和TCP方式存在很多的不同，最大的一个不同就是“无状态”。该特点指每次服务器端都收到信息，但是这些信息和连接无关，换句话说，也就是服务器端只是从信息是无法识别出是谁发送的，这样就要求发送信息时的内容需要多一些

- **DatagramSocket：用于发送或接收数据报包，“网络连接”**

当服务器要向客户端发送数据时，需要在服务器端产生一个DatagramSoc ket对象，在客户端产生一个DatagramSocket对象。服务器端的DatagramSoc ket将DatagramPacket发送到网络上，然后被客户端的DatagramSocket接收。

构造函数：

DatagramSocket() ：构造数据报套接字并将其绑定到本地主机上任何可用的端口。常用于客户端。

DatagramSocket(int port) ：创建数据报套接字并将其绑定到本地主机上的指定端口。建立连接时一般没必要，常用于服务器端。

常用方法：

Ø send(DatagramPacket p) ：从此套接字发送数据报包。

Ø receive(DatagramPacket p) ：从此套接字接收数据报包。

Ø close() ：关闭此数据报套接字。

- **DatagramPacket：数据容器(封包)的作用**

此类表示数据报包。 数据报包用来实现封包的功能。

常用方法：

Ø DatagramPacket(byte[] buf, int length) ：构造数据报包，用来接收长度为 length 的数据包。

Ø DatagramPacket(byte[] buf, int length, InetAddress address, int port) ：构造数据报包，用来将长度为 length 的包发送到指定主机上的指定端口号。

Ø getAddress() ：获取发送或接收方计算机的IP地址，此数据报将要发往该机器或者是从该机器接收到的。

Ø getData() ：获取发送或接收的数据。

Ø setData(byte[] buf) ：设置发送的数据。

Ø getLength() ：获取数据长度。

UDP通信编程基本步骤：

1. 创建客户端的DatagramSocket，创建时，定义客户端的监听端口。

2. 创建服务器端的DatagramSocket，创建时，定义服务器端的监听端口。

3. 在服务器端定义DatagramPacket对象，封装待发送的数据包。

4. 客户端将数据报包发送出去。

5. 服务器端接收数据报包。

UDP：对象的传递之客户端。 发送字节数组

先对象流包装字节数组流，在writeobjet序列化写入对象，再获取字节数组流中的字节数组。

```
1  import java.io.ByteArrayOutputStream;
2  import java.io.ObjectOutputStream;
3  import java.net.DatagramPacket;
4  import java.net.DatagramSocket;
5  import java.net.InetSocketAddress;
6
7  public class Client {
8      public static void main(String[] args) throws Exception {
9          //创建要发送的对象
10         Person person = new Person(18, "高淇");
11         ByteArrayOutputStream bos = new ByteArrayOutputStream();
12         ObjectOutputStream oos = new ObjectOutputStream(bos);
13         oos.writeObject(person);
```

```
        //获取字节数组流中的字节数组（我们要发送的数据）
        byte[] b = bos.toByteArray();
        //必须告诉数据报包要发到哪台计算机的哪个端口，发送的数据以及数据的长度
        DatagramPacket dp = new DatagramPacket(b,b.length,new
                                    InetSocketAddress("localhost",8999));
        //创建数据报套接字：指定发送信息的端口
        DatagramSocket ds = new DatagramSocket(9000);
        //发送数据报包
        ds.send(dp);
        //关闭资源
        oos.close();
        bos.close();
        ds.close();
    }
}
```

UDP：对象的传递之服务器端

```
import java.io.ByteArrayInputStream;
import java.io.ObjectInputStream;
import java.net.DatagramPacket;
import java.net.DatagramSocket;

public class Server {
    public static void main(String[] args) throws Exception {
        //创建数据报套接字：指定接收信息的端口
        DatagramSocket ds = new DatagramSocket(8999);
        byte[] b = new byte[1024];
        //创建数据报包，指定要接收的数据的缓存位置和长度
        DatagramPacket dp = new DatagramPacket(b, b.length);
        //接收客户端发送的数据报
        ds.receive(dp); // 阻塞式方法
        //dp.getData():获取客户端发送的数据，返回值是一个字节数组
        ByteArrayInputStream bis = new ByteArrayInputStream(dp.getData());
        ObjectInputStream ois = new ObjectInputStream(bis);
        System.out.println(ois.readObject());
        //关闭资源
        ois.close();
        bis.close();
        ds.close();
    }
}
```