

Java

Networking

TCP

- TCP stands for Transmission Control Protocol
- TCP is connection-oriented
- It provides reliability
- What is Server and Client?
 - A server is a piece of software which advertises and then provides some service on request
 - A client is a piece of software (usually on a different machine) which makes use of some service

TCP Sockets

- Two types of TCP Sockets
- ***ServerSocket***
 - ServerSocket is used by servers so that they can accept incoming connections from client
- ***Socket***
 - Socket is used by clients who wish to establish a connection to a (remote) server

Scenario

Client

```
Socket s = new Socket  
("192.168.0.63", 22222);
```



```
s.getInputStream();  
s.getOutputStream();
```

Server (192.168.0.63)

```
ServerSocket ss=new  
ServerSocket(22222);
```



```
Socket cs = ss.accept();
```



```
cs.getInputStream();  
cs.getOutputStream();
```



TCP Sockets Code

- **Packages:**
 - **tcpsimple** (no threading)
 - **tcpstring** (multithreading, string send and receive)
 - **tcpobject** (multithreading, object send and receive)
 - **tcpdiff** (multithreading, the server sends messages to multiple clients)
 - **tcpforward** (multithreading, the server forwards messages between multiple clients)

UDP

- UDP stands for User Datagram Protocol
- UDP is not connection-oriented
- It does not provide reliability
- It sends and receives packets known as Datagram

Datagram Packet & Socket

- One type of Packet and one type of Socket
- ***DatagramPacket***
 - Used to encapsulate Datagram
- ***DatagramSocket***
 - DatagramSocket is used by both server and client to receive DatagramPacket
- ***Example: DatagramServer.java, DatagramClient.java***

InetAddress

- Java has a class ***java.net.InetAddress*** which abstracts network addresses
- Major methods
 - `getLocalHost()`
 - `getByAddress()`
 - `getByName()`
- ***Example: HostInfo.java, AddressGenerator.java, Resolver.java***

HttpURLConnection

- Java provides a class ***java.net.HttpURLConnection*** that provides support for HTTP connections
- You can obtain HttpURLConnection by calling `openConnection()` on URL object
- You must cast the result to HttpURLConnection
- You can then read/write from/to the connection
- ***Example: TestHttpURL.java***

HttpClient

- Java 11 introduced a new networking package **java.net.http** (aka HTTP Client API)
 - It provides enhanced networking support for HTTP clients
 - Superior alternative to **HttpURLConnection**
- **Classes:** HttpClient, HttpRequest, HttpResponse
 - Create an instance of HttpClient
 - Construct an HttpRequest, send it by HttpClient's send()
 - The response is returned by send() from which the headers and response body can be obtained
- **Example:** *TestHttpClient.java*