**Quiz for Java Programming** (Tutor: cyd@bupt.edu.cn)

Unit 09 Network Programming

1. Two protocols in Transport Layer are TCP and UDP
2. Transport layer addressing includes port and ip address
3. Socket is the endpoint in an inter-process (application) communication
4. When using stream socket in server side application, we must create a serveSocket

object to listen on a specified port

1. When using stream socket in client side application, we should create a socket

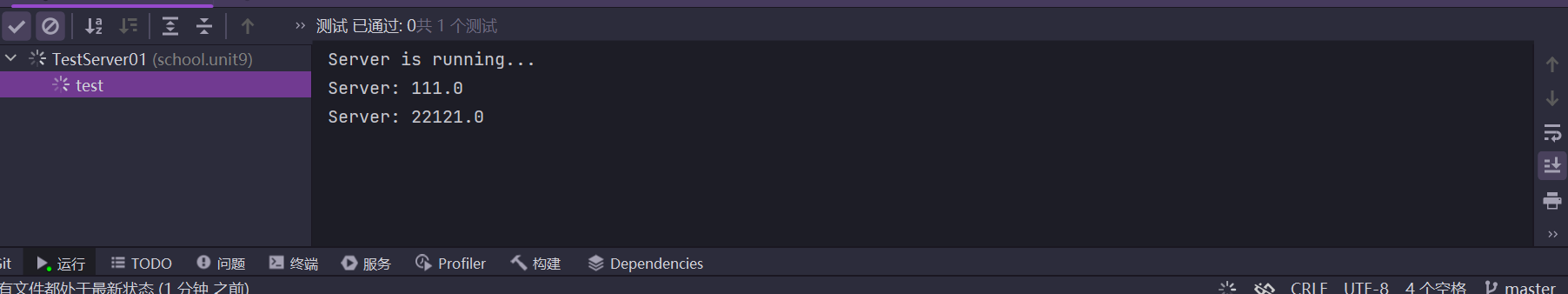
object to connect to the server with specified server ip addr and port

1. The Datagram socket does not establish a dedicated connetction channel between a client and a server
2. True or False:
   1. The Datagram socket provide a reliable transmission channel F
   2. The Stream socket provide a lossless transmission channel T
3. Datagram packet provide a connectionless packet delivery service
4. When using Datagram socket in server side application, the programmer must create a

DatagramSocket object and when using datagram socket in client side, the programmer must create a DatagramSocket object

1. Create a TCP server which listens on 6666 port and waiting to receive a double number; then create a TCP client which send a double number to the server. When the server receives the double number, print it out.





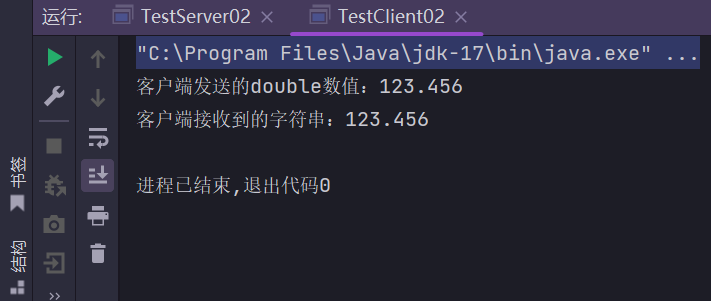
Server:

package school.unit9**;**import java.io.BufferedReader**;**import java.io.IOException**;**import java.io.InputStreamReader**;**import java.io.OutputStream**;**import java.net.InetAddress**;**import java.net.ServerSocket**;**import java.net.Socket**;**import java.net.UnknownHostException**;**import org.junit.Test**;**public class TestServer01 {  
  
 @Test  
 public void test() throws IOException {  
 //在6666端口等待连接,接受double类型的数据  
 try(ServerSocket serverSocket = new ServerSocket(6666)) {  
 System.*out*.println("Server is running...")**;** Socket socket = serverSocket.accept()**;** try(InputStreamReader socIsr = new InputStreamReader(socket.getInputStream())**;** BufferedReader socIn = new BufferedReader(socIsr)**;** //用于输出到socket  
 OutputStream socOut = socket.getOutputStream())  
 {  
 //读取double类型的数据  
 while (true) {  
 String line = socIn.readLine()**;** if (line == null) {  
 break**;** }  
 double num = Double.*parseDouble*(line)**;** System.*out*.println("Server: " + num)**;** }  
  
 }  
 catch (Exception e) {  
 e.printStackTrace()**;** }  
 } catch (UnknownHostException e) {  
 e.printStackTrace()**;** }  
  
  
  
  
  
  
 }  
  
}

Client:

package school.unit9**;**import java.io.BufferedReader**;**import java.io.IOException**;**import java.io.InputStreamReader**;**import java.io.OutputStream**;**import java.net.InetAddress**;**import java.net.Socket**;**import org.junit.Test**;**public class TestClient01 {  
 public static final String *HOST* = "127.0.0.1"**;** public static final int *PORT* = 6666**;** @Test  
 public void testClient(){  
 //建立连接，发送double类型的数据  
  
 try(Socket socket = new Socket(*HOST***,** *PORT*)) {  
 try(  
 //从键盘获取输入流  
 BufferedReader sysIn = new BufferedReader(  
 new InputStreamReader(System.*in*))**;** //从Socket获取输入流  
 BufferedReader socIn = new BufferedReader(  
 new InputStreamReader(socket.getInputStream()))**;** //从Socket获取输出流  
 OutputStream socOut = socket.getOutputStream()**;** )  
 {  
 //读取double类型的数据  
 while (true) {  
 String line = sysIn.readLine()**;** if (line == null) {  
 break**;** }  
 socOut.write((line + "\n").getBytes())**;** socOut.flush()**;** }  
  
 }  
 } catch (IOException e) {  
 throw new RuntimeException(e)**;** }  
  
 }  
  
}

1. Based on Quiz 10, when the server receives the double number, convert the number to string and send the string back to the client. When the client receives the string (NOT String object), print it out.

结果: 

Server端

package school.unit9**;**import java.io.\***;**import java.net.\***;**public class TestServer02 {  
 public static void main(String[] args) {  
 try {  
 // 创建服务器端Socket，监听6666端口  
 ServerSocket serverSocket = new ServerSocket(6666)**;** System.*out*.println("服务器端已启动，等待客户端连接...")**;** // 等待客户端连接  
 Socket clientSocket = serverSocket.accept()**;** System.*out*.println("客户端已连接...")**;** // 获取输入流，用于接收客户端发送的数据  
 DataInputStream inputStream = new DataInputStream(clientSocket.getInputStream())**;** // 读取客户端发送的double数值  
 double receivedNumber = inputStream.readDouble()**;** System.*out*.println("服务器端接收到的double数值：" + receivedNumber)**;** // 将double数值转换为字符串  
 String numberAsString = String.*valueOf*(receivedNumber)**;** // 获取输出流，用于向客户端发送数据  
 DataOutputStream outputStream = new DataOutputStream(clientSocket.getOutputStream())**;** // 发送字符串给客户端  
 outputStream.writeUTF(numberAsString)**;** System.*out*.println("服务器端已将字符串发送回客户端：" + numberAsString)**;** // 关闭连接  
 clientSocket.close()**;** serverSocket.close()**;** } catch (IOException e) {  
 e.printStackTrace()**;** }  
 }  
}

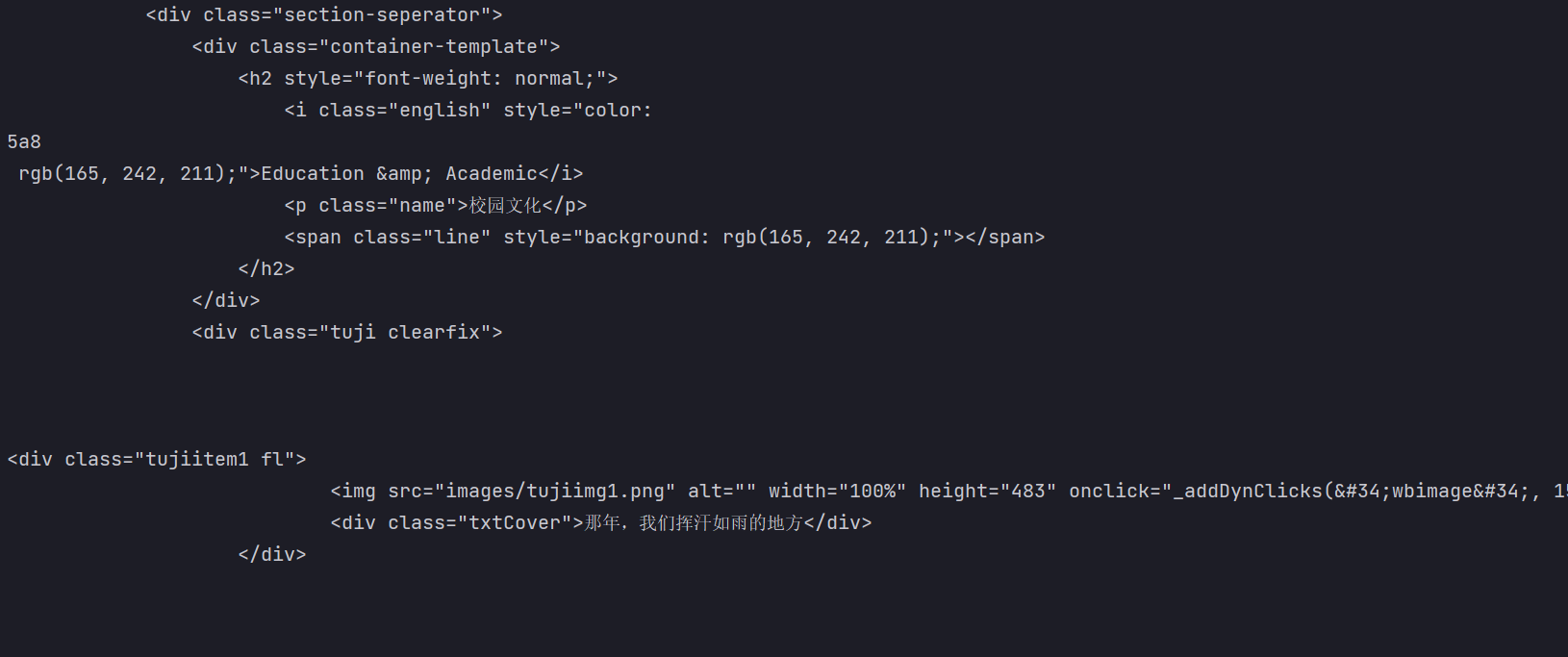
Client端

package school.unit9**;**import java.io.\***;**import java.net.\***;**public class TestClient02 {  
 public static void main(String[] args) {  
 try {  
 // 创建客户端Socket，连接服务器的6666端口  
 Socket clientSocket = new Socket("localhost"**,** 6666)**;** // 获取输出流，用于向服务器发送数据  
 DataOutputStream outputStream = new DataOutputStream(clientSocket.getOutputStream())**;** // 发送double数值给服务器  
 double numberToSend = 123.456**;** System.*out*.println("客户端发送的double数值：" + numberToSend)**;** outputStream.writeDouble(numberToSend)**;** // 获取输入流，用于接收服务器返回的数据  
 DataInputStream inputStream = new DataInputStream(clientSocket.getInputStream())**;** // 读取服务器返回的字符串  
 String receivedString = inputStream.readUTF()**;** System.*out*.println("客户端接收到的字符串：" + receivedString)**;** // 关闭连接  
 clientSocket.close()**;** } catch (IOException e) {  
 e.printStackTrace()**;** }  
 }  
}

1. Write a client program, using InetAddress to get the IP address of the web server

[www.bupt.edu.cn.](http://www.bupt.edu.cn/) Then read the information from the web server and print the information out.

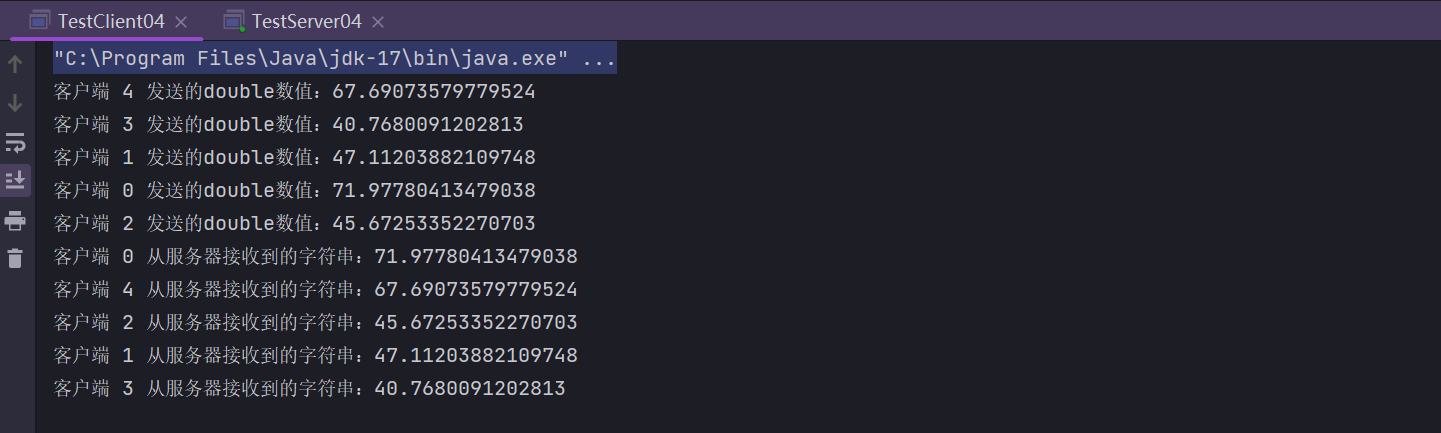
结果



Client端

package school.unit9**;**import java.net.\***;**import java.io.\***;**public class TestClient03 {  
 public static void main(String[] args) {  
 try {  
 // 获取 www.bupt.edu.cn 的 IP 地址  
 InetAddress ipAddress = InetAddress.*getByName*("www.bupt.edu.cn")**;** System.*out*.println("www.bupt.edu.cn 的 IP 地址：" + ipAddress.getHostAddress())**;** // 建立与服务器的连接  
 Socket socket = new Socket(ipAddress**,** 80)**;** // 获取输出流，用于向服务器发送 HTTP 请求  
 OutputStream outputStream = socket.getOutputStream()**;** PrintWriter writer = new PrintWriter(outputStream)**;** // 发送 HTTP 请求  
 writer.println("GET / HTTP/1.1")**;** writer.println("Host: www.bupt.edu.cn")**;** writer.println("Connection: close")**;** writer.println()**;** // 刷新缓冲区，确保请求被发送  
 writer.flush()**;** // 获取输入流，用于读取服务器的响应  
 InputStream inputStream = socket.getInputStream()**;** BufferedReader reader = new BufferedReader(new InputStreamReader(inputStream))**;** // 读取并打印服务器的响应信息  
 String line**;** while ((line = reader.readLine()) != null) {  
 System.*out*.println(line)**;** }  
  
 // 关闭连接  
 socket.close()**;** } catch (UnknownHostException e) {  
 System.*err*.println("无法解析主机名")**;** e.printStackTrace()**;** } catch (IOException e) {  
 System.*err*.println("与服务器通信时发生错误")**;** e.printStackTrace()**;** }  
 }  
}

1. Based on Quiz 11, modify the server program. When receiving the double number, the server sleeps for 3 seconds and then return a String object. The client receives the String object and print it out. Then modify the server program (using thread) to serve multiple clients concurrently.



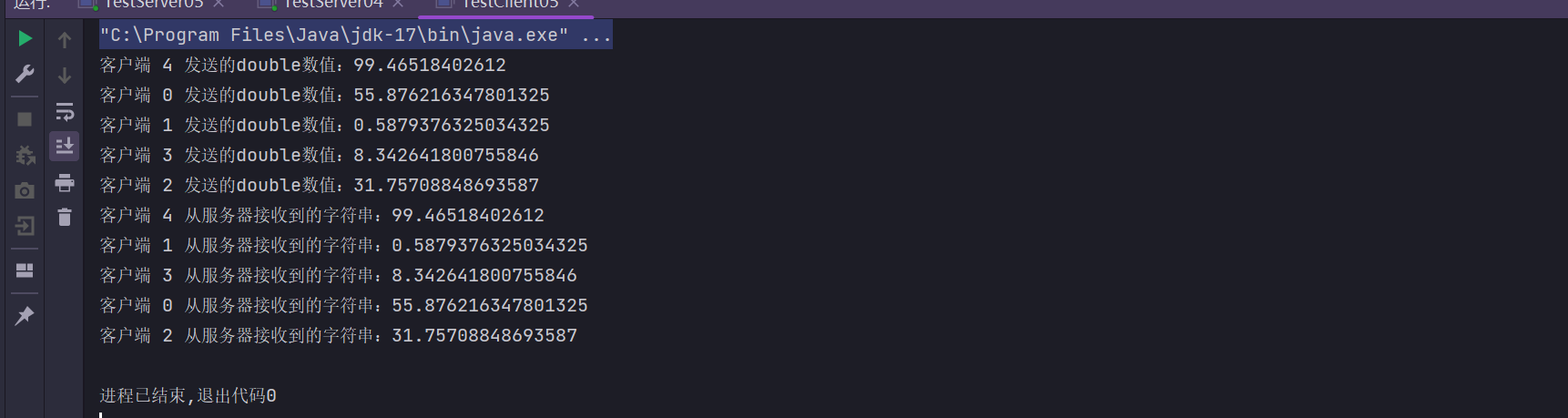
Client端:

package school.unit9**;**import java.io.\***;**import java.net.\***;**public class TestClient04 {  
 public static void main(String[] args) {  
 int numClients = 5**;** // 你可以调整模拟客户端的数量  
  
 for (int i = 0**;** i < numClients**;** i++) {  
 // 为每个客户端创建一个新线程  
 Thread clientThread = new Thread(new ClientTask(i))**;** clientThread.setName("客户端 " + i)**;** clientThread.start()**;** }  
 }  
  
 private static class ClientTask implements Runnable {  
 private int clientId**;** public ClientTask(int clientId) {  
 this.clientId = clientId**;** }  
  
 @Override  
 public void run() {  
 try {  
 // 创建客户端Socket，连接到localhost和端口6666的服务器  
 Socket clientSocket = new Socket("localhost"**,** 6666)**;** // 获取用于向服务器发送数据的输出流  
 DataOutputStream outputStream = new DataOutputStream(clientSocket.getOutputStream())**;** // 向服务器发送一个随机的double数值  
 double numberToSend = Math.*random*() \* 100**;** System.*out*.println("客户端 " + clientId + " 发送的double数值：" + numberToSend)**;** outputStream.writeDouble(numberToSend)**;** // 获取用于从服务器接收数据的输入流  
 DataInputStream inputStream = new DataInputStream(clientSocket.getInputStream())**;** // 读取服务器发送的字符串  
 String receivedString = inputStream.readUTF()**;** System.*out*.println("客户端 " + clientId + " 从服务器接收到的字符串：" + receivedString)**;** // 关闭客户端Socket  
 clientSocket.close()**;** } catch (IOException e) {  
 e.printStackTrace()**;** }  
 }  
 }  
}

Server端:

package school.unit9**;**import java.io.\***;**import java.net.\***;**public class TestServer04 {  
 public static void main(String[] args) {  
 try {  
 // 创建服务器端Socket，监听6666端口  
 ServerSocket serverSocket = new ServerSocket(6666)**;** System.*out*.println("服务器端已启动，等待客户端连接...")**;** //接受多个客户端的连接  
 while (true) {  
 // 等待客户端连接  
 Socket clientSocket = serverSocket.accept()**;** System.*out*.println("客户端已连接...")**;** // 创建线程来处理客户端请求  
 ClientHandler clientHandler = new ClientHandler(clientSocket)**;** new Thread(clientHandler).start()**;** }  
 } catch (IOException e) {  
 e.printStackTrace()**;** }  
 }  
  
 private static class ClientHandler implements Runnable {  
 private Socket clientSocket**;** public ClientHandler(Socket clientSocket) {  
 this.clientSocket = clientSocket**;** }  
  
 @Override  
 public void run() {  
 try {  
 // 获取输入流，用于接收客户端发送的数据  
 DataInputStream inputStream = new DataInputStream(clientSocket.getInputStream())**;** // 读取客户端发送的double数值  
 double receivedNumber = inputStream.readDouble()**;** System.*out*.println("服务器端接收到的double数值：" + receivedNumber)**;** // 暂停3秒  
 Thread.*sleep*(3000)**;** // 将double数值转换为字符串  
 String numberAsString = String.*valueOf*(receivedNumber)**;** // 获取输出流，用于向客户端发送数据  
 DataOutputStream outputStream = new DataOutputStream(clientSocket.getOutputStream())**;** // 发送字符串给客户端  
 outputStream.writeUTF(numberAsString)**;** System.*out*.println("服务器端已将字符串发送回客户端：" + numberAsString)**;** // 关闭连接  
 clientSocket.close()**;** } catch (IOException | InterruptedException e) {  
 e.printStackTrace()**;** }  
 }  
 }  
}

1. Based on Quiz 11, use UDP to implement program.



Client端

package school.unit9**;**import java.io.IOException**;**import java.net.\***;**import java.nio.ByteBuffer**;**public class TestClient05 {  
 public static void main(String[] args) {  
 int numClients = 5**;** // 你可以调整模拟客户端的数量  
  
 for (int i = 0**;** i < numClients**;** i++) {  
 // 创建一个新线程为每个客户端  
 Thread clientThread = new Thread(new ClientTask(i))**;** clientThread.start()**;** }  
 }  
  
 private static class ClientTask implements Runnable {  
 private int clientId**;** public ClientTask(int clientId) {  
 this.clientId = clientId**;** }  
  
 @Override  
 public void run() {  
 try {  
 // 创建客户端Socket  
 DatagramSocket clientSocket**;** try {  
 clientSocket = new DatagramSocket()**;** } catch (SocketException e) {  
 throw new RuntimeException(e)**;** }  
  
 // 获取服务器的IP地址和端口  
 InetAddress serverAddress = InetAddress.*getByName*("localhost")**;** int serverPort = 6666**;** // 发送一个随机的double数值给服务器  
 double numberToSend = Math.*random*() \* 100**;** System.*out*.println("客户端 " + clientId + " 发送的double数值：" + numberToSend)**;** // 将double数值转换为字节数组  
 byte[] sendData = ByteBuffer.*allocate*(8).putDouble(numberToSend).array()**;** // 创建发送数据包  
 DatagramPacket sendPacket = new DatagramPacket(sendData**,** sendData.length**,** serverAddress**,** serverPort)**;** // 发送数据包给服务器  
 clientSocket.send(sendPacket)**;** // 准备接收数据的缓冲区  
 byte[] receiveData = new byte[1024]**;** DatagramPacket receivePacket = new DatagramPacket(receiveData**,** receiveData.length)**;** // 接收服务器返回的数据  
 clientSocket.receive(receivePacket)**;** // 从接收的数据包中提取服务器返回的字符串  
 String receivedString = new String(receiveData**,** 0**,** receivePacket.getLength())**;** System.*out*.println("客户端 " + clientId + " 从服务器接收到的字符串：" + receivedString)**;** // 关闭客户端Socket  
 clientSocket.close()**;** } catch (IOException e) {  
 e.printStackTrace()**;** }  
 }  
 }  
}

Serve端

package school.unit9**;**import java.io.\***;**import java.net.\***;**import java.nio.ByteBuffer**;**public class TestServer05 {  
 public static void main(String[] args) {  
 try {  
 // 创建服务器端Socket，监听6666端口  
 DatagramSocket serverSocket = new DatagramSocket(6666)**;** System.*out*.println("UDP服务器端已启动，等待客户端连接...")**;** while (true) {  
 // 准备接收数据的缓冲区  
 byte[] receiveData = new byte[1024]**;** DatagramPacket receivePacket = new DatagramPacket(receiveData**,** receiveData.length)**;** // 接收客户端发送的数据  
 serverSocket.receive(receivePacket)**;** // 从接收的数据包中提取客户端地址和端口  
 InetAddress clientAddress = receivePacket.getAddress()**;** int clientPort = receivePacket.getPort()**;** // 读取客户端发送的double数值  
 double receivedNumber = ByteBuffer.*wrap*(receiveData).getDouble()**;** System.*out*.println("服务器端接收到的double数值：" + receivedNumber)**;** // 暂停3秒  
 Thread.*sleep*(3000)**;** // 将double数值转换为字符串  
 String numberAsString = String.*valueOf*(receivedNumber)**;** // 发送字符串给客户端  
 byte[] sendData = numberAsString.getBytes()**;** DatagramPacket sendPacket = new DatagramPacket(sendData**,** sendData.length**,** clientAddress**,** clientPort)**;** serverSocket.send(sendPacket)**;** System.*out*.println("服务器端已将字符串发送回客户端：" + numberAsString)**;** }  
 } catch (IOException | InterruptedException e) {  
 e.printStackTrace()**;** }  
 }  
}