Department of
Computer Science

UNIVERSITY OF
BATH

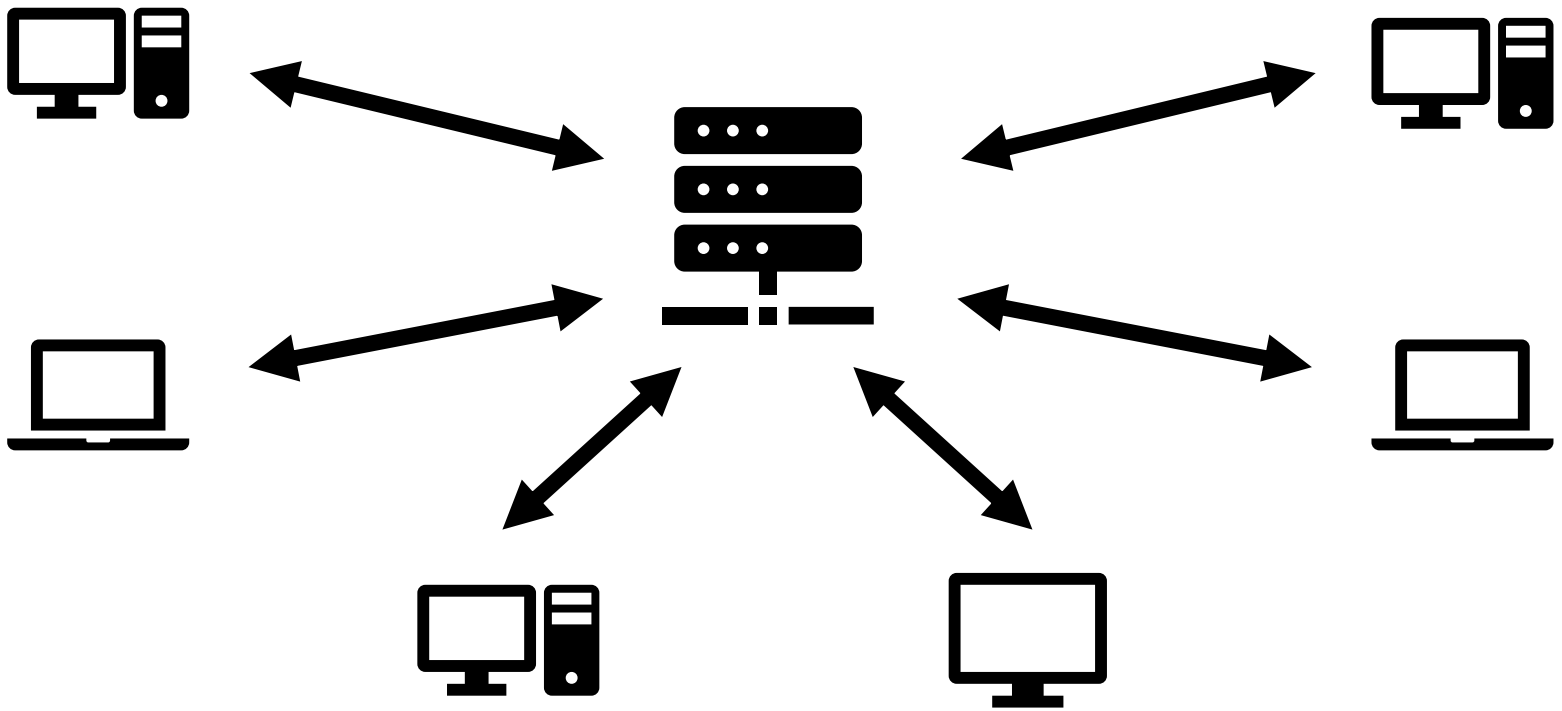
Week Eight Labs – Simple Chat System CM10228

Simple Chat System

- We will be creating our own simple chat system using the lessons we have learnt from the lectures
  - Multithreading
  - Networking
  - Graphical User Interfaces
  - + Encryption/Decryption

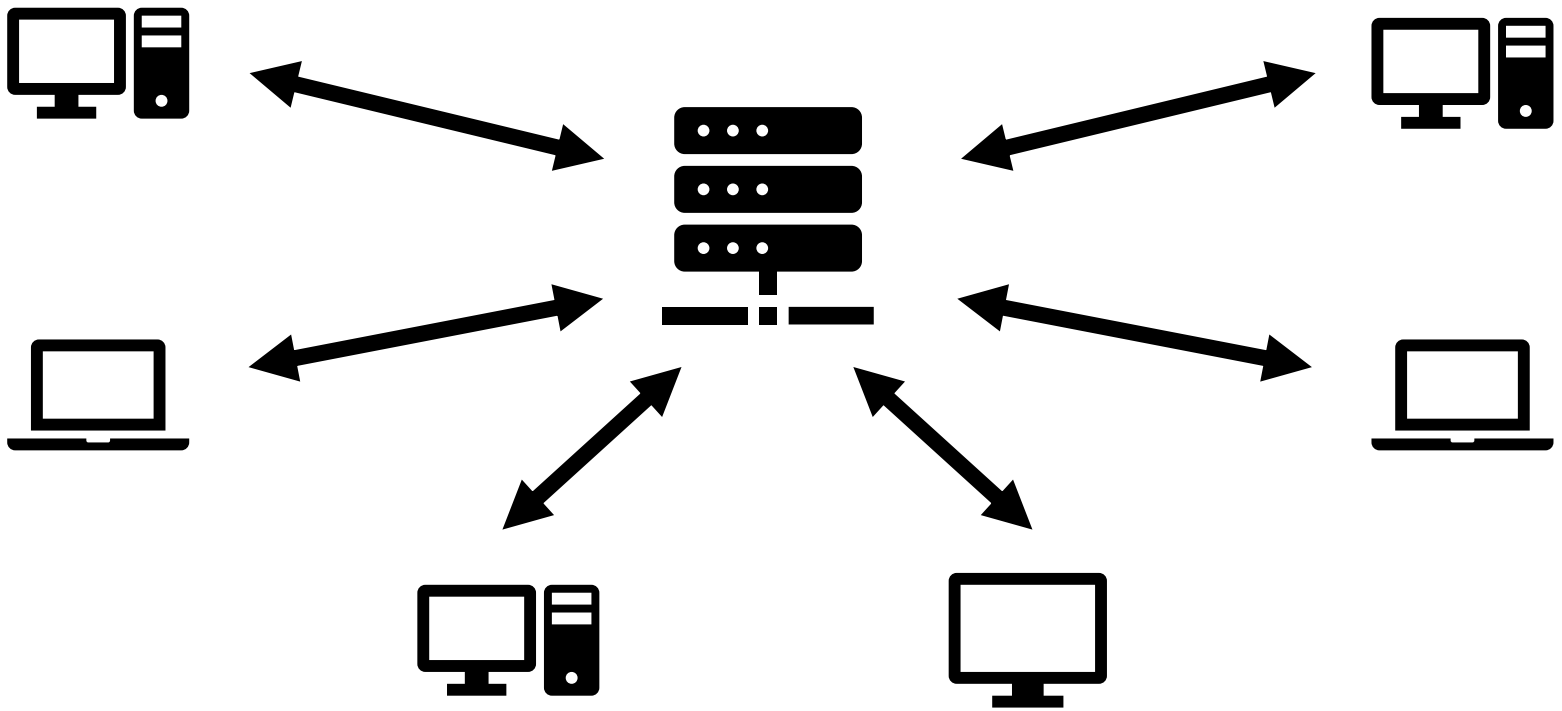
Client Server Design

- Our chat system will use a client server architecture



Client Server Design

- The clients will be able to communicate via the server



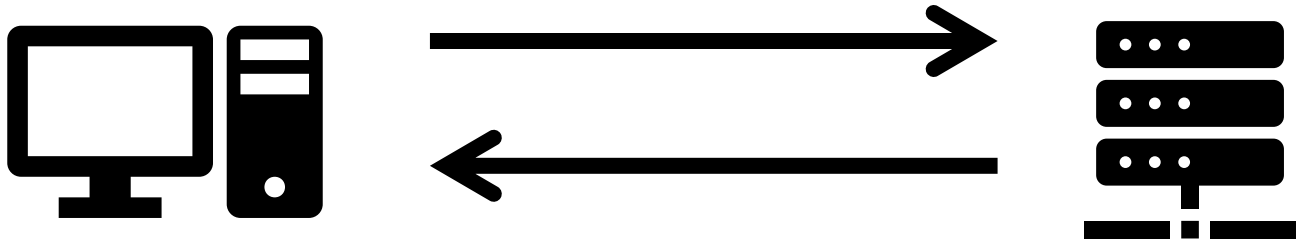
Iterative Approach

Until you end up with a program that meets your requirements:

1. Start with a program that works
2. Make small changes
3. Test after every change

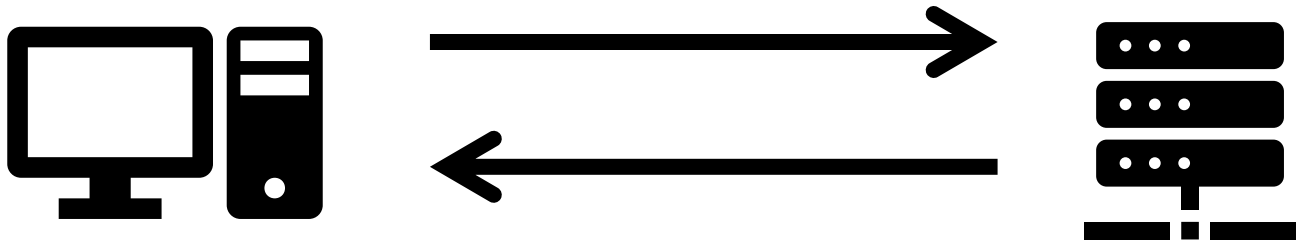
Echo Client Server

- In the lectures we saw a simple client server system
- A client can send a message to the server, the server will echo the message back



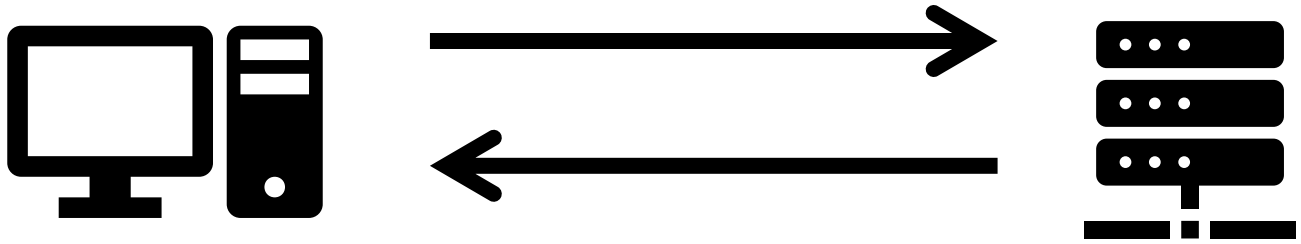
Echo Client Server

- Let’s start with that as a base and extend it…



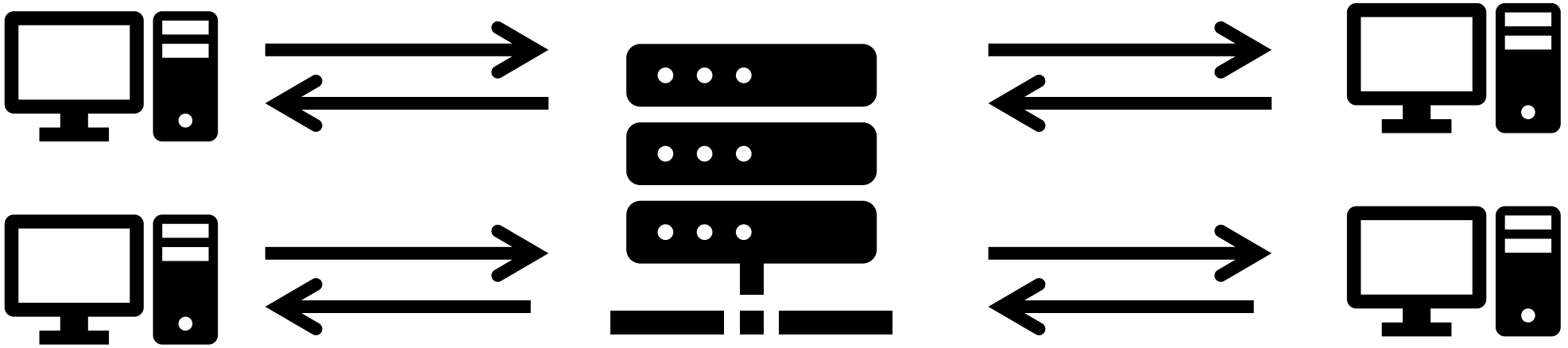
Server

- The server can only handle one connection at the moment…
- The accept() method blocks, meaning nothing can occur on the main thread…



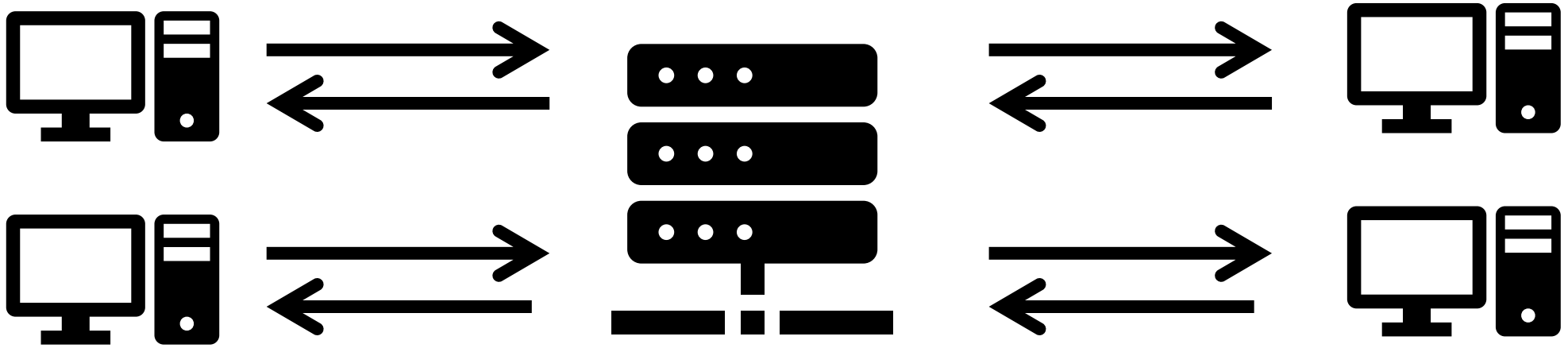
Server

- Let's use multithreading to enable multiple clients to connect to the server using the code we have…



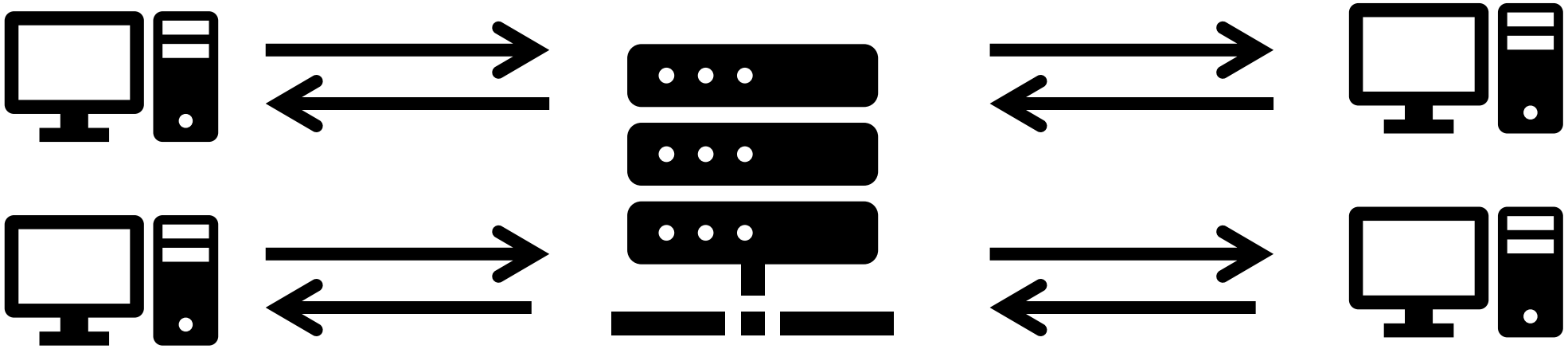
Server

- When we accept a client – let's spawn a thread that will be used to handle communication with that client, allowing the main thread to continue waiting for incoming connections (and spawn new threads when required)

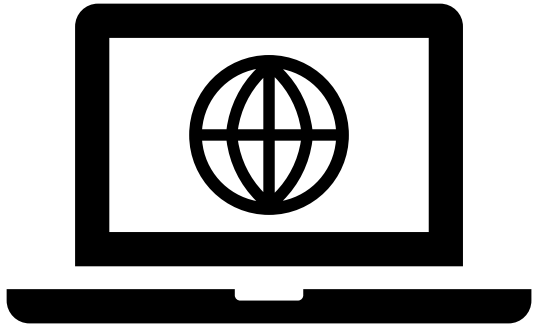


Server

- In the first instance, we can keep the client code the same to just echo the messages back…

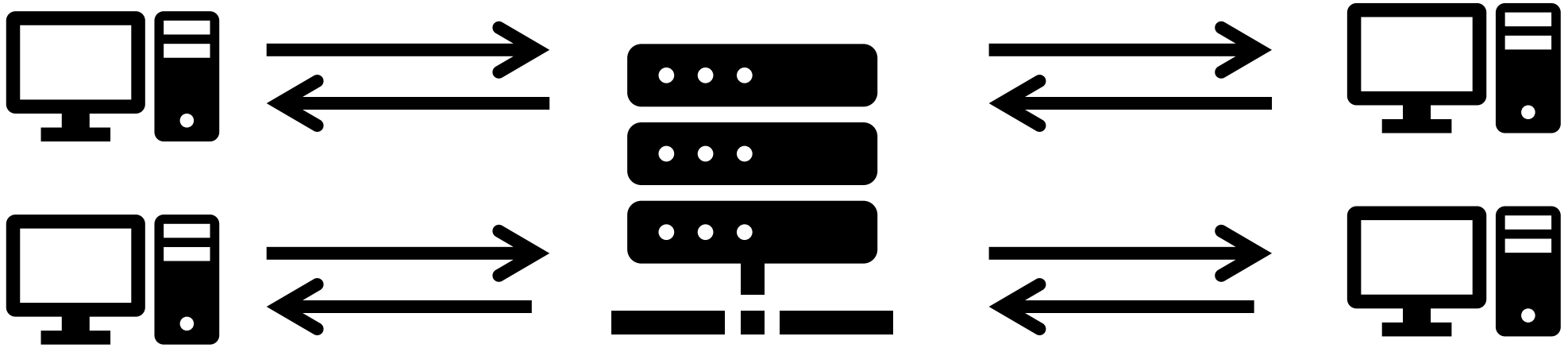


Let’s get coding!



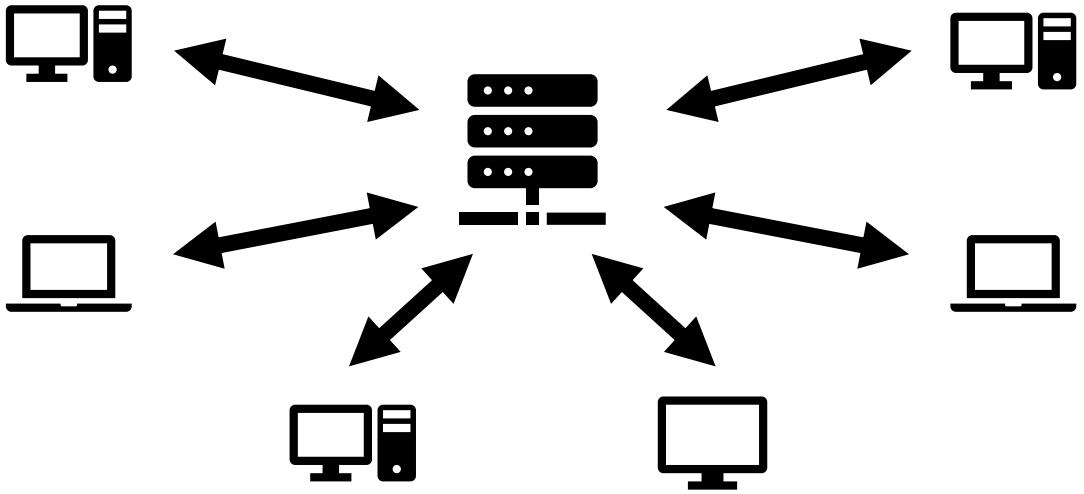
Server

- Now we have multiple clients connecting but they can only talk to the server in isolation
- Let's let the clients talk to each other…



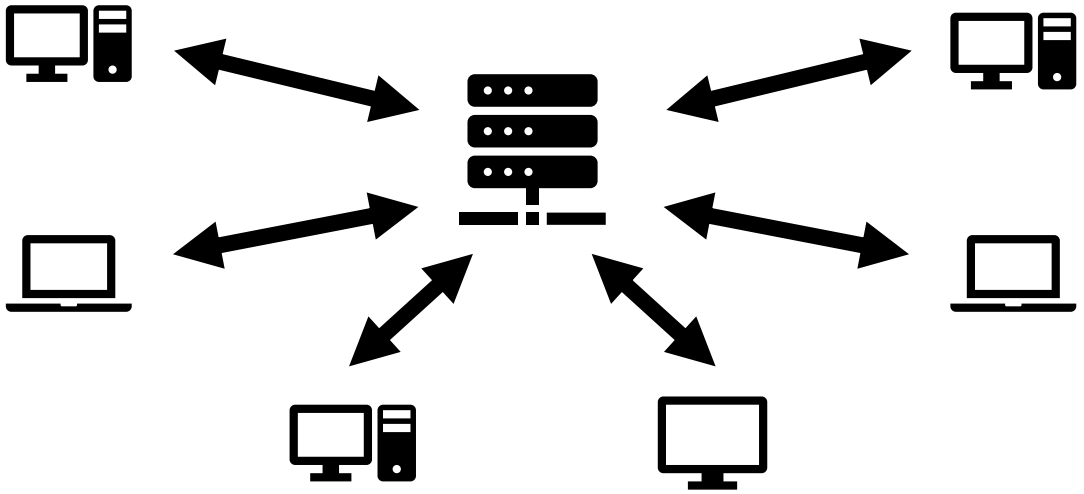
Server

- Let’s keep a track of the clients connected using a data structure, e.g., an ArrayList
- When a client connects, we should add it to our list

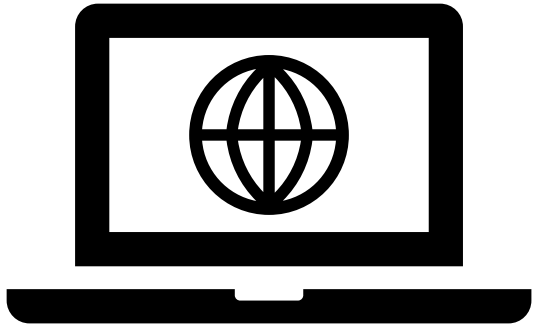


Server

- When we send a message from the server, instead of sending it to one client, we can send it to all using this list…
- … then the clients will be able to talk to each other

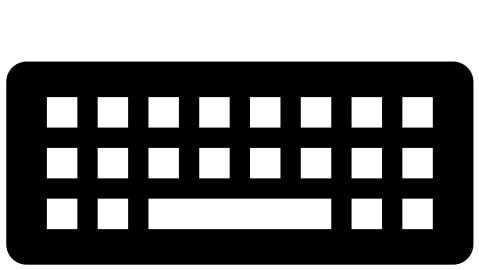


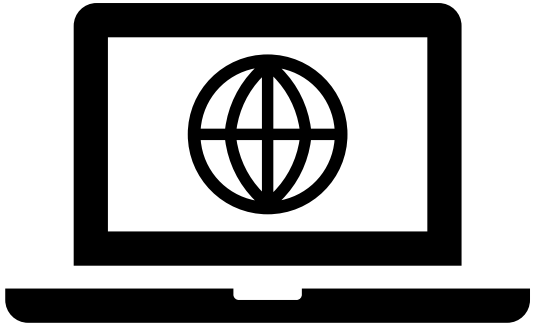
Let’s get coding!



Client

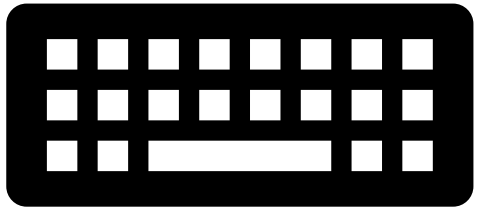
- Oh no!
- You may notice our clients will only receive the messages when (but only when) they send a message…

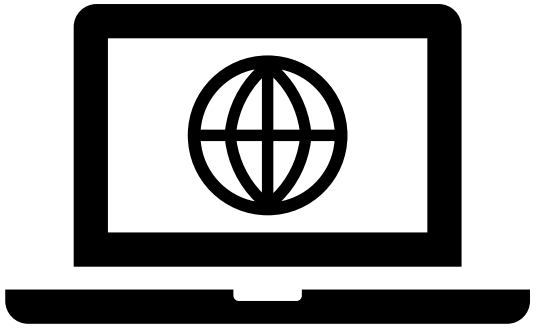




Client

- … this is because reading from System.in is also a blocking call!
- Therefore only when we send a message from the clients will we receive a message

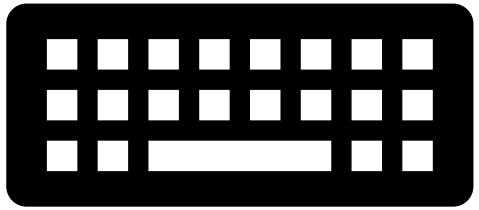


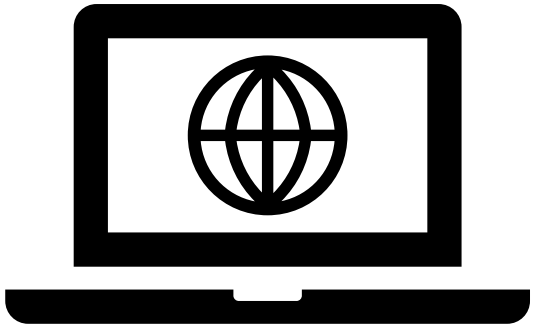


Client

We want the client to be able to simultaneously:

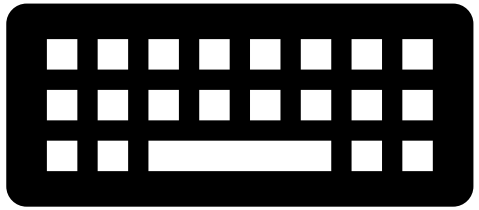
- Wait for input from the user in order to send a message
- Output all messages sent from the server as they come in real-time

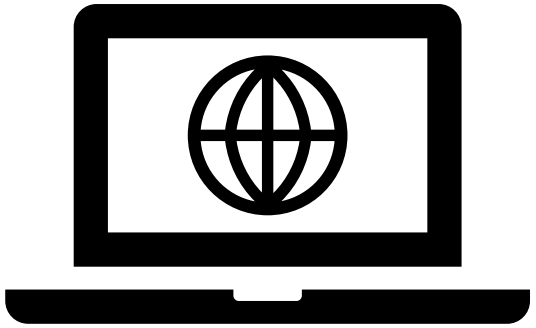




Client

- Sounds like another job for multithreading!
- When our clients start, we should have two threads:
  - One for handling user input and sending messages
  - One for handling incoming messages from the server and displaying them





Let’s get coding!