Coursework 2: Secret Chambers

Krishna Prasanna Kumar (K21004839)

This game is called Secret Chambers, the aim of the game is to kill NPCs throughout the dungeon and obtain the special key which will allow you to open the coveted Secret Treasure Room where unimaginable riches await you. Along the way you can find unique potions and weapons that will help you take down the enemies that are roaming around the Secret Chambers. The game has several rooms that the user can move around finding items along the way, some of which can be picked up and others that cannot.

I have created several classes with their own unique jobs, they can contain several types of collections such as HashSets, HashMaps and ArrayLists. They all have several methods that are used to manipulate the data in the class that the fields are associated with for responsibility-driven design.

How I completed the Base Tasks:

- The game has several locations/rooms. - I created several rooms in the createRooms method in the World class. In this method I create 12 rooms using the Room class each with 2 parameters, a short description (which is usually just the name of the room) and a long description which is a longer description about the room.
- The player can walk through the locations. – Whenever the user types go and a valid direction in the room, they are moved to a new room allowing for movement, this is done by creating a HashMap that links a exit to a room. A method then gets the new room depending on the exit input by the user, for example if they enter “go east” the method will map the exit to the room that is in the east direction and will update the player’s current room to the new room.
- There are items in some rooms, every room can hold any number of items. Some items can be picked up by the player, others can’t. – I first created a new class called “Item” which contains important accessor methods and contains important fields about each item. Each regular Item takes in 3 parameters, the name, weight and a description of the item. I then added an ArrayList of type Item in the room class, this because every room can hold multiple items when the game is created. With the ArrayList I created a method that adds Items to the room which is called a couple of times in the AddItems method in the World class which adds the items to specific rooms, for example a to add Item bread to a Kitchen room it would look like: kitchen.addItem(bread). Some items cannot be picked up because by default they have a greater weight than the bag can hold.
- The player can carry some items with him. Every item has a weight. The player can carry items only up to a certain total weight. – I created a new Class called Backpack which would be the inventory for the player, it has a max weight of 50 and consists of a HashSet of Item which would hold all items that are picked up by the player. When the user uses the take command and enters a valid item, the item will be removed from the room and added to the backpack and the weight of the backpack is updated. If there is not enough space, then it prints a statement that you cannot pick up the item.

- The player can win, there must be some situation that is recognised as the end of the game where the player is informed that they have won. – I have added two treasure rooms by creating a sub class of the room class, they can only be accessed by a door, when the door is unlocked, and the player enters the game ends. Depending on which room the player entered (The secret treasure room or the fake treasure room) they either win or lose.
- Implement a command "back" that takes you back to the last room you've been in. – I have implemented a stack of room type which will push the previous room to the stack whenever the user enters a new room. Whenever the user inputs "back" the room at the top of the stack is popped and the current room is updated to that room.
- Add at least four new commands. – "look", "take", "backpack", "attack". The look command prints the details of the current room like the items and NPC's present and the exits. I used several methods that were created in the Room class that prints out details of the room. The take command is used to pick up an item from the current room, I added a method that removes the item from the current room and adds the item to the backpack. The backpack command displays all items that are currently in the backpack by using a for each loop on the HashSet of items in the backpack class. The attack command is used to attack NPC's, I have added a method that subtracts health from both the player and the NPC when the attack command is called.

How I completed the challenge tasks:

- Add Characters to your game – To create characters I made a new class called Enemy which will be used to create three characters. The Enemy class constructor takes in the name, enemy health, and enemy damage. A strength object is created for each enemy within the enemy class which takes in the enemy health and damage. To make the characters move I generate a key set of exits for each room using the HashMap for exits in the room class. I then add these to an Array List and use a Random object to generate a random number within the size of the array list. Then I get the exit at that index and call the get exit method for all the characters.
- Extend the parser to recognise 3-word commands – I extended the parser class so that it takes in 3 words and stores them instead of 2. Then I extended the command class by adding accessor and mutator methods for the third word. I then updated the World class by adding the "drink" command to the switch case statement. This command currently only works for potions.
- Add a magic transporter room – I created a subclass of room for the transporter room, here I create a Room Randomiser object which is responsible for returning a random room from an array list of rooms that is created in the world class. The transporter room will then return the room and a method in the world class will set the player's current room to the new random room.

My own challenge tasks:

- Adding keys and doors – For this task I created a key subclass of item and a separate class for doors. A key will have the same parameters as an item but will also have a first room parameter. This will be the room that the key can be used in to open a door. A door on the other hand will take in a second room and a direction as parameters, the second room is the room that will be available for

the player to move into when the door is opened, and the direction is the direction in which the door opened in. I created a new command called “open” which will open doors, it currently can be used by typing “open door” when you have the right key and in the right room. This method will check whether the keys and the doors match up and will accordingly open the door.

- Strength System – This class is used to initialise the player’s and the enemy’s health systems. I create strength objects in the player and enemy classes and will call their methods if it is necessary to update health or damage.
- Adding weapons – I created a weapon class which extends the Item class, it has damage as an additional parameter. I can distinguish weapons from other items by using a Boolean and casting methods. They are also used to attack enemies and the backpack will be checked to see if a weapon is present before the enemy is attacked.

Code Quality Considerations:

Coupling – When implementing the items, I considered coupling and decided to add a HashSet in both the backpack class and the room class instead of creating fields or variables that sets it to a room. This helped me later down the line as when I created more items, I only had to add a couple of lines.

Cohesion – When considering how to add the inventory I was thinking about adding it to the Player class, but then I realised that the inventory would require a lot more methods than I first thought. For cohesion purposes I created a new Backpack class which also has separate methods for dealing with special items like the keys and the weapons.

Responsibility-Driven Design – When I was creating the backpack class I was thinking about where to store all the weight information. When I considered responsibility-driven design I decided to add all of it to the backpack class as that class is also responsible for manipulating the data. Then I used several accessors and mutator methods in the backpack class instead.

Maintainability – When I was programming, I was constantly adding and updating the Method comments as they help me and others to understand what methods do and what their role is. This was extremely useful when debugging as it was easy to find errors when comments and appropriate method name were used. This will be really useful when considering maintenance.

Walk-Through:

There are 14 commands in the game that can be used to play the game. You will need to use the go command and a valid exit to move around rooms, for example “go west” will make the player move to the room in the west direction. When you find an item in a room you will need to type “take” and the item name to pick it up, you will need to ensure that your backpack has enough space, and you can only hold one key and weapon at a time so make sure you drop a weapon you’re holding if you want to pick up a new one. When you enter a room, you will be told if there were any locked doors in the room which you would need a key to unlock, the items in the room, the exits and if there is an NPC. If there is a NPC, you can use the attack command to attack then NPC, make sure to have a weapon when attacking an NPC as you will take damage if you try to attack without a weapon. If you

would like to check your current health and damage, then use the “strength” command. The NPCs are really important because they are holding the keys so without killing them you will not win the game, however there is a NPC which is holding a fake key to a fake treasure room which would lead to you losing the game instantly.

If you encounter the transporter room it won’t be instantly recognisable because there will be a cryptic message displayed when you enter the room. However, if you do notice and would like to transport, use the “transport” command to move to a new room randomly.

The drink command is the 3-word command that can only be used on potions but make sure to use them wisely because if you drink the health potion too early then you may end up dying by the NPC’s.

Essentially, you will have to find the correct key by killing the right NPC and will have to open the correct door to win the game.

Known bugs or problems:

One issue I have is that the potions will only be used as one words throughout the programme, for example the damage potion will be referred to as “damage” so you would have to type “take damage” to pick up the potion. However, to use the potion you must type “drink damage potion” as a 3-word command. Another problem is that the enemy random movement will only happen when the command is go even if you spam “go” multiple times which is an issue because the player is not moving but the enemies are. Plus sometimes you can go in a loop trying to find the enemies because you are out of cycle, for example when you start the game the enemies are spawned at an odd number of rooms away from you, so you will meet them eventually, however if you do the issue I was talking about where you type “go” without actually moving rooms then the enemies will be an even number of spaces away so if you move then the enemy may go to the room that you were previously in resulting in you never meeting them, to reset this just type “go” several times to check whether an enemy spawns in your room.