AARHUS UNIVERSITET

Software Engineering and Architecture

Networking in Java:

Sockets

Networking from Code

- The most fundamental abstraction of network is the **Socket**

Definition:

A *socket* is one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent to.

- So, your computer has an IP address, and you then create a Socket with a specific port number
  - And you have something you can address from a client

The Socket Abstraction

- Once a socket is established you can get an input- and output stream

`InputStream`	`getInputStream()` Returns an input stream for this socket.

`OutputStream`	`getOutputStream()` Returns an output stream for this socket.

- That is, you can *read* and *write* to the stream, just as you would read and write to a File!

Example from Java Tutorial

- The Echo Server
  - Just returns anything you send to it (Booooring...)
- Start the server on port 37000

```
csdev@m31:~/proj/frsproject/echo$ java EchoServer
Usage: java EchoServer <port number>
csdev@m31:~/proj/frsproject/echo$ java EchoServer 37000
```

- And a client

```
csdev@m31:~/proj/frsproject/echo$ java EchoClient
Usage: java EchoClient <host name> <port number>
csdev@m31:~/proj/frsproject/echo$ java EchoClient localhost 37000
Dette er en test
echo: Dette er en test
Sooo boring...
echo: Sooo boring...
But - anyway it is working...
echo: But - anyway it is working...
```

Echo

- Echo Server

```java
import java.net.*;
import java.io.*;

public class EchoServer {
    public static void main(String[] args) throws IOException {

        if (args.length != 1) {
            System.err.println("Usage: java EchoServer <port number>");
            System.exit(1);
        }

        int portNumber = Integer.parseInt(args[0]);

        try (
            ServerSocket serverSocket =
                new ServerSocket(Integer.parseInt(args[0]));
            Socket clientSocket = serverSocket.accept();
            PrintWriter out =
                new PrintWriter(clientSocket.getOutputStream(), true);
            BufferedReader in = new BufferedReader(
                new InputStreamReader(clientSocket.getInputStream()));
        ) {
            String inputLine;
            while ((inputLine = in.readLine()) != null) {
                out.println(inputLine);
            }
        } catch (IOException e) {
            System.out.println("Exception caught when trying to listen on port "
                + portNumber + " or listening for a connection");
            System.out.println(e.getMessage());
        }
    }
}
```

Echo

- Echo Client

```java
import java.io.*;
import java.net.*;

public class EchoClient {
    public static void main(String[] args) throws IOException {

        if (args.length != 2) {
            System.err.println(
                "Usage: java EchoClient <host name> <port number>");
            System.exit(1);
        }

        String hostName = args[0];
        int portNumber = Integer.parseInt(args[1]);

        try (
            Socket echoSocket = new Socket(hostName, portNumber);
            PrintWriter out =
                new PrintWriter(echoSocket.getOutputStream(), true);
            BufferedReader in =
                new BufferedReader(
                    new InputStreamReader(echoSocket.getInputStream()));
            BufferedReader stdIn =
                new BufferedReader(
                    new InputStreamReader(System.in))
        ) {
            String userInput;
            while ((userInput = stdIn.readLine()) != null) {
                out.println(userInput);
                System.out.println("echo: " + in.readLine());
            }
        } catch (UnknownHostException e) {
            System.err.println("Don't know about host " + hostName);
            System.exit(1);
        } catch (IOException e) {
            System.err.println("Couldn't get I/O for the connection to " +
                hostName);
            System.exit(1);
        }
    }
}
```

- Note the asymmetry
  - Client: ‘Socket’
  - Server: ‘ServerSocket’

Exercise

- Now, you have enough to build Fortnite or LoL ☺
  - Tack a bit of graphics on...

- *Almost…*
  - *Quality Attributes needed*
    - *Security*
    - *Performance*
    - *Availability*
    - *Modifiability – the programming model is terrible lowlevel*
  - The reason we will do **Broker**
    - **Architectural Pattern to address the programming model…**