**Chapter 8**

Ex. 8.1 – what does the application (zuul-bad) do?

It is a text based adventure game where the player can move through an environment.

What commands does the game accept?

The game accepts go, go east, go west, go north, go south, help, and quit.

What do the commands do?

Any of the go commands moves the player to the next area in the next area. The help command gives information about the other commands. The quit command ends the game.

How many rooms currently exist at the moment?

There are five rooms. Including the room the player starts in.

Day a map of existing rooms.

[Pub] 🡨 [outside] → [theater]

↓

[lab] →[office]

Ex 8.2 - What is each classes purpose?

The project is composed of 5 classes; Game, Command, CommandWords, Room, and Parser. The Game class is the main class of the project. It is utilizes the other classes and runs the game until the player ends it. The Command calss receives commands from the player. The CommandWords class stores the valid commands. The Room class holds the current location and is used to set the connections between areas. The Parser class is used to parse commands provide by the player.

Ex 8.3 – it could be to place a text based adventure game at a manor or some kind of estate. it could have locations such as the mansion and some of its rooms a warehouse/garage, the gardens themselves could be a set of rooms. Maybe it is an adventure because you are trying to escape or find someone. I vote the later of the reasons to be there.

Ex. 8.4 – done and saved as Ivory manor

Ex. 8.5 – implement the printLocationinfo in your project.

Done and saved.

Ex. 8.6 – Make the changes we have described to Room and Game.

Done and saved.

Ex. 8.7 – Fix the printLocationInfo method because the last exercise.

Done and saved.

Ex. 8.8 – Implement changes

Done and saved

Ex. 8.9 – Look up the keyset method in the documentation of HashMap. What does it do?

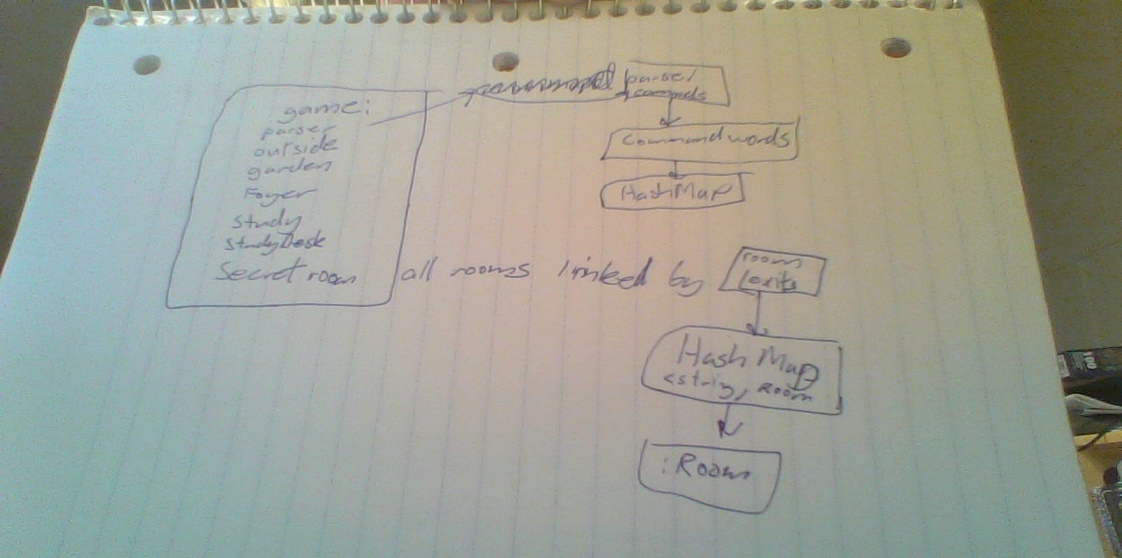
The keySet method returns a set of the keys contained in the HashMap object.

Ex. 8.10 – Explain, in detail how the getExitString methid from 8.7 works.

In short, the method returns all available exits for the current room. It does this by creating a String called “returnString”. Then a for-each loop iterates over the “exits” HashMap adding the exits that are not null to the string and displaying it.

Ex. 8.11 – Implement the given code.

Done and Saved.

Ex. 8.12 – Draw and object diagram with all objects in your game, the way they are just after starting the game. 

Ex. 8.13– How does the object diagram change when you execute the go command?

After starting the game the diagram remains similar but the room we are in changes along with its exits to match the new room.

Ex. 8.14– Add the look command to your game.

Done and Saved

Ex. 8.15– Add another command to the game. (adding eat like suggested)

Done and saved

Ex. 8.16 – Implement the discussed changes.

Done and saved

Ex. 8.17 – If you add another new command, do you still need to change the game class?

No, you no longer need to change the game class because all commands stored in and array. With the new print method a for-each loop goes over every element of the array and prints it so there is no way to miss one.

Ex. 8.18 – Implement the discussed changes.

Done and saved

Ex. 8.19– Find out what the model view control patter is. How is it related to what the text has discussed?

The model view control pattern or MVC divides a program into three pieces. One modles objects and represents the information. The next views objects and deals with the display. The third is control objects which changes and modifies the previous two. To implement this pattern into the zuul project we would have to break the game class in two. One of these new classes will represent the model and the second will make changes to the model.

Ex. 8.20– Extend the project so that a room can have a single object in it. Items have a description and a weight. When a player enters the room the item should be created and its information displayed.

Attempted.

Ex. 8.21– How should the information be produced about an item present in a room? Which class should produce this string? Which class should print it? Make the necessary changes

A class called item should be created and it should hold an items information. It should have methods responsible for printing the details of it and these methods should be called from the room class.

Ex. 8.22– Modify the project so rooms can store many objects.

Done and Saved

Ex. 8.23– Implement a back command

Done and saved

Ex. 8.24– Test the new command. Does it work as expected?

For the most part yes, it works but if it is called a second time in a row it doesn’t work. I expected it would have trouble with that because we had no way of storing more than one move. Absent mindedly I had expected it to just keep flipping between the two rooms but after thinking about the code I realized that was foolish, as it never re-assigned the previous room.

Ex. 8.25– See 8.24 answer

Ex. 8.26– Implement the back command so that if implemented enough it will take the player back to the start of the game. Use a “Stack” to do this.

Done and saved.

Ex. 8.27– What are some baseline functionality tests we might wish to establid for the current version of the game?

Well step one should be see if it complies and runs. If it does that we should then start testing its features to ensure they are all working properly or at least as expected.

Ex. 8.28– Refactor the project to introduce a separate Player class. A player Object should store at least the current room of the player but may also want to store the player name

Ex. 8.29– Implement an extension to allow the player to take and drop items

Attempted

Ex. 8.30–Extend so that player can hold multiple items

Attempted

Ex. 8.31– add a weight restriction to what a player can carry.

Attempted

Ex. 8.32– implement an items command that prints out all carried items and their total weight.

Attempted

Ex. 8.33– Add a magic cookie to a room and have it increase carry weight If eaten. If it makes more sense make up another object that fits better and does the same thing.

Attempted

Ex. 8.34–

Ex. 8.35–

Ex. 8.36 –

Ex. 8.37 –

Ex. 8.38 –

Ex. 8.39 –

Ex. 8.40 –

Ex. 8.41 –

Ex. 8.42 –

Ex. 8.43 –

Ex. 8.44 –

Ex. 8.45 –

Ex. 8.46 –

Ex. 8.47 –

Ex. 8.48 –

Ex. 8.49 –