Head First Java

Chapter-15:
Make a Connection

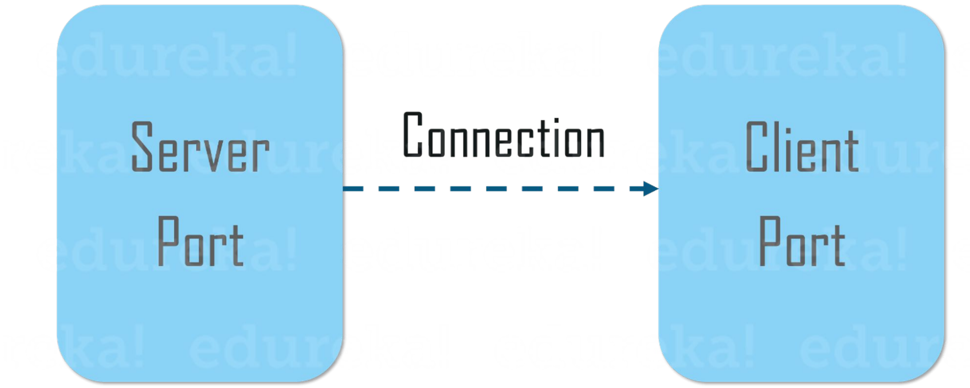
Upcode Software
Engineer Team

-2023-

CONTENT

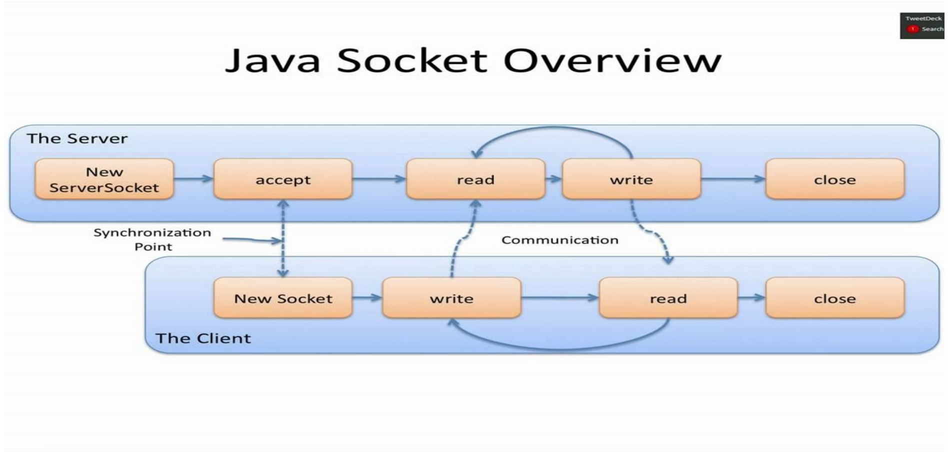
What is Socket ? (1/n)

- To connect to another machine, we need a Socket connection. A Socket (java.net.Socket class) is an object that represents a network connection between two machines. What's a connection? A relationship between two machines, where two pieces of software know about each other



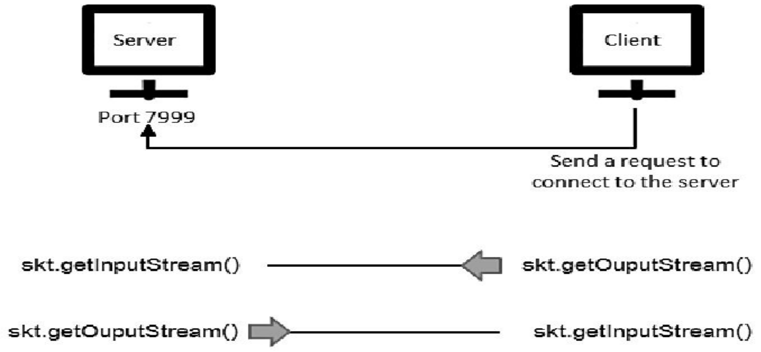
What is Socket ? (2/n)

- The term *socket programming* refers to writing programs that execute across multiple computers in which the devices are all connected to each other using a network.
- There are two communication protocols that we can use for socket programming: **User Datagram Protocol (UDP) and Transfer Control Protocol (TCP)**.



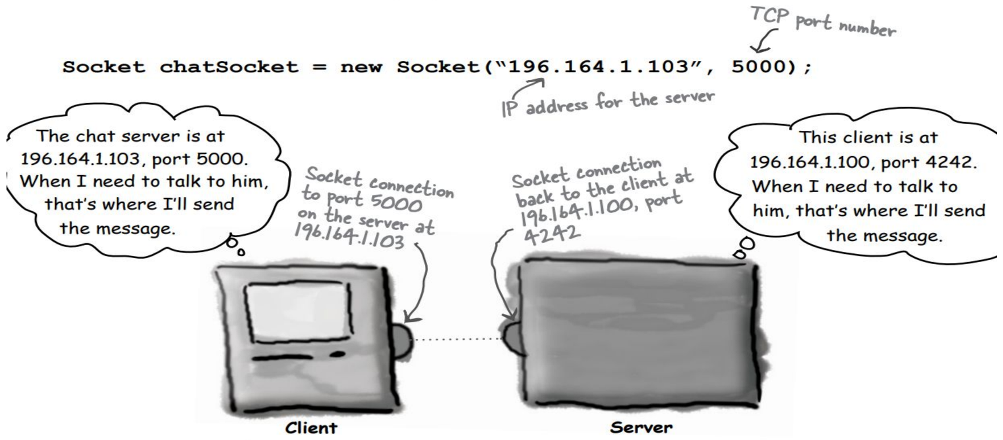
What is Socket ? (3/n)

- The main difference between the two is that UDP is connection-less, meaning there’s no session between the client and the server, while TCP is connection-oriented, meaning an exclusive connection must first be established between the client and server for communication to take place.
- This tutorial presents **an introduction to sockets programming over TCP/IP** networks, and demonstrates how to write client/server applications in Java. UDP isn’t a mainstream protocol, and as such, might not be encountered often.



Make a network Socket connection. (1/n)

- To make a Socket connection, you need to know two things about the server: who it is, and which port it's running on. In other words, IP address and TCP port number.



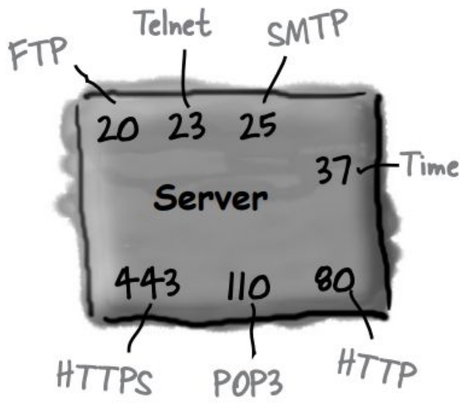
```
Socket chatSocket = new Socket("196.164.1.103", 5000);
```

TCP port number
IP address for the server
The chat server is at 196.164.1.103, port 5000. When I need to talk to him, that's where I'll send the message.
Socket connection to port 5000 on the server at 196.164.1.103
Socket connection back to the client at 196.164.1.100, port 4242
This client is at 196.164.1.100, port 4242. When I need to talk to him, that's where I'll send the message.
Client
Server

A Socket connection means the two machines have information about each other, including network location (IP address) and TCP port.

Make a network Socket connection. (2/n)

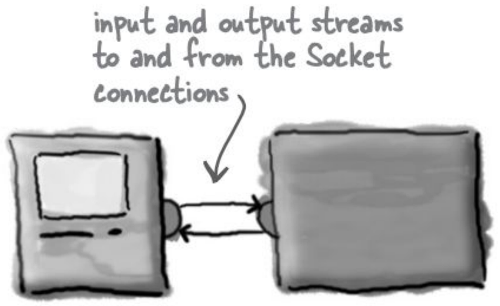
- A TCP port is just a number. A 16-bit number that identifies a specific program on the server.
- Well-known TCP port numbers for common server applications.
- The TCP port numbers from 0 to 1023 are reserved for use them for your own server programs!
- You have 65536 of them on a server (0 - 65535).

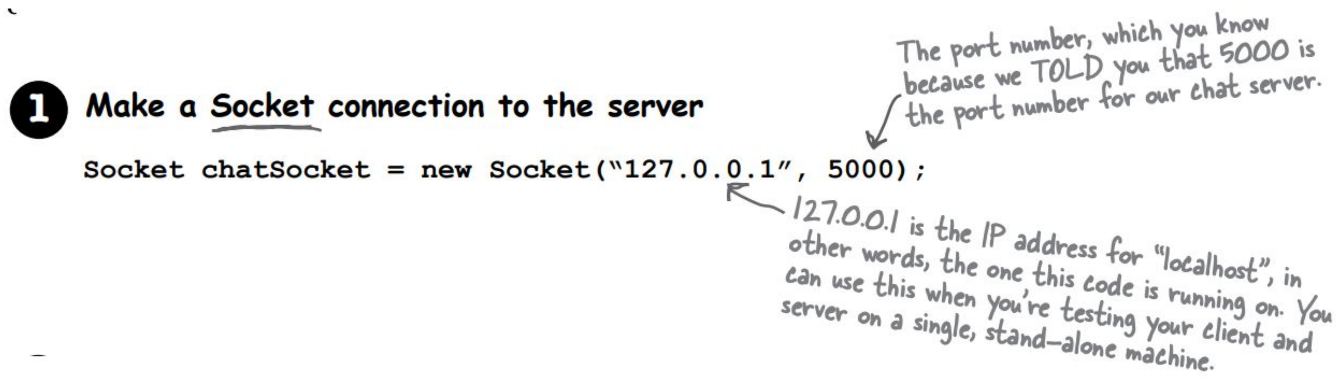


A server can have up to 65536 different server apps running, one per port.

To read data from a Socket. (1/n)

- To communicate over a Socket connection, you use streams.





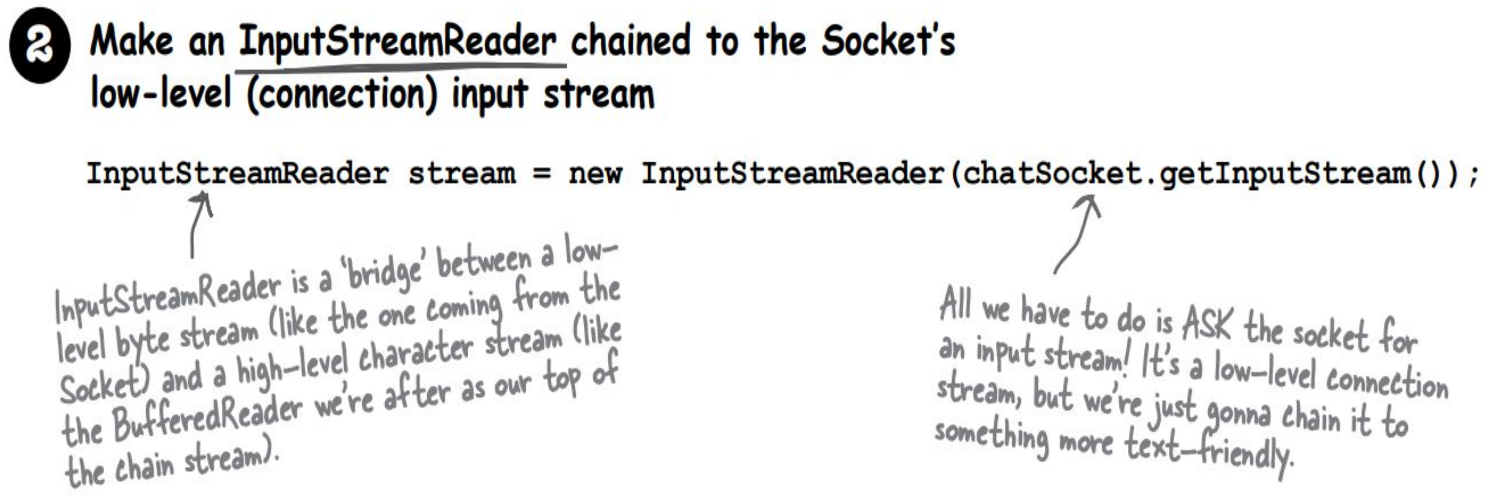
1 **Make a Socket connection to the server**

```
Socket chatSocket = new Socket("127.0.0.1", 5000);
```

The port number, which you know because we TOLD you that 5000 is the port number for our chat server.

127.0.0.1 is the IP address for "localhost", in other words, the one this code is running on. You can use this when you're testing your client and server on a single, stand-alone machine.

To read data from a Socket. (2/n)



2 Make an InputStreamReader chained to the Socket's low-level (connection) input stream

```
InputStreamReader stream = new InputStreamReader(chatSocket.getInputStream());
```

InputStreamReader is a 'bridge' between a low-level byte stream (like the one coming from the Socket) and a high-level character stream (like the BufferedReader we're after as our top of the chain stream).

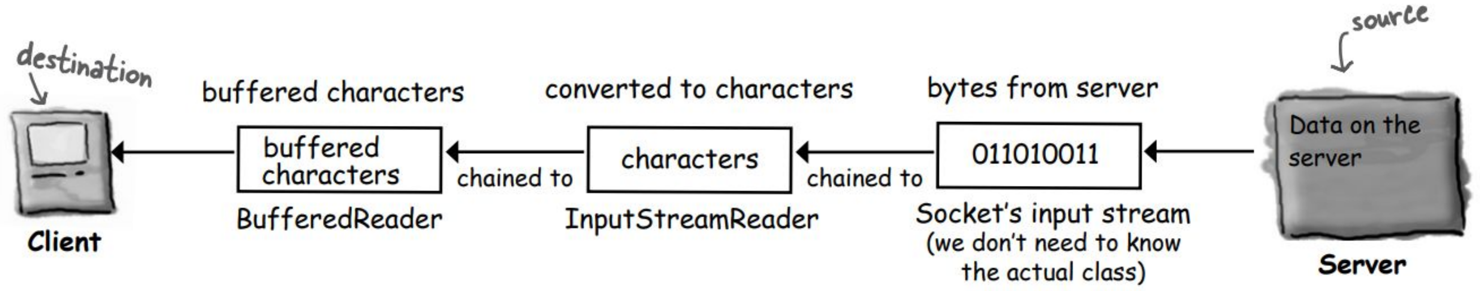
All we have to do is ASK the socket for an input stream! It's a low-level connection stream, but we're just gonna chain it to something more text-friendly.

To read data from a Socket. (3/n)

3 Make a BufferedReader and read!

Chain the BufferedReader to the InputStreamReader (which was chained to the low-level connection stream we got from the Socket.)

```
BufferedReader reader = new BufferedReader(stream);
String message = reader.readLine();
```



Thank you!

Presented by **Jahongir Sherjonov**

(jakhongirsherjonov@gmail.com)