которую будут записываться числа делящиеся на 3  
 //bytes = new byte[1024];  
 boolean choose ;  
 String first=" ",second=" ";  
 first=words[1];  
 second=words[0];  
 choose = words[0].equals(words[1]);  
 for (int i = 0; i < words.length - 1; i++) {  
 System.*out*.println("клиент прислал строку " + words[i]);  
  
  
  
 }  
 if (choose == true)  
 m += "True"; // записывают числа,которые деляется на 3  
  
 else m += "False";  
 System.*out*.println(m);  
  
  
  
  
  
 bytes = m.getBytes();  
 os.write(bytes); // отправляем клиенту информацию  
 }  
 }  
 } catch (Exception e) {  
 System.*out*.println("Error " + e.toString());  
 } finally {  
 is.close();//закрытие входного потока  
 os.close();//закрытие входного потока  
 sock.close();//закрытие сокета, выделенного для работы с подключившимся клиентом  
 System.*out*.println("Client " + *countclients* + " disconnected");  
 }  
 }}

package l3;  
  
import java.awt.Button;  
import java.awt.Frame;  
import java.awt.Label;  
import java.awt.TextArea;  
import java.awt.TextField;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.awt.event.WindowEvent;  
import java.awt.event.WindowListener;  
import java.io.\*;  
import java.net.\*;  
class Client extends Frame implements ActionListener, WindowListener {  
 TextField tf, tf1, tf2, tf3, tf4, tf5, tf6;  
 TextArea ta;  
 Label la, la1, la2, la3, la4;  
 Socket sock = null;  
 InputStream is = null;  
 OutputStream os = null,os2=null;  
 public static void main(String args[]) {  
 Client c = new Client();  
 c.GUI();  
 }  
 private void GUI() {  
// super("Клиент");  
 setTitle("КЛИЕНТ");  
 tf = new TextField("127.0.0.1");//ip adress клиента  
 tf1 = new TextField("1024");// port клиента  
 tf2 = new TextField();  
 tf3 = new TextField();  
 tf4 = new TextField();  
 tf5 = new TextField();  
 tf6 = new TextField();  
 ta = new TextArea();  
 la = new Label("IP ADRESS");  
 la1 = new Label("port");  
 la2 = new Label("sending date");  
 la3 = new Label("result ");  
 la4 = new Label(" ");  
 Button btn = new Button("connect ");  
 Button btn1 = new Button("send ");  
 tf.setBounds(200, 50, 70, 25);  
 tf1.setBounds(330, 50, 70, 25);  
 tf2.setBounds(130, 200, 100, 25);  
 tf3.setBounds(240, 200, 100, 25);  
 // tf4.setBounds(270, 200, 50, 25);  
 //tf5.setBounds(330, 200, 50, 25);  
 // tf6.setBounds(390, 200, 50, 25);  
 ta.setBounds(150, 300, 150, 100);  
 btn.setBounds(50, 50, 70, 25);  
 btn1.setBounds(50, 200, 70, 25);  
 la.setBounds(130, 50, 150, 25);  
 la1.setBounds(280, 50, 150, 25);  
 la2.setBounds(200, 150, 150, 25);  
 la3.setBounds(160, 250, 150, 25);  
 la4.setBounds(600, 10, 150, 25);  
 add(tf);  
 add(tf1);  
 add(tf2);  
 add(tf3);  
 // add(tf4);  
 // add(tf5);  
 // add(tf6);  
 add(btn);  
 add(btn1);  
 add(ta);  
 add(la);  
 add(la1);  
 add(la2);  
 add(la3);  
 add(la4);  
 setSize(600, 600);  
 setVisible(true);  
 addWindowListener(this);  
 btn.addActionListener(al);  
 btn1.addActionListener(this);  
 tf2.getText();  
 tf3.getText();  
 tf4.getText();  
 tf5.getText();  
 tf6.getText();  
 }  
 public void windowClosing(WindowEvent we) {  
 if (sock != null && !sock.isClosed()) { // если сокет не пустой и сокет открыт  
 try {  
 sock.close(); // сокет закрывается  
 } catch (IOException e) {  
 }  
 }  
 this.dispose();  
 }  
 public void windowActivated(WindowEvent we) {} ;  
 public void windowClosed(WindowEvent we) {};  
 public void windowDeactivated(WindowEvent we) {};  
 public void windowDeiconified(WindowEvent we) {} ;  
 public void windowIconified(WindowEvent we) {};  
 public void windowOpened(WindowEvent we) { } ;  
 public void actionPerformed(ActionEvent e) {  
 if (sock == null) {  
 return;  
 }  
 try {  
 is = sock.getInputStream(); // входной поток для чтения данных  
 os = sock.getOutputStream();// выходной поток для записи данных  
 // os2 = sock.getOutputStream();// выходной поток для записи данных  
  
 String numbers = ""; //перменная,в которую записываются введенные числа  
 numbers += tf2.getText() + " ";  
 numbers += tf3.getText() + " ";  
 // numbers += tf4.getText() + " ";  
 // numbers += tf5.getText() + " ";  
 // numbers += tf6.getText() + " ";  
 os.write(numbers.getBytes()); // отправляем введенные данные. Тип string переводим в byte  
 //os2.write(numbers.getBytes()); // отправляем введенные данные. Тип string переводим в byte  
 byte[] bytes = new byte[1024];  
 is.read(bytes); //получаем назад информацию,которую послал сервер  
 String str = new String(bytes, "UTF-8"); // переводим тип byte в String  
 // String[] n = str.split(" "); // разбиваем строку на подстроки пробелами  
 // for (int i = 0; i < n.length - 1; i++) {  
 ta.append(str); // в text area записываем полученные данные  
 // }  
 } catch (Exception ex) {  
 ex.printStackTrace();  
 } finally {  
 try {  
 os.close();//закрытие выходного потока  
 is.close();//закрытие входного потока  
 sock.close();//закрытие сокета, выделенного для работы с сервером  
 } catch (IOException e1) {  
 e1.printStackTrace();  
 }  
 }  
 }  
  
 public void actionPerformed2(ActionEvent e) {}  
 ActionListener al = new ActionListener() { //событие на нажатие кнопки  
 @Override  
 public void actionPerformed(ActionEvent arg0) {  
 try {  
 sock = new Socket(InetAddress.*getByName*(tf.getText()), Integer.*parseInt*(tf1.getText()));  
 //создается сокет по ip адрессу и порту  
 } catch (NumberFormatException e) {  
 } catch (UnknownHostException e) {  
 } catch (IOException e) {  
 }} };}