CM10228 Coursework 3 Write Up

Classes

myServer – This is the main server class, which should be called when first launching the server. It firstly creates a frame to get the port number and an optional map, both of which are used to launch the main server GUI. This shows a bird-eye-view of the whole map, with labels for the IP address and port number the server is using, as well as an additional frame to turn on and off listening for new clients to join the game.

clientThread – called when a client successfully connects to the server. This thread listens to the client and sends the message to the playGame class, with the return message being sent back to the client using input/output streams. Every time a new client connects, their clientThread is added to a list of clients so the DODGame class can interact with all of the players when needed.

myClient – This is the client class which should be called when first launching a client. It firstly creates a frame to get the IP address & port number of the server the user wishes to connect to, as well as the screen name they wish to be seen as when posting messages. These are then used to connect to the server, and then the clientGUI is called to display the main client GUI.

clientGUI – This class shows the main GUI for the client, displaying the client's look messages in an icon grid, an arrow pad for movement, buttons for game actions and a notification/chat area. The 'x' button is disabled so the user must use the quit button supplied to exit the game, for a clean disconnection. Due to look calls when a user carries out an action, each player's map will update when this happens, as well as the server's GUI.

listenToServer – This class listens to the servers output, and deals with its reply messages. If the reply is a lookreply; it will reset the player's map with this reply, else if it is a game won/lost message, it will display a message to each player & the server and they will all exit, else the reply is added to the notifications box. Shuts the client if the server disconnects.

mapGUI – This class is the class that deals with creating and printing out the map, using a ImageIcon 2d array to store the icons in, and then paint each item in the array to the panel. It initialises the grid using the empty icon (plain black) and then can re-paint the map using the setMap method when it is called.

PlayGame – This is the class that connects the server to the DODGame class, checking for a valid player command, before passing it into DODGame. This is the class that now creates the player rather than DODGame, so the player is passed from the PlayGame class into the DODGame class.

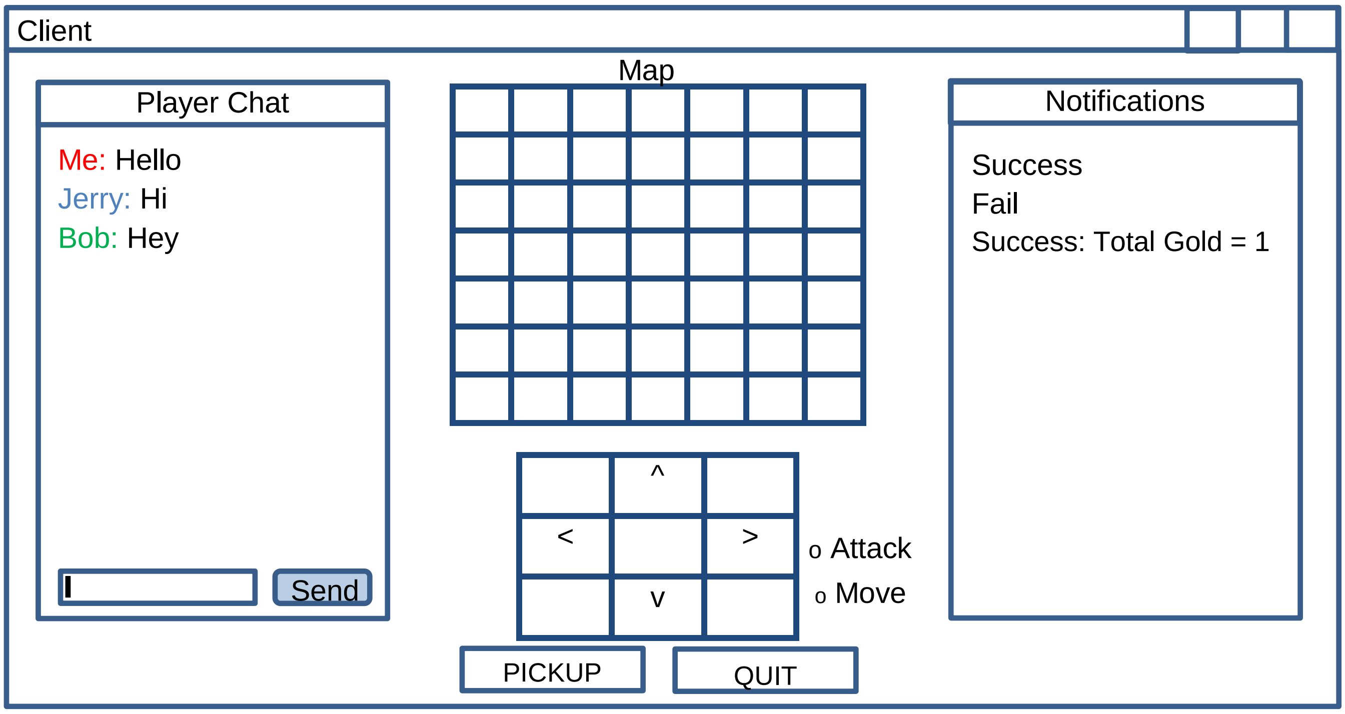
DODGame – carries out all of the main in game commands, most of which were supplied in the code already. A few extra's include dealing with a client quitting the game, a lookreply being sent to all clients, and returning the look for a whole map for the servers use. Action points (AP) are no longer in use.

DODMap – creates a default map/loads a map from a file. Basically unchanged from supplied code.

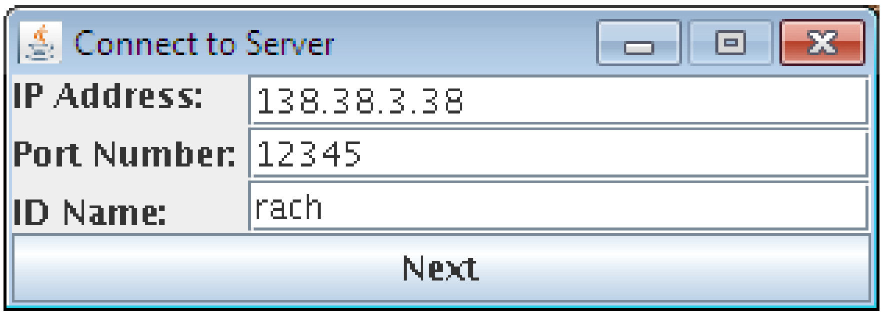
Player – creates a player and sets its starting values. Basically unchanged from supplied code.

Initial Drawings

Below are some initial sketch ups of my server and client GUI's, compared to my final outcome.



Above was the predicted view of a client's GUI they would see whilst playing the game. There would be a player chat facility to the left, a notifications area on the right, and the GUI map would have been placed in the centre of the frame. The user would have N/S/W/E pad for attack and move which they could switch between, and a pickup/quite facility. I have managed to produce a fairly similar final GUI, which I am pleased with.

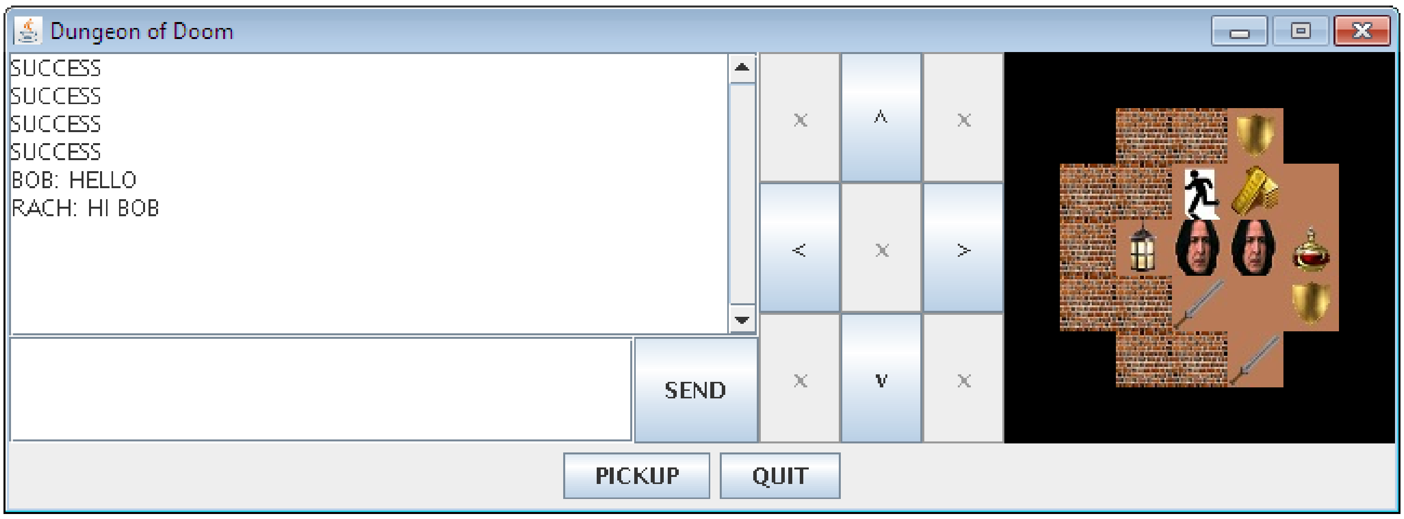


Connect to Server

IP Address: 138.38.3.38

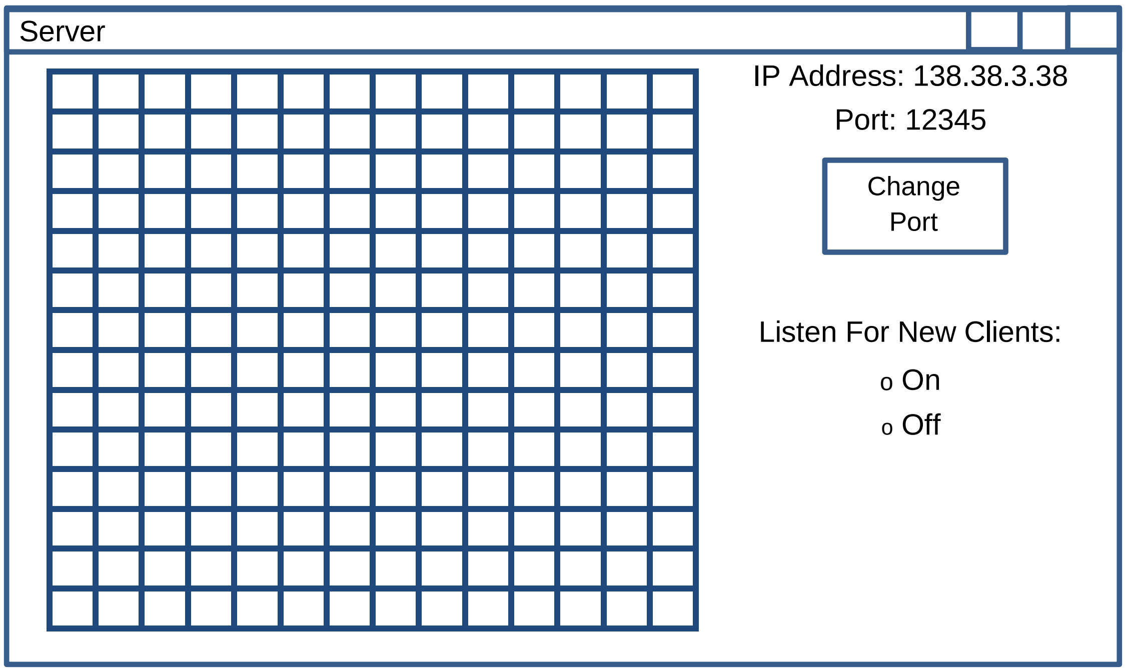
Port Number: 12345

ID Name: rach

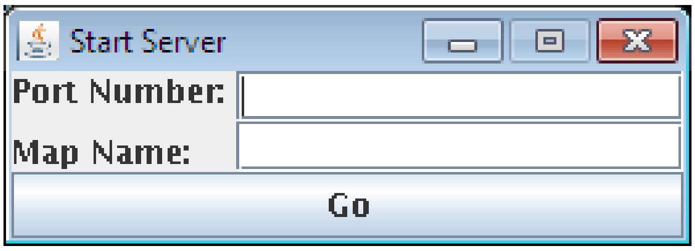


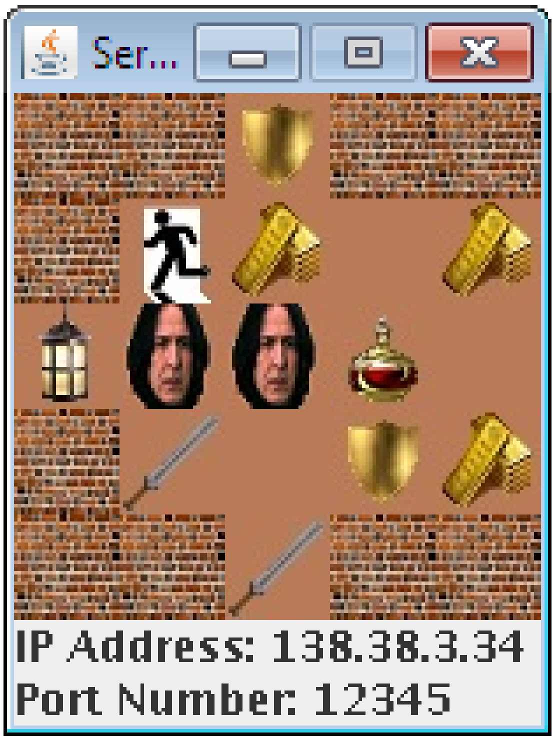
Dungeon of Doom

```
SUCCESS
SUCCESS
SUCCESS
SUCCESS
BOB: HELLO
RACH: HI BOB
```



Above was the predicted view of a server's main GUI, with a large birds-eye-view of the whole map to the left, labels showing the IP address & port number on the right, a change port button and a listen for new client connections toggle on and off. My final product does not have the option to change the port number, but the user is able to enter a port number when the server is first launched, along with a map file. The listening for new client connections is also an option, but it is in a separate Frame.

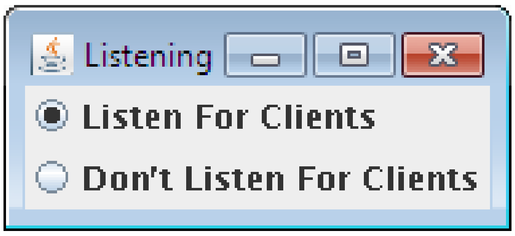




Ser...

IP Address: 138.38.3.34

Port Number: 12345



● Listen For Clients

○ Don't Listen For Clients

Intended Features	**points**	**Working Status**	**Works & marks**	**Special Instructions**
Basic Client GUI	15	Yes		none
Client Graphic Pane	10	Yes		none
Client Graphic Pane Updates	10	Yes		none
Server GUI	10	Yes		none
Total Points:	45			

Running the Program

- Click Start, & Select XMING - LCPU from All Programs
- Enter Username & Password into LCPU
- Enter command "cd filepath" – where filepath is path to file containing program .java files
- First run the server; enter command "javac *.java" to compile the code, then "java myServer"
- Then run the client: open a new LCPU, navigate to the same filepath, enter command "java myClient"

Using the Program

Server

- Entering the port number & map – You should enter a port number and map filepath in the two respective text boxes when the server starts up. If an incorrect port is used, an error will pop up, and if an incorrect map filepath is entered, the default map is opened in the server.
- Turning listening on and off – To turn listening for new clients on and off, simply toggle between the two 'On' & 'Off' radio buttons on the separate listening window will turn this on and off.
- Quitting – Simply hit the 'x' button to close the server – this will notify all clients of the server's disconnection and will in turn close them too.

Client

- Moving- To move simply press one of the 4 arrow buttons on the display to move north, east, south and west respectively. The map will update if a change is made, and the success/fail notification will display in the left hand side text area.
- Pickup- Simply press the button labelled 'PICKUP' to pick an item up – will send a fail notification if there is nothing to pick up, or will display the new amount of treasure in the notification box.
- Chatting – To chat with any other clients, simply type a message in the chat bar and hit the send button. Your name and message will be displayed to all other clients.
- Quit – To quit you must use the button labelled "QUIT" – the 'x' button will not work - and the server will delete this client from the game. Other clients can still play as normal.
- Winning – If you stand on an exit and have the correct number of golds, this client will be notified they have won. Others will be notified of their failure, and all clients and the server will then exit.

Critical Analysis

Overall, I thought my program was fairly good, and I am pleased with the final product. I like my GUI, and that it will update when any player makes a move or action. I also like the added bonus of the chat feature, even though it will not be marked, it was an interesting feature to program in to my code. I also managed to get the networking code to work which I struggled with in coursework 2, which I am now happy that I would be able to program without help. Although I am pleased with the outcome of my program, if I had more time I would have added in some other features such as attacking, or adding a bot into the program too. I would have also liked to program some openGL graphics to animate the map, but I did not quite grasp it when I looked into it.