CS3524 Assessment 2 Multiplayer Game Server

Name: Hasan Ahmedov

ID: 51768277

1. Description

In this assessment, we had to implement a simple game server that allows multiple players (clients) access to a MUD (Multi-User Dungeon or Dimension or Domain). It examined our ability to design and implement distributed systems using Java RMI.

Initially we were provided with a "**MUD**" class, which supplies a graph-based representation of a MUD world. In order to represent this game graph in memory, the "**MUD**" class uses two other classes – "**Vertex**" and "**Edge**":

- The "**Vertex**" class represents a vertex in the graph
    - These are the "locations" in the MUD game
- The "**Edge**", class represents an edge in the graph
    - These are the paths between locations in the MUD game.

Class "**MUD**" contains the following public methods: **addThing()**, **delThing()** **moveThing()**, **locationInfo()**. A "**thing**" in terms of this MUD implementation is everything that is situated at a location – this includes the players that move through the game, or items that can be picked up at a location.

A MUD game server and a client were implemented, based on Java RMI:

- MUD game server application classes:
    - "**MUDServerInterface**" - a remote interface that extends java.rmi.Remote and declares a set of remote methods

    - "**MUDServerImpl**" – a class that implements the interface remote methods
    - "**MUDServerMainline**" – server mainline that registers remote objects with the **RMI registry**

- MUD client application classes:
    - "**MUDClientInterface**" – the client interface which declares a set of methods to utilise the remote ones provided by the remote object.
    - "**MUDClientImpl**" – implements the client interface class methods which invoke the remote object
    - "**MUDClientMainline**" – fetches the remote object and calls the client methods which invoke the remote object

The MUD game server effectively grants multiple clients access to its remote methods and a view of each other's actions.

A user can create / switch to / join another MUD and continue playing in the new MUD, effectively playing multiple MUDs, and being able to switch the users "game focus" between multiple games. The server restricts the number of MUDs (3 MUDs) that can run at any time and the total number of users (2 players) logged on to MUDs.

The additional features of the multiplayer game server comprise of:

- **Responsiveness** – automatic refresh of player consoles:
    - A call back implementation was used to allow the server to push changes in a MUD game to all the clients that are currently playing the game (player arriving at a location, a player collecting items etc.)
- **Robustness** – the server implementation is robust against clients leaving a game or client applications being aborted (a shutdown hook was used):
    - The servers recognizes such a situation, cleans up client logins and unregisters any remote objects in the **RMI registry** if necessary

- **Communication**:
    - The server allows players to send personal messages to each other

When a client connects to the server, they are prompted for a username followed by their MUD choice from the currently running MUDs (Default (2x2) and Expanded (3x3) MUDs) menu. The client is also allowed to create their own default (2x2) MUD at runtime by selecting "Create your own MUD" option in the menu.

After joining a MUD the start location info of the player is printed out to them. Consequently, the player enters the game loop where they are asked to type in a command from the following commands list:

- "**help**" - prints out information about the available commands and how to use them.
- "**move**" - move the player in the MUD
- "**take**" - pick an item at the current location (if any)
- "**drop**" - drop an item at the current location
- "**inv**" - print out items in the player's inventory
- "**players**" - print out current online players in the MUD
- "**msg**" - send a personal message to a player in the MUD
- "**join**" - join another MUD
- "**leave**" – leave the current MUD
- "**switch**" - switch to another MUD (if previously joined)
- "**quit**" - leave all MUDs and quit server
- "**Enter**" ("Return") key for current MUD and location details (acting as a refresh of the console output)

2. How to run

1) To enable you to quickly run the multiplayer game server application, a make file is provided that compiles all the required files. To run the make file you must launch a terminal window and navigate to the appropriate directory where the make file is located (where **XXXX** is cs3524_assessment_2_u01hba17/CS3524 in our case):

    - **your-linux% cd XXXX**

2) Once you are in the appropriate directory just run the following command in the console to compile the files (**make cleanmud** to clean the folder):

    - **make mud**

3) Start the **RMI registry** in one command prompt by providing a port number (50000 and above) at the command line:

    - **rmiregistry 50010**

4) Start the server in another command prompt:

    - **java cs3524.solutions.mud.MUDServerMainline 50010 50011**

5) Start the client application(s) again in separate command prompt(s):

    - **java cs3524.solutions.mud.MUDClientMainline localhost 50010 50012**

    - **java cs3524.solutions.mud.MUDClientMainline localhost 50010 50013**

6) Enjoy!

3. Sample output

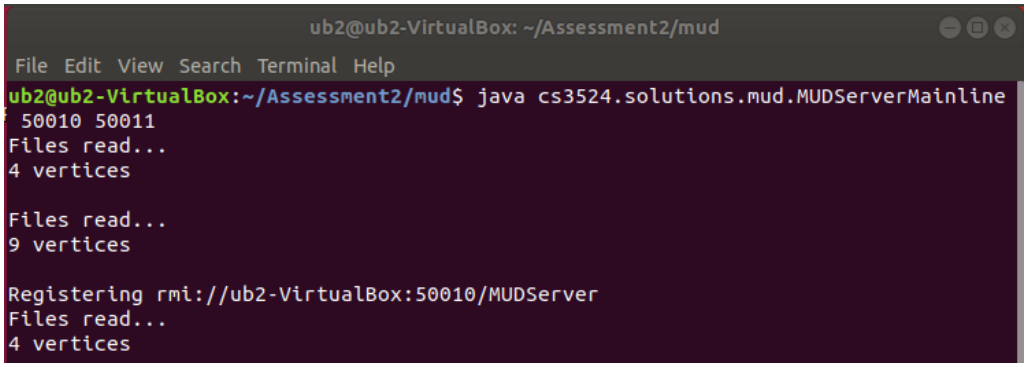

Figure 1. *Running the multiplayer game server mainline.*

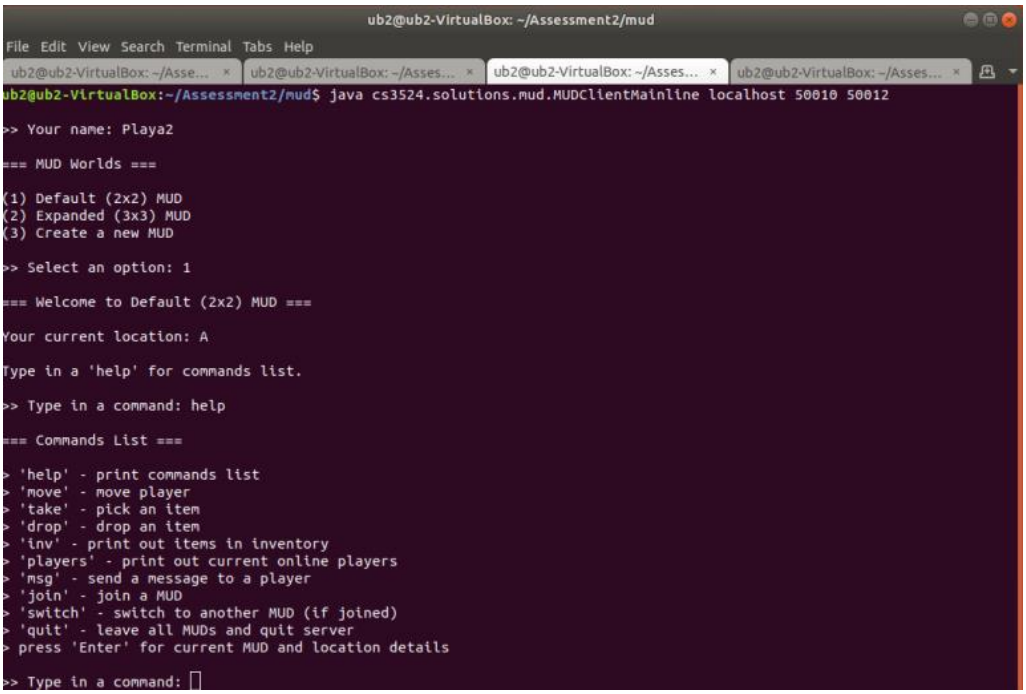

Figure 2. *Connecting a client to the multiplayer game server, entering a MUD and printing the commands list.*

Figure 3. *Two player interaction demo – receiving call back messages from the server for player arrival, item being picked up and movement, and sending a personal message.*

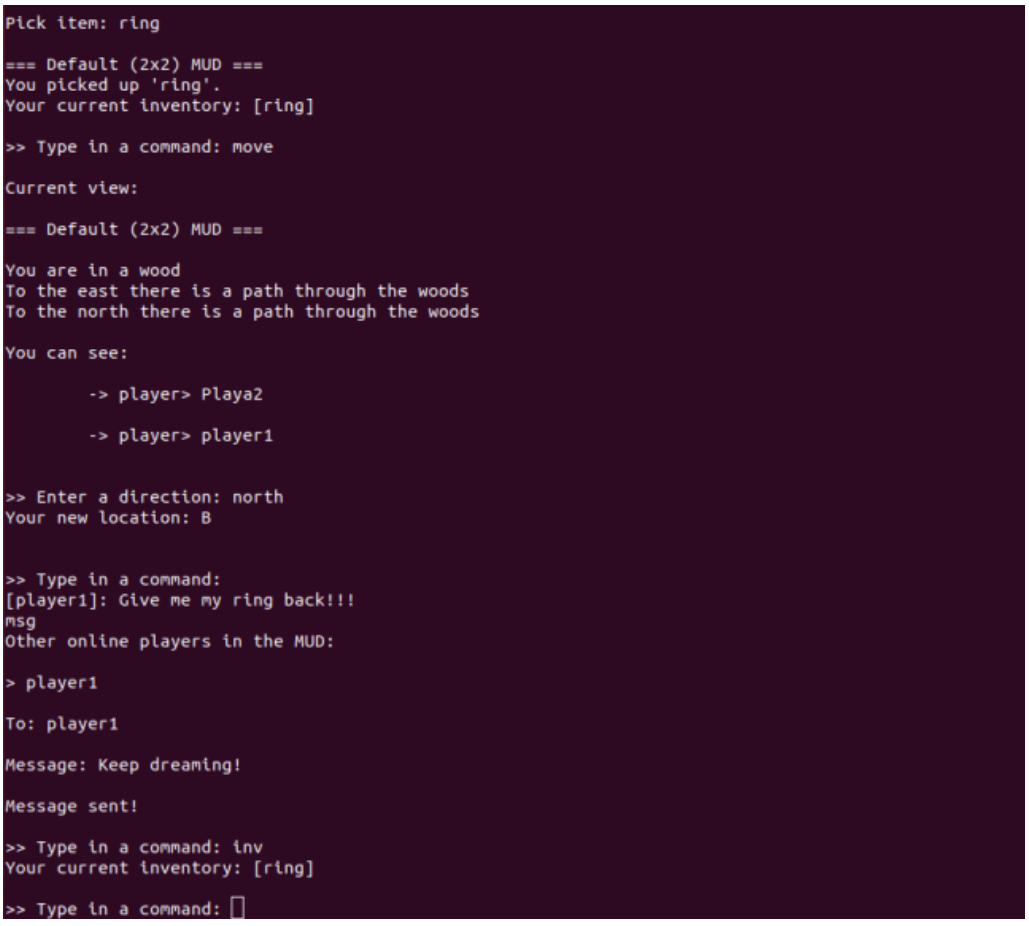

Figure 4. *Two player interaction demo - picking up a ring, moving and sending a response message.*