I started writing the game without first making a UML diagram and the first try was mostly a failure. It did produce basic results like moving between rooms, but it was hard to implement new functions since most of the data was in lists and hard to handle(I had to manipulate list indexes a fair amount for most operations).

Then I made the UML diagram and re-wrote the script from the ground up. The class diagram helped a lot in organising the variables and program flow.

I made an Item class since I knew that I would require different types of items, and created the different types (stationary,movable,usable). I included the pick\_up and drop methods on the movable and usable classes and added a use method for the usable only. I encountered some issues when calling the method though,so I found a workaround by creating a supporting list for method permissions.

Then I made a door class, since doors with their attributes are all that’s needed to generate the playing field,since a door’s orientation in addition with the rooms on each side is enough to link the rooms together. I added 2 direction and 2 room attributes and a status attribute with regards to their locked condition. Since it would take an extra door id attribute and a specific item id for the relevant key to open a specific door, I chose to make my key a skeleton key and use it for every door for this example. Would I require a specific key for each door I would probably use the “text” attribute of the key item(an attribute I used for items with use to make the user read how the item was used) and use it as an identifier for the correct key.

I then created lock/unlock/open/close methods within the class and they did turn out to work great as they made calling a door method just by the door’s id quite easy.

The major problem that I had was handling the “self” attribute within my classes and when calling the methods. I didn’t really understand how it works and sometimes it worked fine to call a method by instance\_name.method(args) but there were a few times where some issues occurred. Perhaps it was due to increased confusion due to inexperience and maybe that’s why I didn’t have the same issues later on when I wrote the door methods.

I did some things differently than the assignment asked, like not describing the room the player enters after a move command. I found that running the game and wanting, for example, to move to a room 3 spaces away, would fill the space with needless text. It felt nicer getting the information you ask for rather than too much of it.

I also used some string manipulation to make some things easier for the user, like using .lower() methods in order to not invalidate inputs with capital letters.

All in all it was a very interesting project and made me think about extending and improving it in various ways, like how to generate a random config file that will have a consistent layout and explore a new room each time, or adding some basic graphics and hotkeys to make it more interactive and less text-heavy.