Overview: A MUD, or Multi-User Dungeon/Dimension/Domain, is a multi-player text environment (The player types commands and the MUD responds with text). This option would involve implementing a functional, networked MUD. They often take the form of an RPG (role-playing game); however, you are not required to make it an RPG.

Though there are many types and uses of MUDs, all of them share several common elements. A MUD consists of players, rooms, items, and MOBs (MOBILE non player characters). Multiple players can connect to the MUD at any given time. They can interact with one another, move about from room to room, and collect items that affect them or allow them to do things.

MOBs can also move from room to room, and players must have complex interactions with them. Though most MOBs are hostile towards the player, they do not all need to be. Some MOBs may offer quests, services, or games to the players.

Rooms in a MUD should be thought of as locations rather than an indoor area with walls and a ceiling (A "room" could be a clearing in the woods with a path leading to the north and another to the west, or the middle of a field with paths leading in all directions). Rooms have exits that connect them to other rooms, descriptions, and contents (including items, players and MOBs).

For more information on MUDs, refer to this article: http://en.wikipedia.org/wiki/MUD

Your Task: You are to implement a fully functional MUD Server that multiple players can connect to from different computers via a telnet client. Your server will run a single persistent game (all players that connect end up in the same MUD game).

This project gives you a certain amount of creative leeway. Your MUD could be any type of (appropriate) environment, so long as you have the required elements. Many MUDs take the form of Dungeons and Dragons style Fantasy Adventure games. In such games, players have statistics (health points, strength points, and other things that affect their effectiveness in combat) and engage MOBs in combat in order to gain experience points and advance “levels.”
Requirements:
Connections:
- Multiple players must be able to connect to the MUD at the same time over the Internet. To do so, they will launch a telnet client which will allow them to connect to the MUD Server running at a different location. Read the Client/Server section for more details.
- Game should be able to detect dropped connections: Announce to all when a player’s connection is dropped and save that player’s state.

Players:
- Players must have a name.
- Players must be able to login with a unique username and password.
- There must be an in-game text-based menu system for the creation and customization of a new player's character.
- Players must have a location (a room they are in).
- Players must have statistics that are changed via their actions in the MUD. In a traditional RPG, this would be things like hp (health point/hit points), defense, level, etc. For something less traditional, it could be number of questions answered correctly or amount of money won.
- Players must be able to carry items they find/buy/earn with them as they play.
- Players must be able to move between rooms through the use of commands (See Interaction section).
- Players must be able to interact with one another, and be able to see the locations and movements of other players in the same room as them.
- Players must be able to interact with MOBs.

Rooms:
- The MUD must have at least 30 rooms.
- Rooms must be data-driven, not hard-coded. Room templates (types) can be defined in code, but instantiated rooms in the game should be loaded from persistence.
- A room must have a description and a list of possible exits, as well as a list of contents (items, MOBs, and other players in the room). This must be viewable by the player via ‘look’ and should be displayed by default upon their entrance into the room.

Items:
- The MUD must have at least 15 different types of items.
- Items must be data-driven, not hard-coded. Item templates (types) can be defined in code, but instantiated items in the game should be loaded from persistence.
- All items must have a use in the game. You could have potions to heal damage taken, keys to open doors, special items that affect the player’s stats, equipment the player can wear, or items needed to finish a quest.
- At least 3 of the 15 required items must allow some additional interaction with the world (i.e. keys to open door, switches that change the environment, etc.).

MOBs:
- The MUD must have at least 10 different types of MOBs.
- MOBs must be able to move between rooms.
- MOBs must have complex behaviors—these can involve player interactions that take into account player statistics or items a player is carrying, or the contents of the room that the MOB is in, etc.
• MOBs should have actions that are triggered periodically, without direct interaction from players. For instance, movement, speech, interacting with other MOBs, etc.
• MOBs must be data-driven, not hard-coded. MOB templates (types) can be defined in code, but instantiated MOBs in the game should be loaded from persistence.

Interaction:
• When players are in the system, they should be able to send input and receive output without interference from other players.
• Players must be able to interact with the game with a set of commands that include at least the following:
  • A set of movement commands that allows players to navigate through rooms (north, south, east, west, up, down, etc)
  • look (shows description of the room that the player is in, or if an argument is provided, such as an item/player/MOB in the room, it should provide the description of said item/player/MOB). This command gives a 360 degree report of the environment (the player is not assumed to be looking in a specific direction).
  • commands (lists all the commands useable by a player)
  • ooc <message> (Out of Character channel—the basic MUD wide chat command —message goes to everyone currently connected)
  • who (lists all players that are logged in)
  • say (sends a message to all players in the same room as the player executing the command)
  • tell <player> <message> (sends a message to only the player targeted)
  • score (displays the players current status/information)
  • give <item> <player> (gives item in your inventory to player/MOB)
  • get <item> (gets item from room)
  • get <item> <target> (gets item from player/MOB/item)
  • inventory (lists the items that you are carrying)
  • drop <item> (drops an item from your inventory to the room)
  • use <item> (executes the item’s default behavior)
  • wear <item> (if the item is wearable, equips the item on the player)
  • remove <item> (if the item is equipped, un-equips the item and puts it into the player’s inventory)
  • eq (displays a list of items currently equipped by the player)
  • quit (allows a player/client to exit the system—should not shutdown the MUD server)
  • shutdown (saves the MUD’s data and shuts the server down – see next section)

System:
• MUD must be persistent. Changes made to players and the MUD must remain even after the MUD is shutdown and restarted. Note: You are allowed to have a data bootstrap class which will do an initial population of rooms, MOBs, items, and a base set of players the first time the MUD server is started. However, upon subsequent server restarts, all MUD data must be loaded from the persistence model.
• MUD must be able to be shutdown from within the game. This function should not be available to all players (i.e. Reserved for administrative-specific players, etc).
Consistency:
- The MUD should be logically consistent. For example, a player should only be able to give items to another player/MOB if they are in the same room. Also, walking north to the next room and then south should take the player back to the original room they were in (unless the area is designed to randomly connect rooms to create a maze like environment. If so, the area should be clearly marked with a sign before entering it or a warning in the description of the room before entering it).

Client/Server:
- The main portion of the MUD will be written as the MUD Server. All interactions with the client should be sent to the server and should trigger an appropriate response by the server that is sent back to the client. The server should run as a terminal based program on one computer and must accept connections from telnet clients that run on other computers over the internet.
- The client you will be using to connect to your MUD is the command line program telnet: http://en.wikipedia.org/wiki/Telnet. Your MUD must fully support connections from telnet, even if you choose to implement an additional GUI client for extra credit.

Extra Credit (10 Points Max):

This is by no means an exhaustive list of extra credit features, but merely a list of suggestions for potential extra features. Some of these are much harder to implement than others, and are worth more accordingly. Talk to your grader if you have other ideas to ensure that you receive points for your feature.
- **Room Behaviors (2 Points)** - Rooms that have behaviors independent of the player (for instance, weather).
- **Dynamic MUD Editor (3-8 Points)** - An in-game menu system for creating new items, mobs and rooms. This could also use a GUI to simplify the process. This option should not be available to all players.
  - **Shareable MUDs (1 Point)** – Provide a way for MUDs created with the editor above to be shared.
- **Complex Skills/Classes (2 Points)** - Complex skills and classes that allow players to have more sophisticated and a wider variety of interactions with their environment.
- **ANSI Color Support (1 Point)** - ANSI color schemes for text output.
- **Player Grouping (2 Points)** - if a player joins a group, he or she follows the movement of the leader, and executes certain commands (for instance, attack) if the leader executes them.
- **Emoting/Posing (1 Point)** - A command that allows a player to write a sentence from the third person perspective: For example, the following command: ‘emote looks around the room curiously.’ outputs the following to everyone in the same room: ‘<player's name> looks around the room curiously’.
- **Player Preferences (1+ Points depending on complexity)** - Player options such as removing oneself from the ooc channel, turning color (if implemented) on and off, etc. These preferences must persist.
- **Room Exit Description (1 Point)** - Room exits that have descriptions viewable with the ‘look’ command. For example, when you're in a room with exits to the north and south, if you type 'look north', you should see a description of the path leading north.
• **Social Commands (1 Point)** – An implementation of at least 10 “social” commands (such as, ‘giggle, laugh, wink, slap,’ etc, that would display something like, `<player> giggles at <target>` or just `<player> giggles`).

• **Abbreviated Commands (1 Point)** - Players able to type abbreviated versions of command names (for instance, ‘l’ for look, ‘sc’ for score, ‘”’ for say, etc).
  o **User defined aliases (1 Point)** – In addition to the above, players would be able to create their own abbreviations for commands that they can use in-game. These should be persistent.

• **GUI Interface (1+ Points depending on complexity)** – A separate GUI client.

• **Mini-Games (1+ Points depending on complexity)** – Mini games are games within a game. This could take the form of a card game/dice game or something more complicated. This could be played between players or between players and MOBs.

• **Administrator commands (2 Points for at least 4 commands)** – Allow an “administrator” user to have special commands in the game, which cause responses in the game for all players. Examples include **shutdown, restart, kick, ban, banbyIP, deleteprofile**, and many more. Maximum of 2 points.

• **Database persistence (6 points)**: Utilizing an RDBMS (like MySQL) and a JDBC library to store and retrieve all persistent data for your MUD, instead of file based persistence.