# Operating System

### Declan Kelly - G00378925

# Design

I decided to put all the Java into a single file,
making it easier to make edits to the Member and Club classes while in development.

The Member and Club classes store the properties of the type, example Club stores club id, and is abstracted
with getter and setter methods, when the record is read of the file system on startup, the line is split by
seperating space character and parsed into its properties,
each class has a toString method, which serialises the Object into a String, allowing it to be stored in the text f

When a connection is made to the server, a thread is spawned to deal with that connection,
allowing other connections to be made at the same time.

Added synchroization to saveToFile methods, to prevent two threads from writing to the file at the same time

# Communication

The client communicates with the Server using TCP, abstracted with Socket and SocketServer Java Classes
I made a custom protocol, that alerts the client when to (1) print text on client side, or (2) prompt for text or t

Example, three types of commands

| Server | Client | Notes |
|---|---|---|
| "1Enter club name: "=> | "Enter club name" input=>back to server | First character of string (1) alerts client input is needed and server is waiting on text to be sent back |
| "2This is a message" | "This is a message" | Just prints a message on client side Command 2 |
| "3" | Client exits | Server tells client to hangup client calls System.exit(0) Server kills running Thread |

# Persistent Storage

Clubs and Members are stored in two text files, these are read into a ArrayList on startup,
when a change is made to the ArrayList, the changes are sent to the files
each record is stored on a line.

Note: I choose text based storage because I was familar with the text parsing functionality of Java, rather
storing the raw bytes, which I could have done too.