50005 - Networks and Communications - Lecture 9

Oliver Killane

21/04/22

Physical Layer

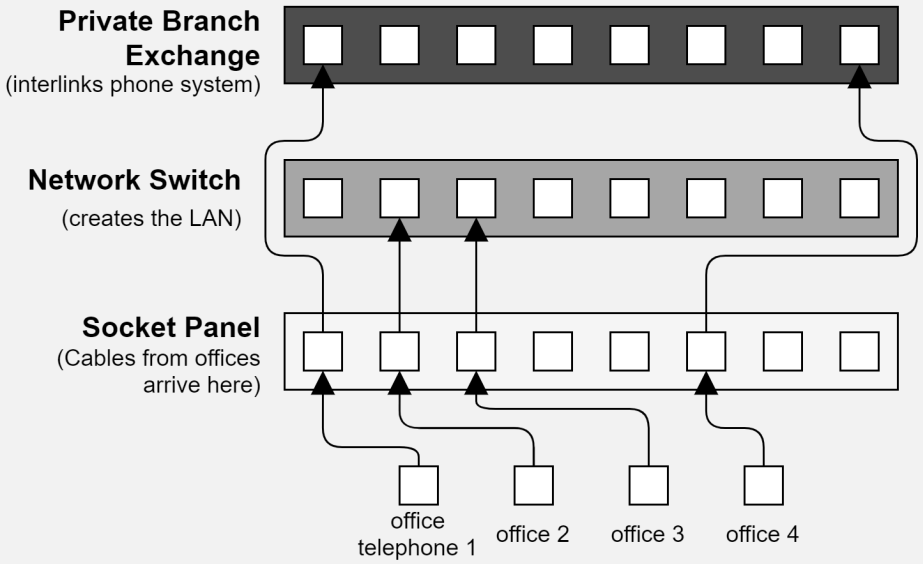
Lecture Recording

Lecture recording is available here

Network Architecture

A **network architect** designs the network (topology, standards, connections, where to put cables).
A **network engineer** installs the equipment to setup the network.

Definition: Patch Panel/Socket Panel



The **PBX** (**Private Branch Exchange**) is used for phones, however if the phone system is **IP** based, a separate **PBX** is not needed.

Wired Transmission

Definition: Unshielded Twisted Pair (UTP)

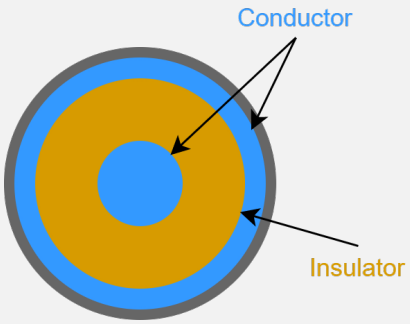
Two wires twisted together.

- Cheap & easy to mass-produce.
- Twisting reduces interference and crosstalk between cables.
- Used in telephone systems.

Type	Speed	Description
CAT1	$1 Mbps$	Voice grade for POTS (plain old telephone service).
CAT5	$100 Mbps$	10Base-T Ethernet Cables and 100Base-TX Fast Ethernet.
CAT6	$1,000 Mbps$	1000Base-T Gigabit Ethernet.

Definition: Coaxial Cable

Conductors placed concentrically (one inside the other), separated by an insulator.



- Good sheilding, electromagnetic field mainly between inner and outer conductor.
- Large bandwidth from high range of frequencies.
- Higher cost per meter (hence **UTP** is more popular for common consumption).

Definition: Optical Fibre

Transmits data using light and refraction (explained well here).

- Single optical fibre is 2 - 125 micrometers in diameter.
- **Attenuation** (signal loss) is low, so can be used for long distances.
- Very high bandwidth.

Year	Speed	Organisation
2011	26 *Tbps*	Karlsruhe Institute of Technology
2014	43 *Tbps*	Technical University of Denmark
2014	255 *Tbps*	Eindhoven University of Technology and University of Central Florida
2021	319 *Tbps*	Japan National Institute of Information & Communications Technology
2021	1000 *Tbps*	Japan National Institute of Information & Communications Technology

	Freq Range	Attenuation	Delay	Repeater Spacing
UTP	$0 - 1\ MHz$	$0.7\ dB/km$ @ $1\ KHz$	$5\ \mu s/km$	$2\ km$
Coaxial Cable	$0 - 500\ MHz$	$7\ dB/km$ @ $10\ MHz$	$4\ \mu s/km$	$1 - 9\ km$
Optical Fibre	$186 - 370\ THz$	$0.2 - 0.5\ dB/km$	$5\ \mu s/km$	$40\ km$

Wireless Transmission

Done using electromagnetic radiation (typically radio).

- No need for wires (expensive & take time to install).
- Bidirectional communication by default.
- Typically broadcast (e.g all/most recievers can see transmissions) (works with many stations).
- Inverse square law - signal strength reduces with range.
- Environment degrades signal (interference, obstruction, reflection of signal).

Wave Types

For more look at chapter 3 of Communication Systems.

Information Representation

Definition: Digital

Discrete information, represented by a finite number of states.

e.g 0 and 1 for binary.

Definition: Analogue

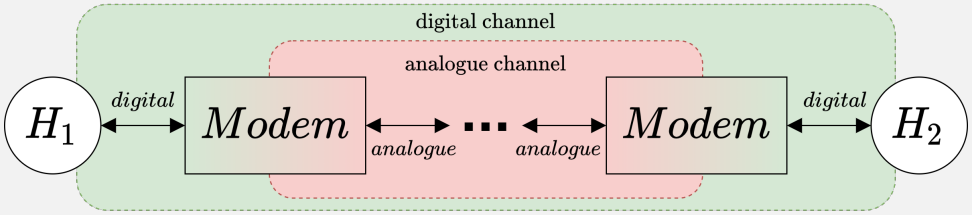
Continuous information, represented by changes in some physical state.

e.g light intensity, voltage.

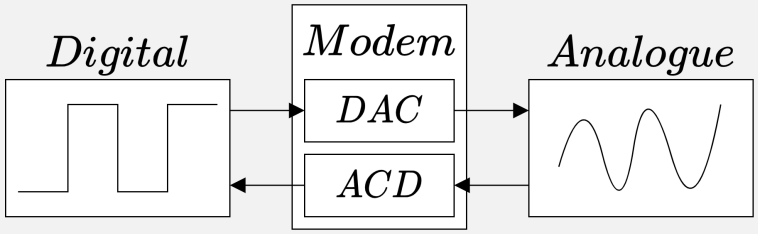
Definition: Baud Rate (Bd)

Symbol rate per second for a digital channel, where a symbol may represent more than 1 bit.

Definition: Modem



A **Modulator-Demodulator** implements a digital channel using an analogue channel.

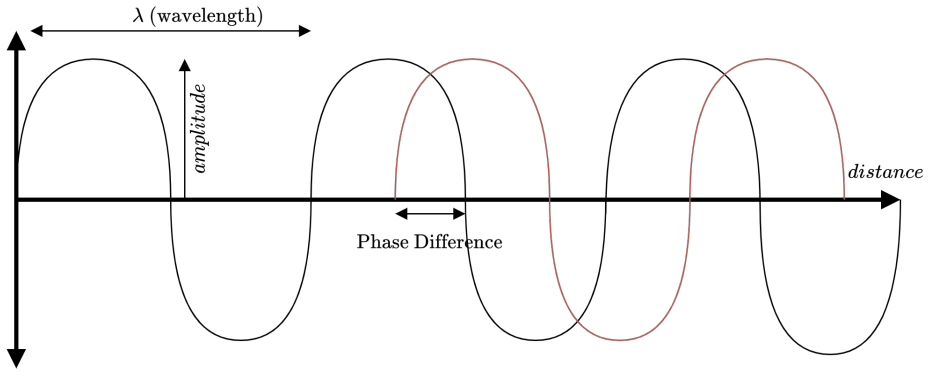


- **DAC → Digital to Analogue Converter**
- **ADC → Analogue to Digital Converter**

Definition: Codec

A **Coder-Decoder** implements an analogue channel using a digital channel.

Waves



Amplitude		Maximum displacement/strength of the signal.
Wavlength	λ	Length of a single cycle.
period	p	The time taken to complete a cycle.
Frequency	f	The number of cycles per second.

$$p = \frac{1}{f} \text{ (period and frequency)}$$

$$wavespeed = f\lambda \text{ (for radio waves} wavespeed = c = 3 \times 10^8 ms^{-1} \text{)}$$

Phase

Given two waves of the same wavelength and speed/frequency, they may be offset by some distance.

The pahse difference can be considered as a distance, or fraction of a cycle. In the latter anglke units may be used (full cycle $= 360\,\text{deg} = 2\pi$).

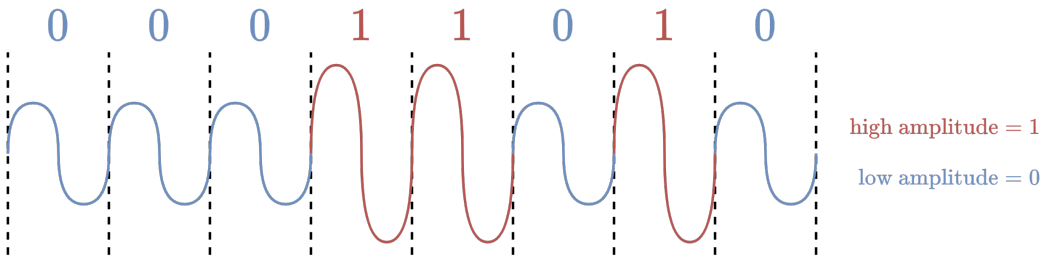
The maximum phase difference is π, where the waves are in opposite displacements for any given time during their cycle.

Modulation

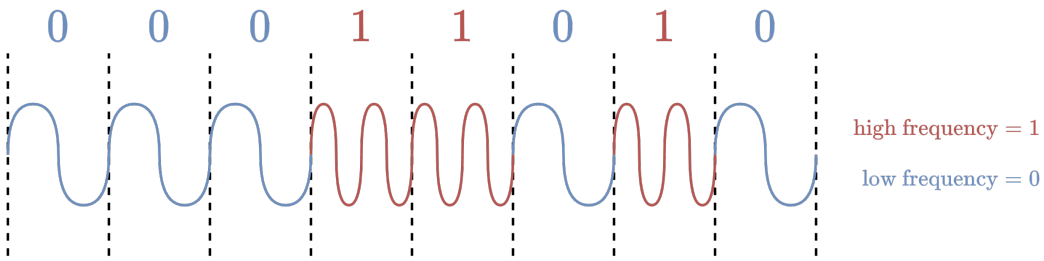
A **modulation** scheme is used to change some information signal into one more suitable for transmission.

Baseband Modulation	Transmit unmodified (dedicated line sending in full).
Broadband Modulation	Uses a basic carrier signal to encode information. The carrier signal has modifications added to encode information (e.g changing amplitude, frequency or phase).

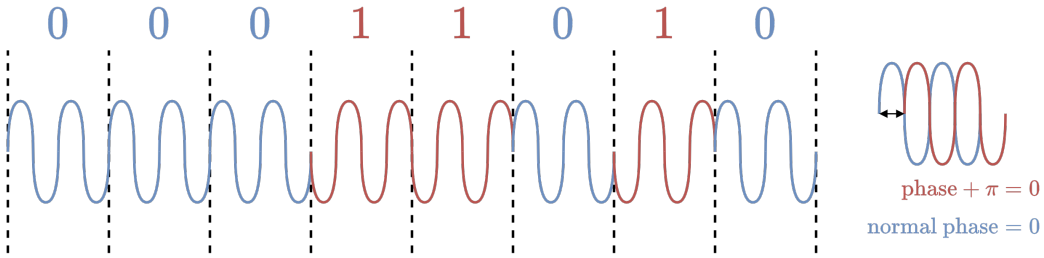
Amplitude Modulation / Amplitude Shift Keying (ASK)



Frequency Modulation / Frequency Shift Keying (FSK)



Phase Modulation

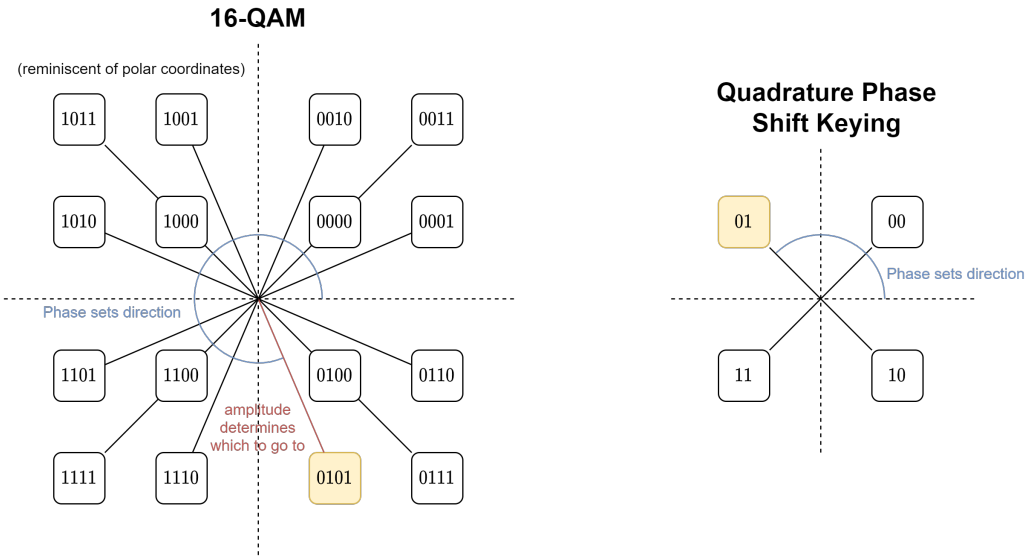


Better Modulation

To improve the data rate we can transmit multiple bits per symbol (in modulation scheme).

- Use more phase differences, amplitudes.
- Use a combination of phase, amplitude to determine symbol.

For example we can use phase (interpreted as an angle) in combination with amplitude in a scheme such as **QAM**.



Digital Subscriber Line (DSL)

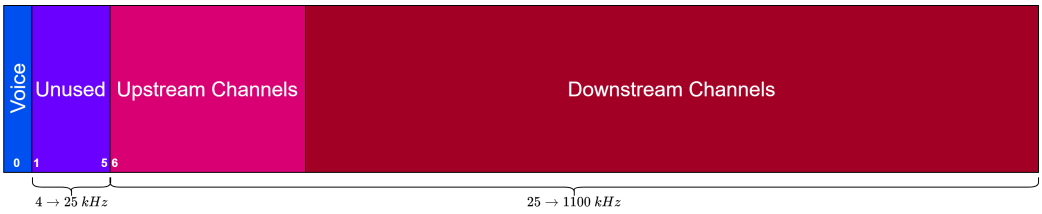
With the V.90 Modem Standard, using conventional phone lines to transfer data.

- Maximum $56,000$ *bps* downstream (download) and $33,000$ upstream (upload)
- Limited as phone lines limited to $3,000$ Hz bandwidth (human voice goes to $3,400$ Hz and was originally developed only for voice communication).
- Anything outside that range is filtered out as noise.

By removing the limitation (by removing the bandwidth filter) **DSL** allows for more bandwidth and hence a higher data rate.

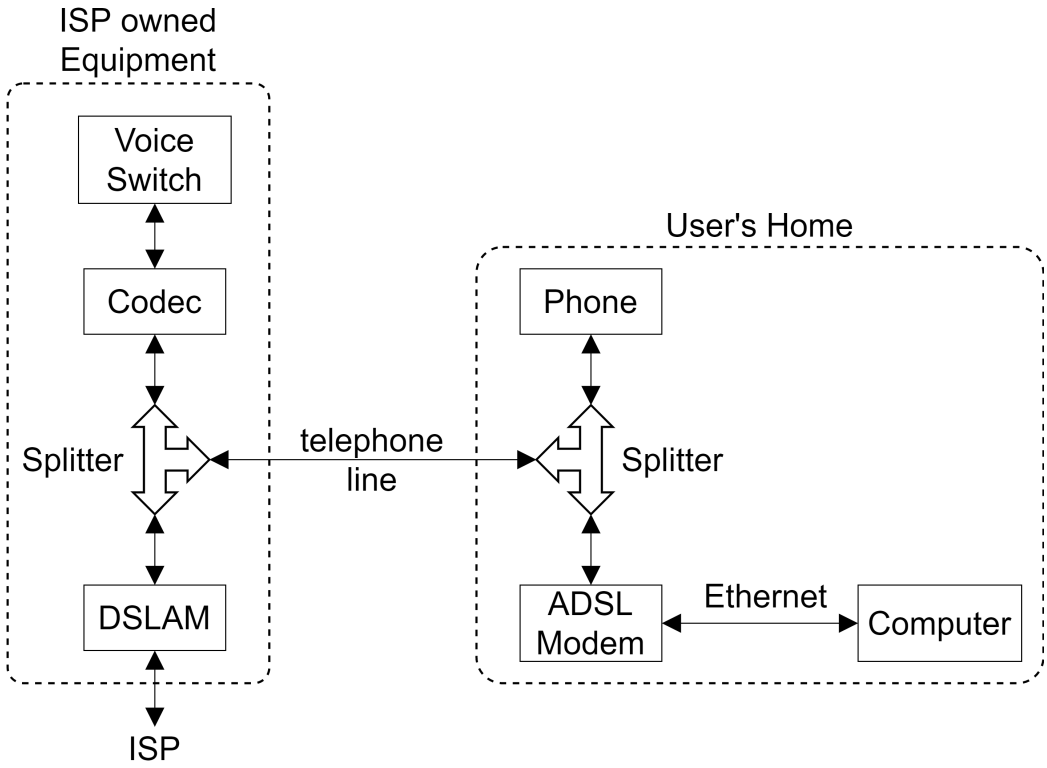
However noise now becomes a limiting factor.

Asymmetric Digital Subscriber Line (ADSL)



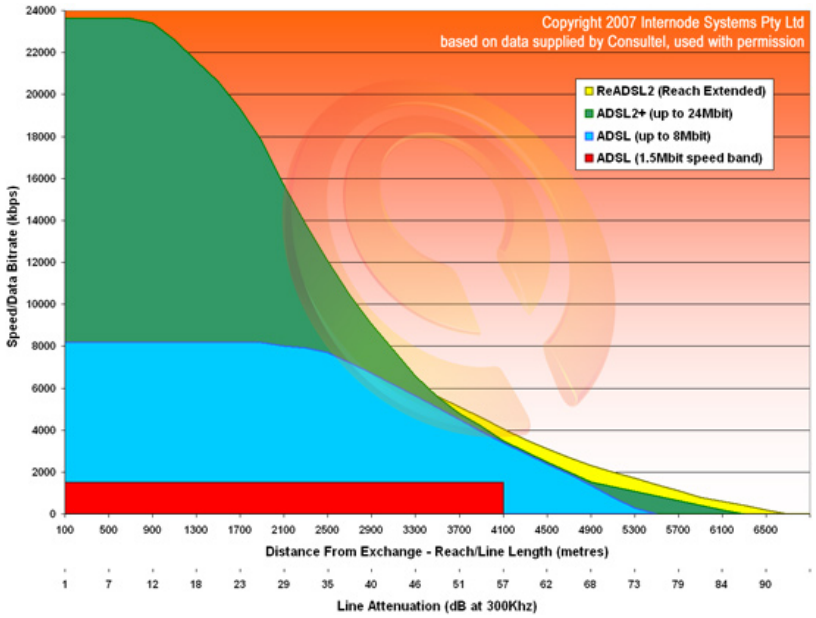
- 1.1 MHz of bandwidth divided into 256 4000 Hz channels.
- Channels $1 \rightarrow 5$ ($4 \rightarrow 25$ kHz) are unused to avoid interference between voice and data channels.
- More channels are allocated to download than upload as download used more heavily.
- Voice is cahnnel 0 ($0 \rightarrow 4$ kHz)
- V.24 modulation uses 224 downstream channels (13.44 $Mbps$)
- A **ADSL Splitter** separates the voice band from data.
- An **ADSL modem** performs modulation.

DSL Access Multiplexer (**DSLAM**) (typically owned by the **ISP**) connects local telephone cables to the **ISP**



DSL Advancement

ADSL2	12 $Mbps$	2.2 MHz	
ADSL2+	12 $Mbps$	2.2 MHz	More bits per symbol
VDSL	52 $Mbps$	12 MHz	Very-high-bit-rate DSL
VDSL2	200 $Mbps$	30 MHz	Currently popular



Network Simulation

Lecture Recording

Lecture recording is available here

Network simulation is used to design networks cheaply.

Different simulators provide different features:

- **Cisco Packet Tracer** Strong adacemic backing
- **gns3** Strong, open community backing
- **OPNET** Professional use, quite technical

Cisco packet Tracer allows code to be run inside the simulation:

- Cisco IOS commands (Cisco's proprietary Operating System)
- Terminal commands inside applications on Desktop/Laptops
- Web documents (through server's http service)
- Python & Javascript
- MQTT (Message Queue Telemetry Transport) (a lightweight machine-to-machine Publisher/-Subscriber messaging protocol)
-

Network Programming

Lecture Recording

Lecture recording is available here

Simple Echo

1. Run server, waiting for connections on a user-defined port.
2. Client connects to the port.
3. Server listens for input from the client.
4. User types into client, client sends message to server.
5. Server echos recieved data back to client.
6. Client disconnects.
7. Server closes.

```
1  import java.io.BufferedReader;
2  import java.io.IOException;
3  import java.io.InputStreamReader;
4  import java.io.PrintWriter;
5  import java.net.Socket;
6  import java.net.UnknownHostException;
7
8  class EchoClient {
9
10     static String hostName;
11     static int portNumber;
12
13     public static void main(String args[]) {
14         try (Socket echoSocket = new Socket(hostName, portNumber);
15
16             // Communication with the server through the socket
17             PrintWriter out = new PrintWriter(echoSocket.getOutputStream(), true);
18             BufferedReader in = new BufferedReader(new InputStreamReader(echoSocket.
                   ↪ getInputStream()));
19
20             // Read user input from terminal
21             BufferedReader stdIn = new BufferedReader(new InputStreamReader(System.in
                   ↪ ))) {
22
23
24             String userInput;
25                 while ((userInput = stdIn.readLine()) != null) {
26                 out.println(userInput);
27                 System.out.println("echo: " + in.readLine());
28             }
29
30         } catch (UnknownHostException e) {
31             System.err.println("Don't know about host " + hostName);
32             System.exit(1);
33
34         } catch (IOException e) {
35             System.err.println("Couldn't get I/O for the connection to " + hostName);
36             System.exit(1);
37
38         }
39     }
40 }
```

```
1  import java.io.BufferedReader;
2  import java.io.IOException;
3  import java.io.InputStreamReader;
4  import java.io.PrintWriter;
5  import java.net.ServerSocket;
```

```java
import java.net.Socket;

public class EchoServer {

    static int portNumber;
    public static void main(String args[]) {

        // try with resources statement closes the writer, reader and sockets
        // after the statement
        try (
            ServerSocket serverSocket = new ServerSocket(portNumber);

            // wait for the client to connect
            Socket clientSocket = serverSocket.accept();
            PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
            BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket
                ↪ .getInputStream()));
            ) {

            System.out.println("Client connected on port " + portNumber + ".
                ↪ Servicing requests.");
            String inputLine;
            while ((inputLine = in.readLine()) != null) {
                System.out.println("Received message: " + inputLine + " from " +
                    ↪ clientSocket.toString());
                out.println(inputLine);
            }
        } catch (IOException e) {
            System.out.println("Exception caught either when trying to listen on port
                ↪  " + portNumber + " or while listening for a connection");
        }
    }
}
```

Concurrent Executor

We can use a thread pool to handle running tasks for many clients connecting, and sending input.

```java
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class ConcurrentServer {
    static int threads = 5;
    static int portNumber;
    public static void main(String args[]) {
        try (ServerSocket serverSocket = new ServerSocket(portNumber);) {
            ExecutorService executor = Executors.newFixedThreadPool(threads);

            System.out.println("Waiting for clients to connect...");

            while (true) {
                Socket clientSocket = serverSocket.accept();
```

```
                executor.execute(new RequestHandler(clientSocket));
            }

        } catch (IOException e) {
            System.out.println("Exception caught when trying to listen on port " +
                ↪ portNumber + " or listening for a connection");
        }

    }
}

class RequestHandler implements Runnable {
    Socket clientsocket;

    RequestHandler(Socket clientsocket) {
        this.clientsocket = clientsocket;
    }

    @Override
    public void run() {

        try (
            BufferedReader in = new BufferedReader(new InputStreamReader(clientsocket
                ↪ .getInputStream()));
            BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(
                ↪ clientsocket.getOutputStream()));
            ) {
            System.out.println("Thread started with name:" + Thread.currentThread().
                ↪ getName());

            String userInput;
            while ((userInput = in.readLine()) != null) {
                userInput = userInput.replaceAll("[^A-Za-z0-9 ]", "");
                System.out.println("Received message from " + Thread.currentThread().
                    ↪ getName() + " : " + userInput);
                writer.write("You entered : " + userInput);
                writer.newLine();
                writer.flush();
            }

        } catch (IOException e) {
            System.out.println("Exception raised while attempting to handle request")
                ↪ ;

        }
    }
}
```

Oracle Guides

The guides these examples were based on can be found here.