```java
package weightsimulator.socket;

import java.io.BufferedReader;

public class SocketController implements ISocketController {
    Set<ISocketObserver> observers = new HashSet<ISocketObserver>();
    //TODO Maybe add some way to keep track of multiple connections?
    private BufferedReader inStream;
    private DataOutputStream outStream;


    @Override
    public void registerObserver(ISocketObserver observer) {
        observers.add(observer);
    }

    @Override
    public void unRegisterObserver(ISocketObserver observer) {
        observers.remove(observer);
    }

    @Override
    public void sendMessage(SocketOutMessage message) {
        if (outStream!=null){
            //TODO send something over the socket!
            PrintWriter out = new PrintWriter(outStream);
            out.println(message.getMessage());
            out.flush();
        } else {
            System.err.println("Connection closed"); //Tells the user that to connection with the socket could not be made
        }
    }

    @Override
    public void run() {
        //TODO some logic for listening to a socket //(Using try with resources for auto-close of socket)
        try (ServerSocket listeningSocket = new ServerSocket(Port)){
            while (true){
                waitForConnections(listeningSocket);
            }
        } catch (IOException e1) {
            notifyObservers(new SocketInMessage(SocketMessageType.P111, "Something went wrong"));
            e1.printStackTrace();
        }


    }

    private void waitForConnections(ServerSocket listeningSocket) {
        try {
            Socket activeSocket = listeningSocket.accept(); //Blocking call
            inStream = new BufferedReader(new InputStreamReader(activeSocket.getInputStream()));
            outStream = new DataOutputStream(activeSocket.getOutputStream());
            String inLine;
            //.readLine is a blocking call
            //TODO How do you handle simultaneous input and output on socket?
            //TODO this only allows for one open connection - how would you handle multiple connections?
```

```
68              while (true){
69                  inLine = inStream.readLine();
70                  System.out.println(inLine);
71                  if (inLine==null) break;
72                  switch (inLine.split(" ")[0]) {
73                  case "RM208": // Display a message in the secondary display and wait for
    response
74                      //Depending on the number after RM20 4 or 8, notify with either RM204
    or RM208
75                      notifyObservers(new SocketInMessage(SocketMessageType.RM208, ""));
76                      break;
77                  case "D":// Display a message in the primary display
78                      notifyObservers(new SocketInMessage(SocketMessageType.D, inLine.split("
    ")[1]));
79                      break;
80                  case "DW": //Clear primary display
81                      notifyObservers(new SocketInMessage(SocketMessageType.DW, ""));
82                      break;
83                  case "P111": //Show something in secondary display
84                      notifyObservers(new SocketInMessage(SocketMessageType.P111,
    inLine.split(" ")[1]));
85                      break;
86                  case "T": // Tare the weight
87                      notifyObservers(new SocketInMessage(SocketMessageType.T, ""));
88                      break;
89                  case "S": // Request the current load
90                      notifyObservers(new SocketInMessage(SocketMessageType.S, ""));
91                      break;
92                  case "K":
93                      if (inLine.split(" ").length>1){
94                          notifyObservers(new SocketInMessage(SocketMessageType.K,
    inLine.split(" ")[1]));
95                      }
96                      break;
97                  case "B": // Set the load
98                      notifyObservers(new SocketInMessage(SocketMessageType.B, inLine.split("
    ")[1]));
99                      break;
100                 case "Q": // Quit
101                     notifyObservers(new SocketInMessage(SocketMessageType.Q, ""));
102                     break;
103                 default: //Something went wrong?
104                     System.err.println("Command not found");
105                     break;
106                 }
107             }
108         } catch (IOException e) {
109             notifyObservers(new SocketInMessage(SocketMessageType.P111, "Something went
    wrong"));
110             e.printStackTrace();
111         }
112     }
113
114     private void notifyObservers(SocketInMessage message) {
115         for (ISocketObserver socketObserver : observers) {
116             socketObserver.notify(message);
117         }
118     }
119
120 }
```