各位听众，今日我将探讨JAVA网络编程这一主题。在深入JAVA网络编程之前，我们需对网络编程的基本概念进行了解。首先，我们需明白IP地址的概念。众所周知，IP地址主要分为IPv4和IPv6两大类。当前，互联网主要采用IPv4协议。IPv4地址由四个字节组成，而IPv6则由16个字节表示。此外，IPv4地址采用点分十进制格式，而IPv6则使用十六进制表示。

接下来，我们需要了解域名这一概念。简单来说，域名就是IP地址的映射。端口号在计算机网络中起到至关重要的作用，它主要用于标识计算机上的特定应用程序。常见的端口号包括Tomcat使用的8080端口。端口号的范围在0到6535之间，占用两个字节。

此外，我们还需要回顾一下网络通信协议的相关知识。OSi模型主要是一种理论模型，但在实际应用中并不常用。因此，我们重点关注TCP/IP模型。TCP/IP模型将OSi模型简化为四层。

那么tcp协议在使用之前必须先建立tcp连接，形成传输数据通道。从前必须要进行三次握手。优点就是比较可靠，可以进行大量数据传输。缺点呢就是需要释放连接效率比较低。

UDP协议会将数据重装为数据包。而且不需要进行连接。但是数据包的大小现在64k以内。优点啊就是通信速度比较快，缺点呢就是不可靠。

下面简单介绍一下tcp的三次握手和四次挥手。

三次握手（Three-Way Handshake）：

场景：

假设两个人通过书信进行沟通。一个人发送消息（A），另一个人接收并确认消息（B），然后再次发送确认消息（A）。

第一步（SYN）：

A：“嗨，你在吗？我想和你通信。”

A向B发送一个带有SYN（同步）标志的消息。

第二步（SYN-ACK）：

B：“嗨，我在的，我也想和你通信。”

B接收到A的消息后，回复一个带有SYN和ACK（确认）标志的消息，表明确认收到A的请求，并表示自己也想和A通信。

第三步（ACK）：

A：“太好了，我们可以开始通信了。”

A收到B的回复后，向B发送一个确认消息，表示已收到B的确认，连接建立成功，可以开始通信。

这个三次握手确保了通信双方都愿意建立连接，理解彼此的状态，并准备好传输数据。

四次挥手（Four-Way Handshake）：

场景：

两个人结束书信沟通。一个人发送结束消息（A），另一个人确认收到消息并准备结束（B），然后再次发送确认消息（A）并最终确认结束（B）。

第一步（FIN）：

A：“我不想再和你通信了。”

A向B发送一个带有FIN（结束）标志的消息，表示A不再发送数据。

第二步（ACK）：

B：“好的，我知道了，但是我还有一些数据要传输完。”

B接收到A的结束消息后，发送一个确认消息（ACK），表示已经收到A的结束请求。

第三步（FIN）：

B：“现在我也准备结束了。”

B在完成数据传输后，向A发送一个带有FIN标志的消息，表示自己也不再发送数据。

第四步（ACK）：

A：“好的，我知道了，我们可以结束通信了。”

A收到B的结束消息后，向B发送确认消息（ACK），表示知道B也已经结束了，连接断开。

四次挥手确保了双方都知道彼此不再发送数据，并且完成了所有必要的关闭操作，安全地结束了连接。

下面介一下Inetaddress类。

1.获取本机InetAddress对象getLocalHost

2.根据指定主机名/域名获取ip地址对象getByName

3.获取InetAddress对象的主机名getHostName

4.获取InetAddress对象的地址getHostAddress

dos控制台下执行

1.netstat-an可以查看当前主机网络情况，包括端口监听情况和网络连接情况

说明：

(1)Listening表示某个端口在监听

(2)如果有一个外部程序(客户端)连接到该端口，就会显示一条连接信息.

那么两台主机之间要实现网络通信。就需要借助socket来实现。也就是说，socket实现了传输层的协议。Socket这个英文单词。翻译过来的意思是有插口的意思。也就是说我们可以把网络通信看成电流的呢，就是电流两端的插口。

我们利用socket可以连接远程机器2发送数据3接受数据4关闭连接5绑定端口6监听入站数据7在所绑定得端口上接受来自远程的连接

Java目前已经支持了tcp和udp的Socket编程已经暂时还不不支持icmp。Socket对程序员掩盖了网络底层实验的细节，比如说纠错，重传，网络地址等。啊？它通过IP和端口得出一个套接字。这样就可以完全分辨网络上运行的程序。

socket允许程序把网络连接当成一个流，数据在两个socket间通过Io流传输。

Socket可以理解为数据通道两端的插头。但我们需要从网络上读取和写入数据的时候可以通过。Get的get input stream和get output stream方法去读取和去读取读取数据通道上的数据。或者把出去处理完通过输出流写入到数据通道上。

那么tcp编程的方式是如何实现网络通信呢？

服务器端先进行一个监听。等待客户端的连接。连接成功以后，socket可以调用输出流。把数据传入到数据通道里。把数据发送给服务端。服务器这边儿就会通过Socket去调用输入流读取数据。都写完之后，如果还要把数据发送给客户端，那么就继续调用输出流。把数据传输到网络通道里。然后客户端这边在用输入去读取。最后我们需要去关闭流和Socket。

下面我们看一个tcp编程的简单案例。第一案例我们使用字节流的方式去实现

在这个案例，服务端监听8080端口。客户端向服务端发送一个请求信息hello。然后退出。服务端在收到信息之后，就退出。

首先我们分析一下思路：服务端在本地的8080端口进行监听。等待连接。但没有客户端连接8080端口时手机会阻塞。等待连接。客户端通过IP地址和端口号区连接。服务端连接完之后会生成一个socket那么socket的就可以通过输出流把数据写入到数据通道里。之后服务端通过输流就可以读取数据通道里的数据并显示出来。

我们这里在服务端New一个Serversocket在8080端口进行监听。这里要求的前提是本机的8080端口没有被占用。如果被占用会抛出异常。

下面Seversoccket调用这个accept方法。当没有客户端连接8080端口时，程序会在这里阻塞。当口端连接8080端口的时候，这里会返回一个Socket的对象。serversocket可以对应多个socket,可以实现。一台服务器可以和多客户端进行连接。

客户端第一步要New soccket传入服务端的IP地址和端口号,连接成功就返回一个socket。之后通过输出流就可以把数据发送过去。服务端调用输入流就可以拿到数据。之后我们要关闭对应的输入和输出流和socket对象。这里客户端。在发送请求的时候，其实也是有一个端口号，但它是随机的。

第二个案例。我们要求客户端，服务端在接收到消息之后，发送一个消息到客户端。前面的几个步骤都一样.只不过我们在服务端获取到数据之后，继续调用输出流。返回消息。客户端呢也是用的输入流读取数据。不过这里要特别注意的是,无论是客户端还是服务端，在调用输出流写入数据时，都必须设置对应的结束标记。要不然客户端，服务端都不知道对方什么时候发送完数据？程序就会阻塞在这里。

我们还可以通过字符流的方式进行网络通信,这时候我们需要把对应的input stream转换成 buffer的reader用readline方法去读取数据。输出数据的时候，同样要进行输出流的转换。但是这里的结束标记需使用new line方法方法来替代。还需要注意的是，这里需要手动刷新输出流。

下面我们看一下udp网络通信编程。

1.类DatagramSocket和DatagramPacket[数据包/数据报]实现了基于UDP协议网络程序。

2.UDP数据报通过数据报套接字DatagramSocket发送和接收，系统不保证UDP数据报一定能够安全送到目的地，也不能确定什么时候可以抵达。

3.DatagramPacket对象封装了UDP数据报，在数据报中包含了发送端的IP地址和端口号以及接收端的IP地址和端口号。

4.UDP协议中每个数据报都给出了完整的地址信息，因此无须建立发送方和接收方的连接

1.核心的两个类/对象DatagramSocket与DatagramPacket

2.建立发送端，接收端(没有服务端和客户端概念)

3.发送数据前，建立数据包/报DatagramPacket对象

4.调用DatagramSocket的发送、接收方法

5.关闭DatagramSocket

下午我们看一个案例。这个案例有一个发送端，一个接收端。同样也是发送端发送数据给接收单。接收端收到数据回复数据给发送端。然后退出。发送端在收到回复数据后，同样退出。

服务端的代码。通过DatagramSocket来监听端口。通过new一个DatagramPacket对象啊来封装数据。然后我们的DatagramSocket调用receive方法会把数据填充到DatagramPacketpacket对象里。之后我们可以对DatagramPacketpacket进行拆包的操作。