**Report**

**NOTE**

* **In the ‘List’ Class – please changed the file path of item.txt to its location on your machine.**
* **I have used text colours for the output to terminal – depending on your build of Windows these may not work. The variable begins ‘ANSI’ for each one so can be easily removed if this is the case.**

In this section of the report I will briefly summarise the role of each class and how it works alongside the others.

**Room & Temple**

The Room & TempleOfWishes classes haven’t changed a great deal from the previous assignment. The only notable differences are:

* The Room class now includes an (initially) empty LinkedList called contents.
* The Room class now has a getter method that returns itself
* The TempleOfWishes size is set as a fixed parameter (8) of the constructor, rather than a user input.

**Item**

The item class has 3 properties – size, value and integer.

There are two random generator methods in the class that set the size and value within their permitted ranges. These function are in the constructor.

The constructor also accepts in the a string that will represent the item name. It receives this name from the items.txt file when the list is created.

**List**

This class represents the list of artefacts. It reads in the text file full of names, and create as many items as there are names in the file. It passes the item name as a parameter of the Item constructor.

**Hero**

This class represents the player. It has a LinkedList to represent the inventory, a current Room (as a room), and then two integer values to represent the X,Y coordinates. The row and column are altered by the navigation process, and then the currentRow and currentColumn are passed to the ‘getRoom’ method.

The hero also has capacity and total value integers. These are incremented based on the backpack contents. When the hero picks up an item, it will only be allowed if capacity + the size of the new item < 50.

**DungeonMaster**

Houses the main method. Essentially instantiates one of each object and passes it to the Game class.

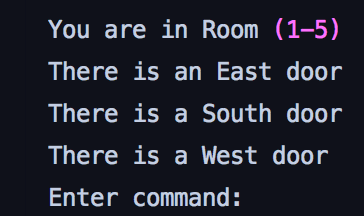
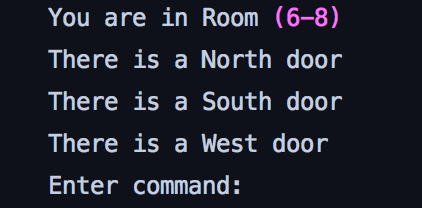
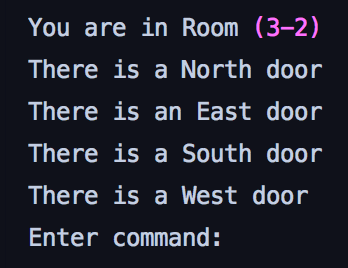
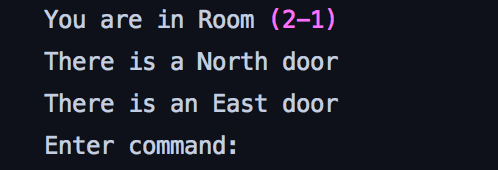
**Game**

The primary worker class for the game logic.

Based around a switch statement with cases for each valid input that call the relevant functions. It is passed the hero, temple and gamestate from the main (dungeonmaster) class.

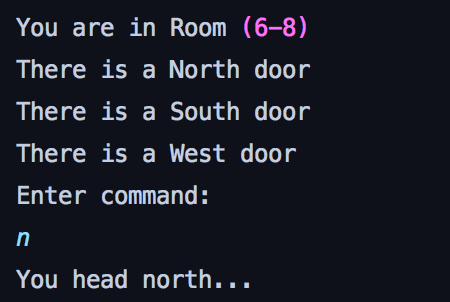
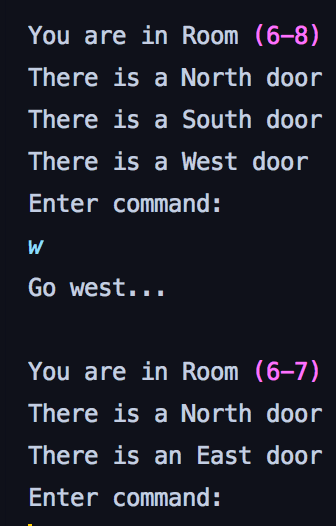
**Assignment 9 – Testing**

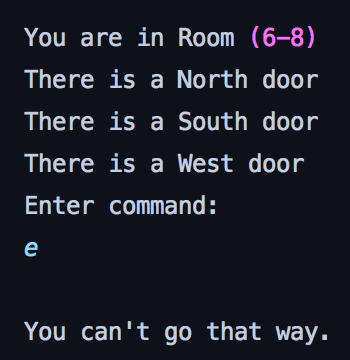
**Test 1 – Works As Expected**

Check that the Hero starts in a different random room each time.

**Test 2 – Works as Expected**

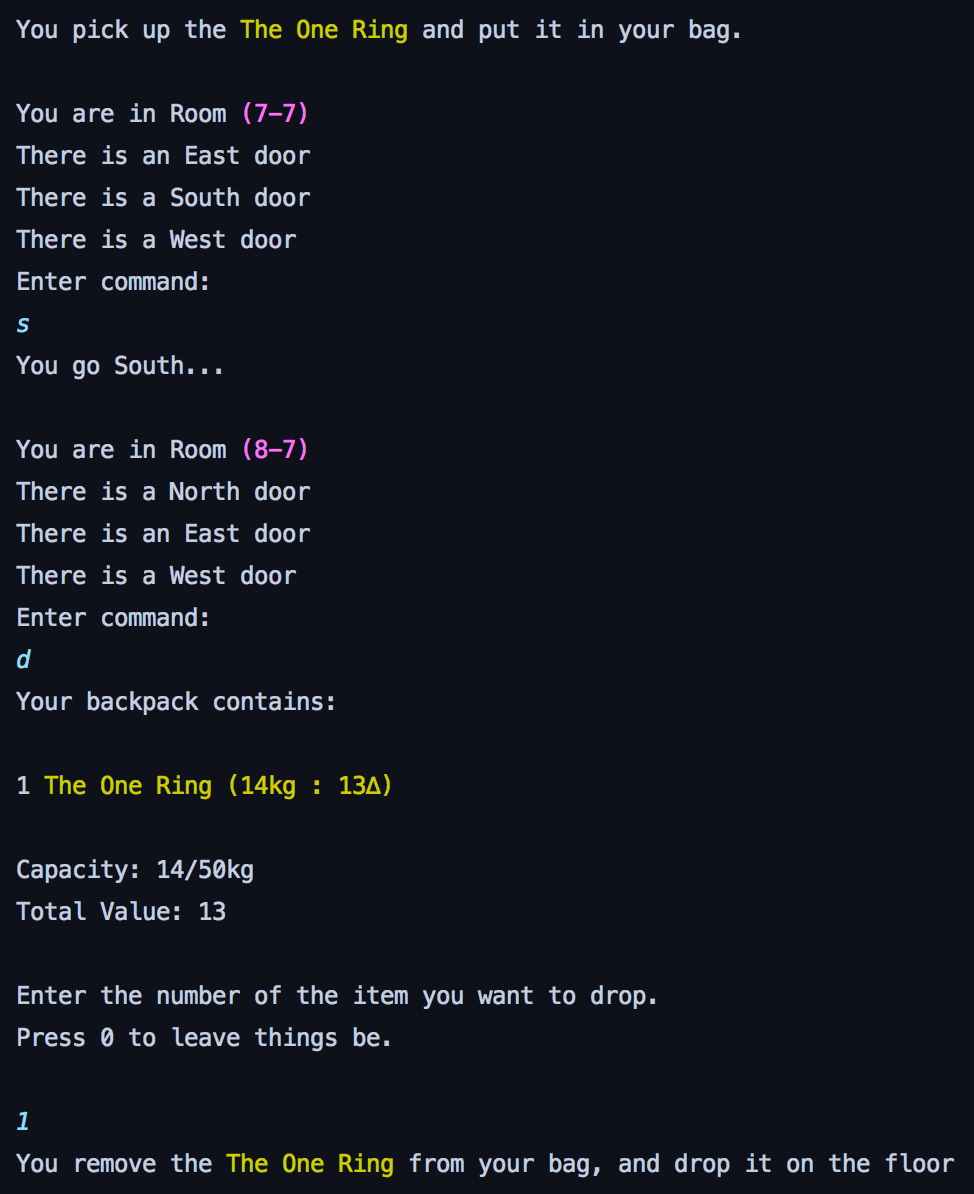
Check that you can only pass through doors that exist.

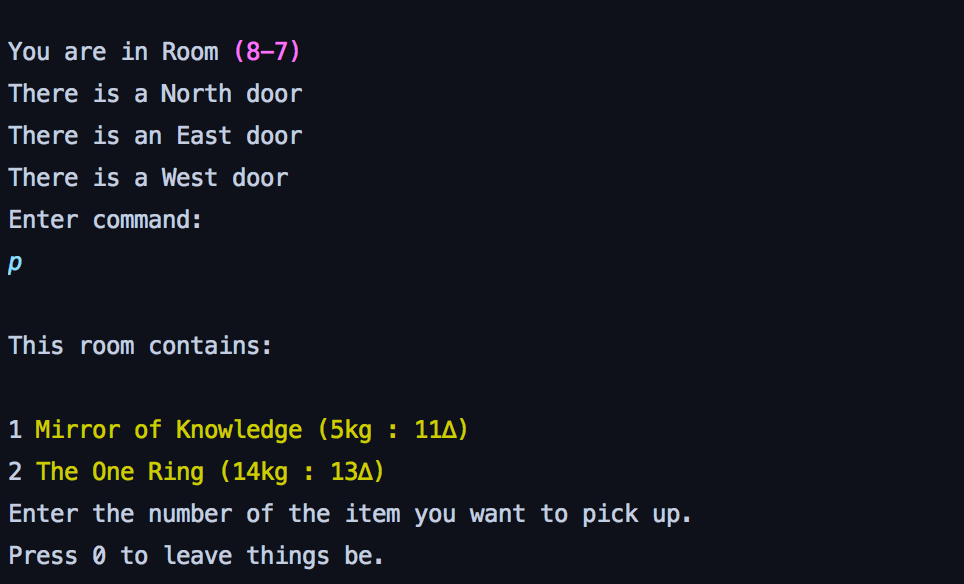




**Test 3 – Works as expected**

Getting an item from a room, then dropping it into a different room.

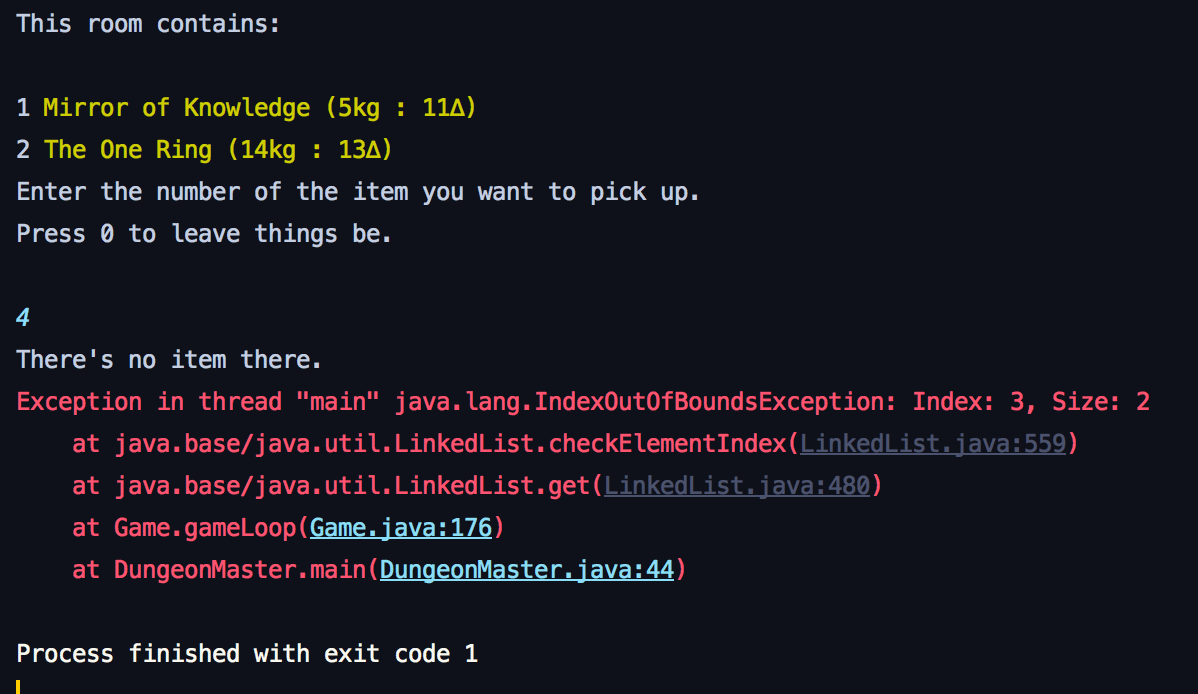




**Test 4**

Test what happens if player tries to pick up a non existent item.

Result – prints custom error message but still crashes.



**Test 5 – Works as expected**

Overfilling the bag – expect an error message saying the bag is full.

