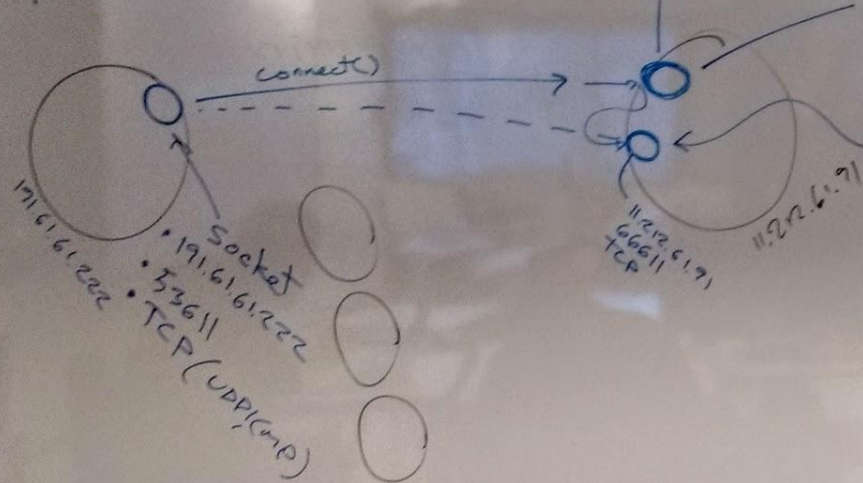


On sockets and threads

Sockets and getting connected

DNS
metropolis.fi → 11.212.61.91



```
ServerSocket ss = new ServerSocket(0, 5);  
loop  
{  
    Socket s;  
    s = ss.accept();  
    i = new Interpreter(s);  
    i.run();  
}
```

```
s.getInputStream()  
s.getOutputStream()  
s.getLocalPort()
```

Sockets and threads in simple chat server

Make your Interpreter Runnable - it already has run() method

Create a new constructor for Interpreter - to be able to pass the socket to Interpreter, you can get input and output streams from the socket object

Make up the *name* instance variable in Interpreter from the socket information (IP address and port, for example)

Observer pattern in simple chat server

Interpreter should implement HistoryObserver interface and register as observer when it starts and deregister when it is about to exit

History should implement methods to register() and deregister() as HistoryObserver

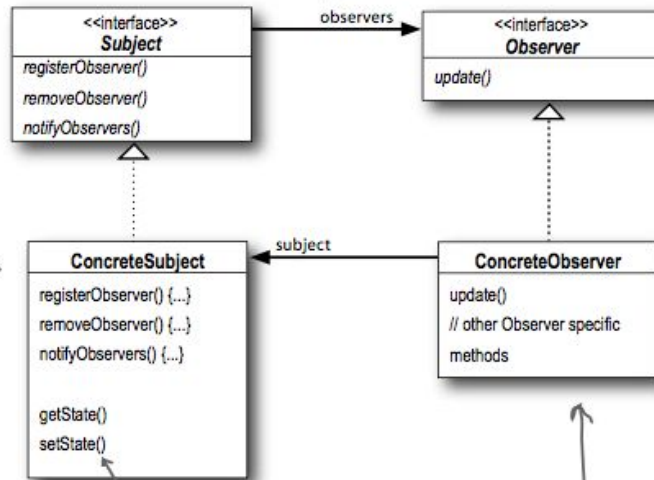
In History, make sure to call an update method defined in HistoryObserver interface to keep History observers updated

The Observer Pattern defined: the class diagram

Here's the Subject interface. Objects use this interface to register as observers and also to remove themselves from being observers.

Each subject can have many observers.

All potential observers need to implement the Observer interface. This interface just has one method, update(), that gets called when the Subject's state changes.



A concrete subject always implements the Subject interface. In addition to the register and remove methods, the concrete subject implements a `notifyObservers()` method that is used to update all the current observers whenever state changes.

The concrete subject may also have methods for setting and getting its state (more about this later).

Concrete observers can be any class that implements the Observer interface. Each observer registers with a concrete subject to receive updates.

Head First Design Patterns,
pages 37-78

Reading list

On sockets:

<https://docs.oracle.com/javase/tutorial/networking/sockets/index.html>

On threads:

<https://docs.oracle.com/javase/tutorial/essential/concurrency/procthread.html>

<https://docs.oracle.com/javase/tutorial/essential/concurrency/threads.html>

On observer patterns:

Head First Design Patterns, pages 37-78