COMP 2005 Iteration 4: Accounting for Networking

To account for networking the most important thing we need to do is set up a client-server relationship with four new classes: ServerMain, ServerListener and ClientMain and ClientListener.

First the ServerMain class will have to have an object attribute for the game and a ServerSocket with the desired port it wishes to run on. You then call the accept() method on the ServerSocket object and wait for a connection. Once a connection is established you have to create an ObjectOutputStream and ObjectInputStream from the socket that connected. It's at this point you create a new ServerListener object, pass it the input and output streams, game object and create a new Thread object and start it on the ServerListener. This process has to be completed for each player you wish to connect and all the objects except the ServerSocket have to be recreated, IE. if you wanted to connect four players you would need to do it four times.

All the ClientMain class does is create a Socket object with the host IP parameter being the IP of the computer running the server and the port parameter being the port that the server is opening connections on. It then creates Object input and output streams from this socket and creates a new ClientListener Object and starts a Thread on it.

The ServerListeners job can be described by this process

- Listen for input from the client
- Checks if input is valid
- Updates the server's version of the game
- Sends the input to the client
- Clients game updates
- Checks for disconnects

The Client Listeners job can be described by this process

- Listen for inputs
- Check if valid
- Update the game based on the inputs

We would also have to change our main game controller – "ThirdPanel" as well, we would have to make a second constructor for it that asks for an ObjectOutputStream object. This output stream would be passed from the ClientMain Class so that the Client game object can write to the server. We would use this output stream to write to the server whenever the user clicked or typed on the board.

Lastly in order to be able to encapsulate all the input into one object we would have to make a class that represents the user's input. Its attributes would be moves the user can do in the game, for instance it would have a "Bet" Integer attribute that represented a player's move guess count. This would be what all the output and input streams send to and from each other to make the process more automated rather than having to hard code the whole game process.