Program Report
Team L

UML:

In making our game we went through many iterations of design which means we changed our UML design and overall structure of our code as we we're coding. Because of this our current implementation of our game is not ideal in the sense it does not follow any specific design pattern, or the skeletal UML that was presented in class. After reviewing our code, UML and the general situation of our project with Mrs. Wilson and Prof. Tareque, we came to the conclusion that we could not change our program to follow the guidelines provided to meet the deadline. With this in mind we discussed that a suitable solution would be to present a "Redesign" in the form of an additional UML diagram, and keep our original design. This redesign would be the sort of perfect scenario where we had more time to work on the project, as we understand where we went wrong, and how to fix it, but simply lack the time between our other classes and work schedules.

Redesign:

After reviewing the design patterns provided in the labs, talking to Mrs. Wilson, and a lot of reflection we came up with Appendix 4 as a way to reimplement our code so things would utilize polymorphism, encapsulation, and proper inheritance. In this model rooms have been condensed into 1 centralized class with various attributes, and ingredient room, puzzle room, and npc room extend rooms as they add attributes, and methods that are unique to them. We would create objects of each type of room in replacement to our 10 separate room classes in Appendix 3. Items would be reworked into an object system as well with each item containing a name and description, which would be passed into characters inventory via pointers. Character has been implemented to have a centralized point to store all variables in relation to a character instead of our original implementation where health, inventory, and current room are spread across 4 classes. Player extends the Character by giving a user actions, Friendly npc contains purely dialogue as they will never take damage, and Hostile npc has health but no dialogue as they will only ever be hit and not return conversation. Finally Command UI serves as a way to display the information within each class, and take in a user's decisions. Overall we believe this to be a much better design as it is significantly more flexible than our current design and offers room for growth, modification, and maintenance capabilities our current program could only dream of.

Design process explained:

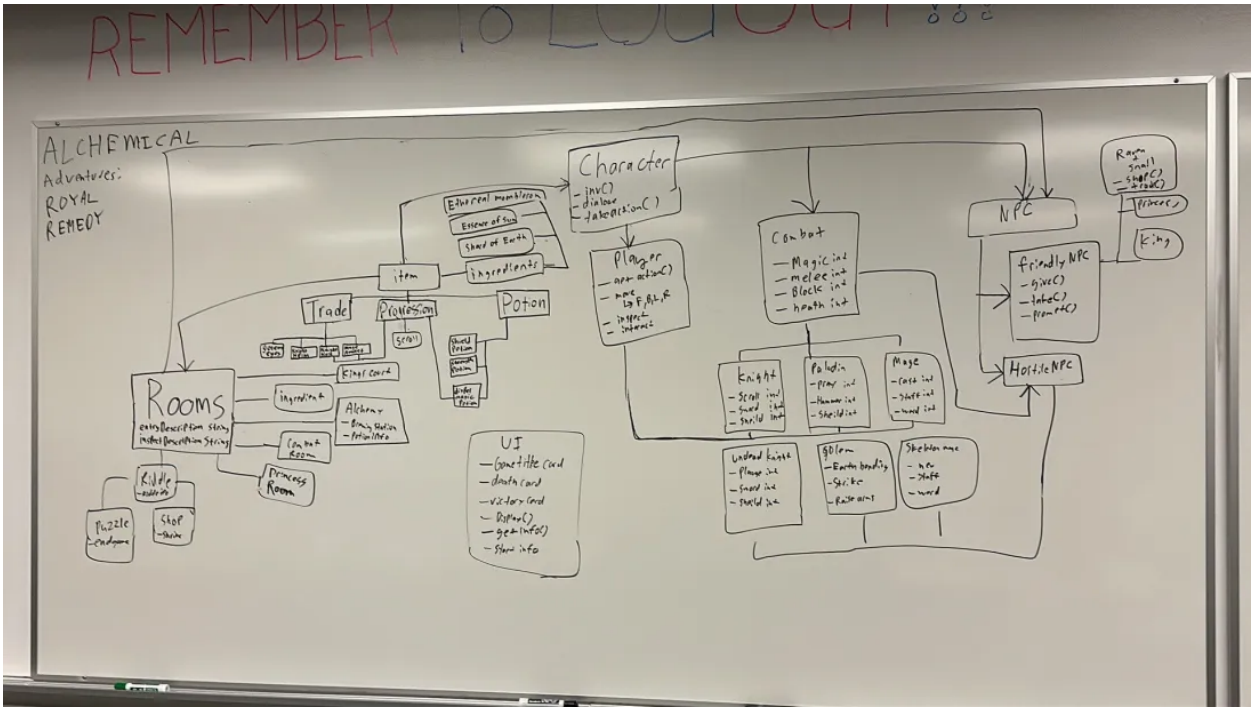
In reference to Appendix 1, this is the first iteration of our UML diagram and it clearly shows the ideas of polymorphism and inheritance. Because it is a rough draft and was never intended to be used for marking purposes it lacks clarity, and some critical points such as the exact functions, attributes, and relations between classes, but conceptually it shows the framework we were aiming for. After implementing part of Appendix 1 we had a shift in mindset, that frankly none of us can actually explain, but we decided to change the UML to a simpler design. This was the beginning of our derailment, Appendix 2 cuts down a lot of the polymorphic and encapsulating aspects. In hindsight this was a terrible idea and we still have no clue what pushed us to do this. Finally we have Appendix 3 which is our current UML design. This is a product of not coding a large project since 2620, inexperience, and not fully grasping the concepts of the 2720 practices.

Requirements:

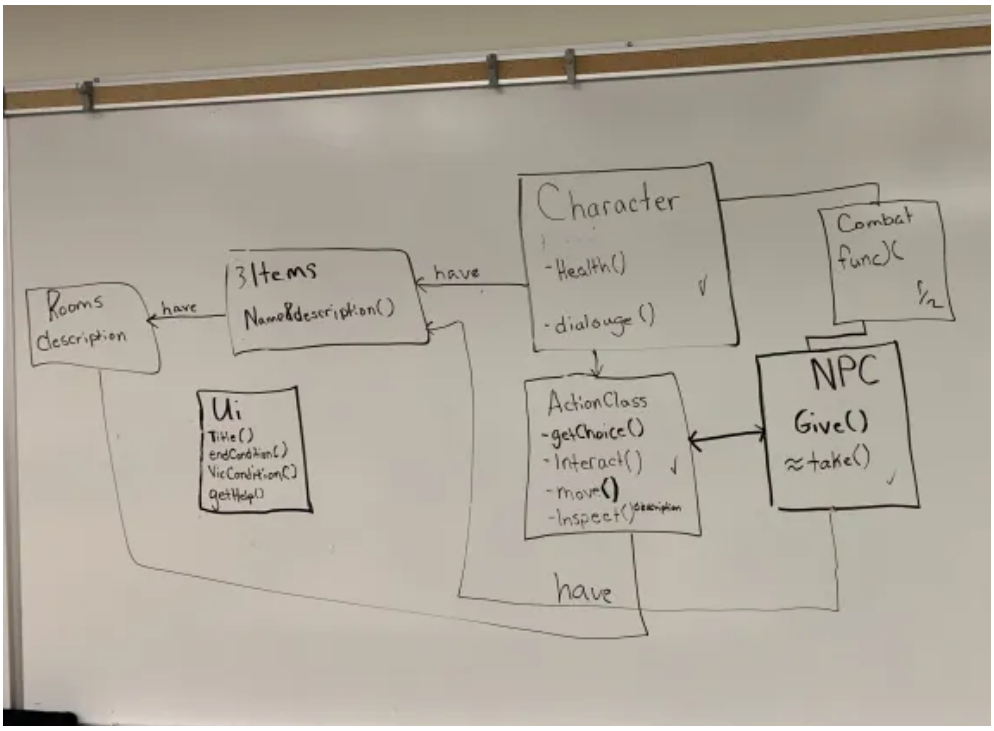
The current implementations hit all the requirements of the assignment PDF, but the structure of the code is not ideal.

- We have 10 total environments, 3 combat rooms, 3 ingredient rooms, 2 puzzle rooms, and a shop room.
- 2 NPCs in the form of a trader, and an information broker.
- 3 exploration options being inspect, move, and interact
- 5 usable items, 3 used in puzzles, and 2 used as trade interactions
- 1 overarching plot of collecting ingredients to make a potion to save a princess
- 1 PDF called AlchemicalAdventure.pdf for help
- 40+ tests that test the critical points of inputs and outputs. We exempted a large majority of our cout statements from testing as they are generally copied from somewhere else.
- A purely text and ASCII game with all interactions from the command line.

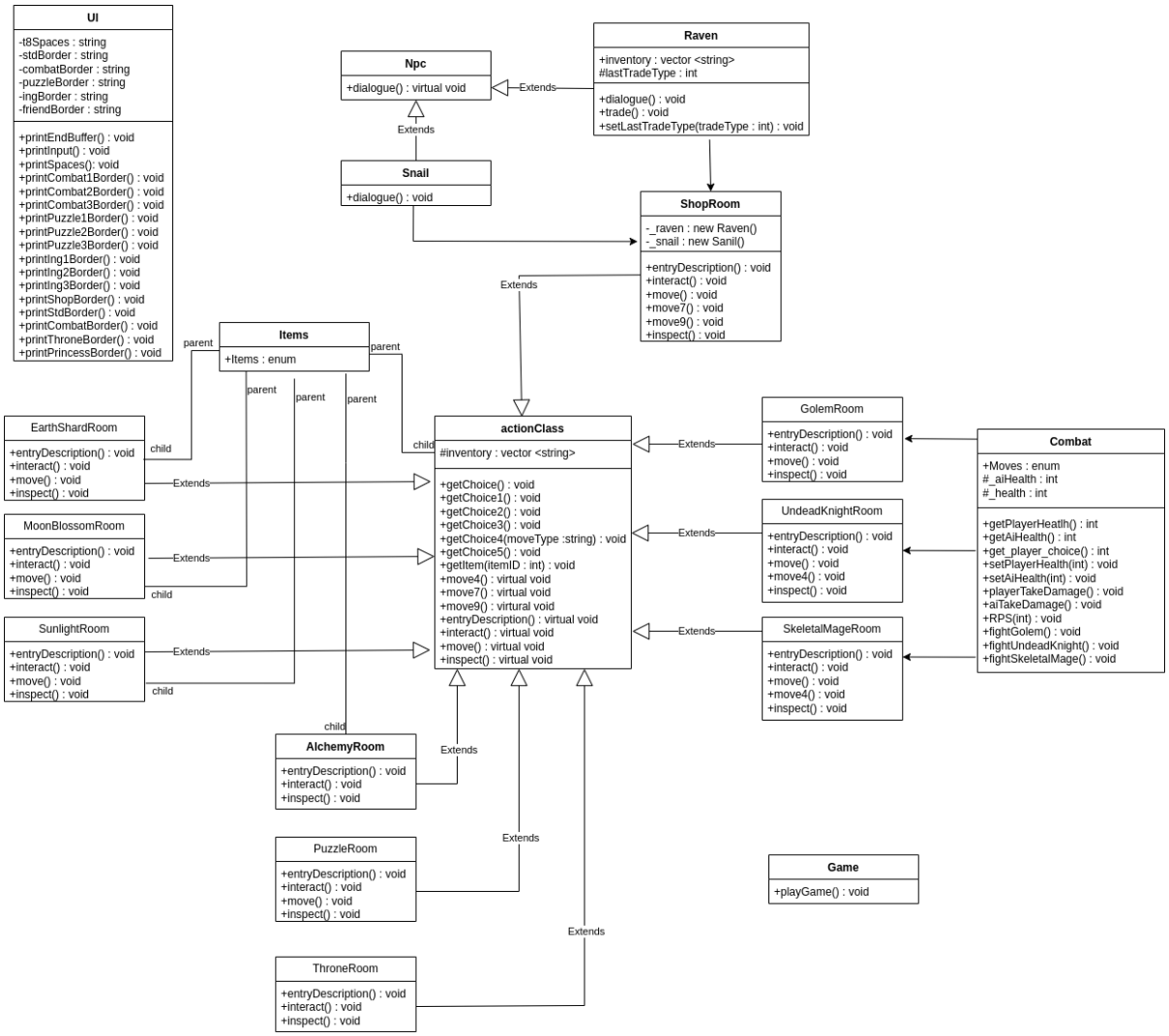
Appendix 1: Original design



Appendix 2: The first step to derailment



Appendix 3: Current implementation



Appendix 4: Proposed redesign