*1. List the features that were implemented (table with ID and title).*

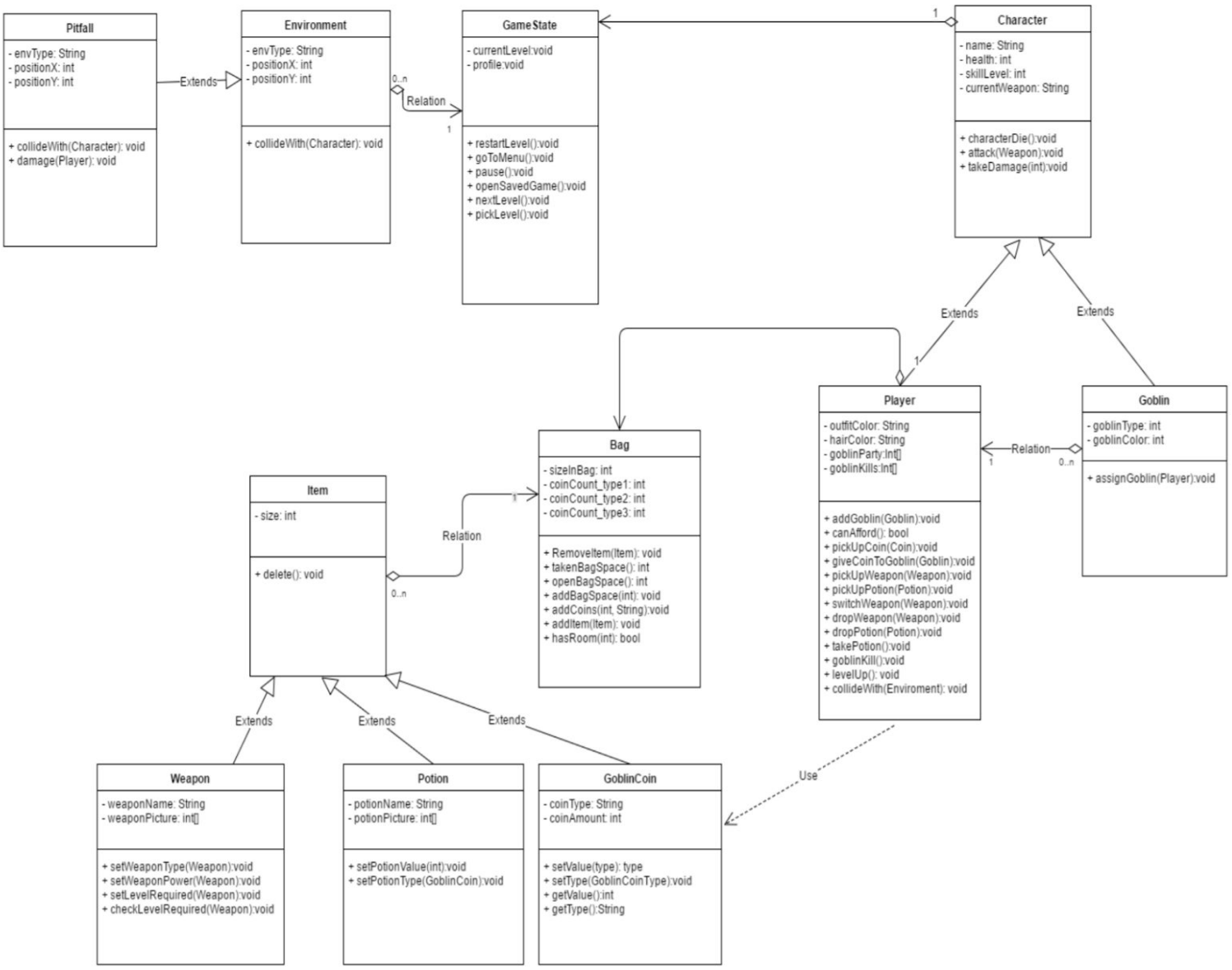
| Feature ID | Original Feature Description | Implementation Description |
|---|---|---|
| FR-2 | The Players strength shall increase by a set amount of points for each goblin it defeats. | When the Player defeats a goblin, its game level and strength level increases. |
| FR-3 | The user shall be able to make friends with a good goblin by offering the goblin a set amount of goblin treats. | Player can give coins to Good Goblins and add them to their party. Can have maximum of 8 Goblins in party at a given time. |
| UR-2 | As a user I need to be able to battle goblins, and keep track of total number of goblins killed so that I can advance in the game. | Player can battle a goblin and has an attribute which keeps track of the number of Goblin it has defeated. |
| UR-3 | As a user I need to be able to make friends with nice goblins and have them. follow me so that I have them to assist me on my journey. | Player can pay friendly Goblins to join the party if the player has sufficient coins of the Goblin's color. The Goblin then joins the party as a follower. |
| UR-4 | As a user I need to be able to pick up different types of weapons with different attributes, so that I can have control over the fighting style of my player. | Player can accumulate weapons of different types, that have different attributes. The player can choose which weapon to equip if they have more than one. |
| UR-6 | As a user I need to be able to collect Goblin Coins so that I can use them make friends with nice Goblins. | Player picks up coins that appear in the world. These coins are of different colors, and can be used to buy the loyalty of Good Goblins of the same color of the coins. |
| NFR-1 | The user shall be able to start the game using a minimum of 2 commands, once they are in the folder that contains the game. | User is able to start the game by simply typing in the command line, ./runGame.sh |
| NFR-3 | The game should have a small file size, no more than 1MB. | The entire directory is under 1MB. |

*2. List the features were not implemented from Part 2 (table with ID and title).*

| Feature ID | Original Feature Description | Implementation Description |
|---|---|---|
| FR-1 | There shall be 4 options to choose from to customize each of the 3 attributes of the character. | Was not implemented. Would not add too much extra interest the game since it was switched from being graphical to command line. |
| FR-4 | When a user acquires 5 goblin friends, they will be awarded the title, “Goblin Master,” which will provide a boost to their health attribute. | Was not implemented - we realized the game scaled better if bonuses were not used, it became too easy too quickly otherwise. |
| FR-5 | When a user has defeated 10 evil goblins, they will be awarded the title, “Goblin Destroyer,” which will provide a boost to their skill attribute. | Was not implemented - we realized the game scaled better if bonuses were not used, it became too easy too quickly otherwise. |
| UR-1 | As a user I need to customize three attributes of my character so that I can have a personalized game experience. | Was not implemented. Would not add too much extra interest to the game since it was switched from graphical to command line based. |
| UR-5 | As a user I want to move my character around the map, avoiding pit obstacles as I explore, so that I can advance in the game. | There was a change in the implementation, instead of using a map we have directions that the user has the player explore. There are still pit obstacles (if the player goes SouthWest), but the mechanism for the world is slightly different. |
| NFR-1 | The game should be pleasing to the eye, with bright colors and simple shapes. | Was not implemented - the game was change to a command line game, in order to focus on the objects rather than graphics. |

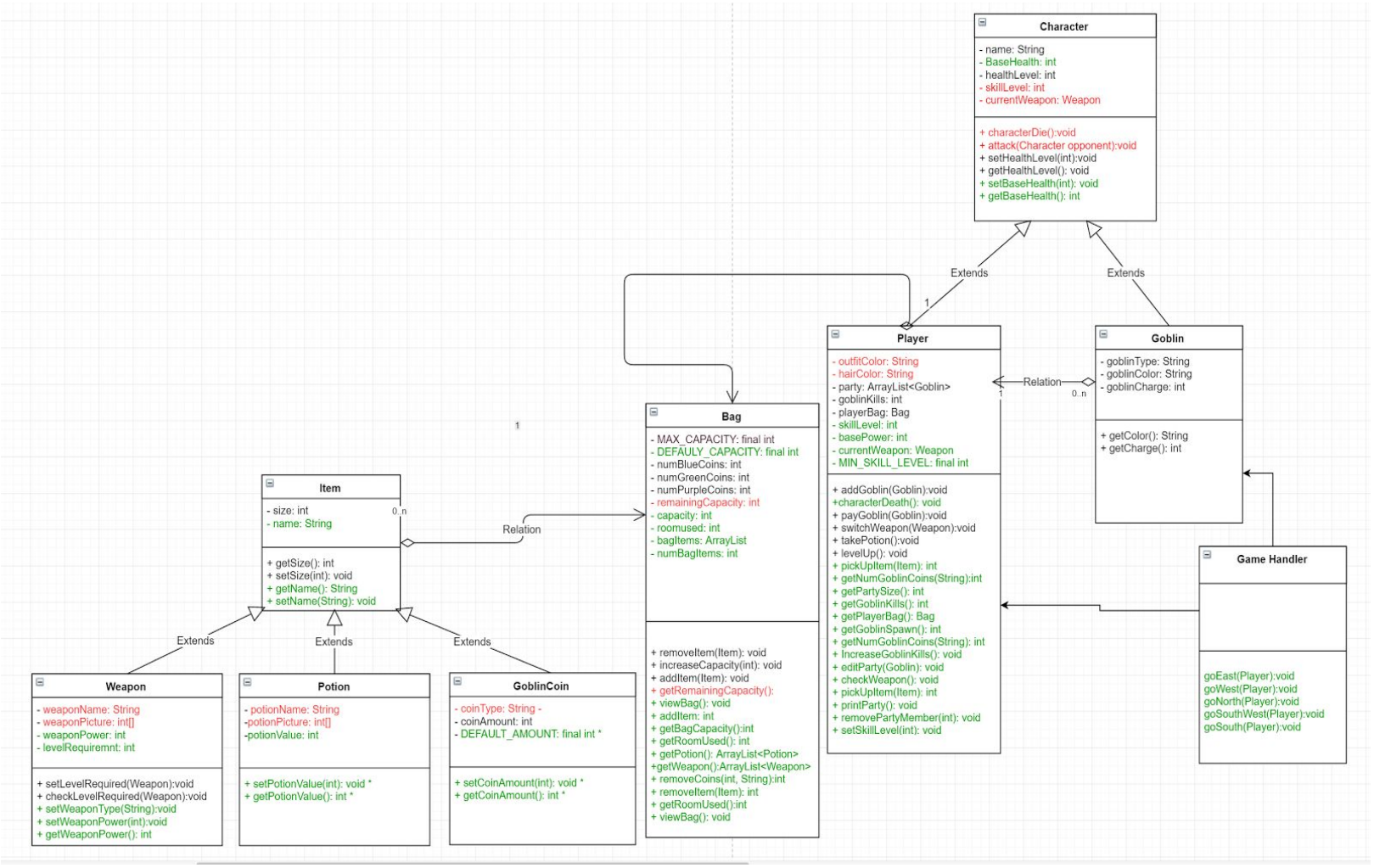
*3. Show your Part 2 class diagram and your final class diagram. What changed? Why? If it did not change much, then discuss how doing the design up front helped in the development.*

The class diagram did not change much in structure from the original to implementation. The classes that dealt with environment were, for the most part, not implemented. The aspect that changed a lot, if not the most, were the function definitions for each class. It was discovered that some functions redundant and could be consolidated by utilizing polymorphism, or that they were just not needed. Upon actual implementation we found that we needed some additional functions we had not planned for.




Old Class Diagram

Doing the class diagram up front helped us to figure out how our design all went together. That we did not produce a lot of code that we did not end up using is probably partially due to planning. We were able to somewhat figure out what made sense on paper before actually putting in all the work to produce code. This greatly helped the efficiency of the coding process. We realized quickly that making this game command line based instead of graphically based allowed us to focus on the object oriented principles.

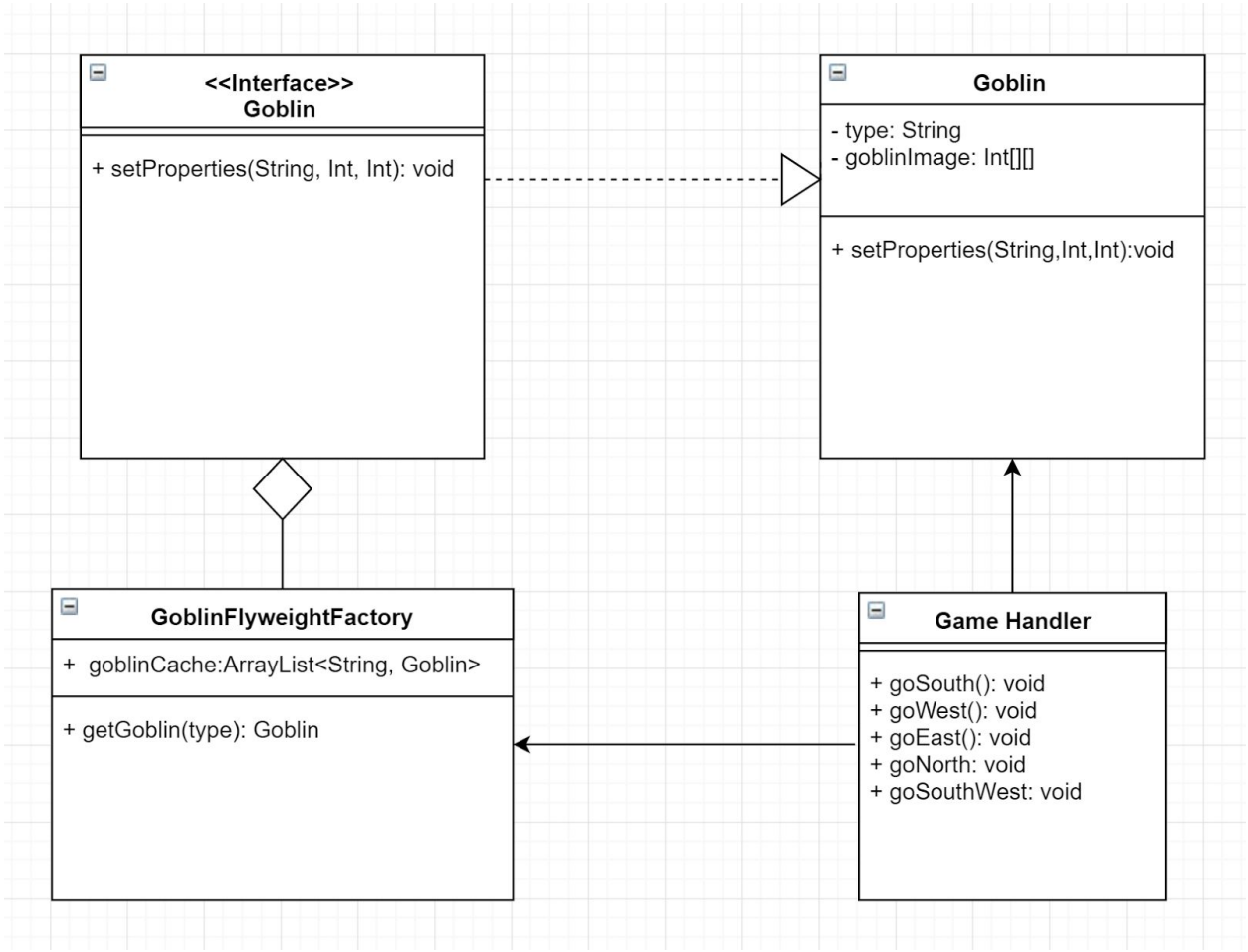



New Class Diagram

*4. Did you make use of any design patterns in the implementation of your final Prototype? If so, how? Show the classes from your class diagram that implement each design pattern (each design pattern as a separate image in the .PDF). If not, where could you make use of design patterns in your system? Show a class diagram of how you could implement each design pattern and compare how it would change from your current class diagram.*

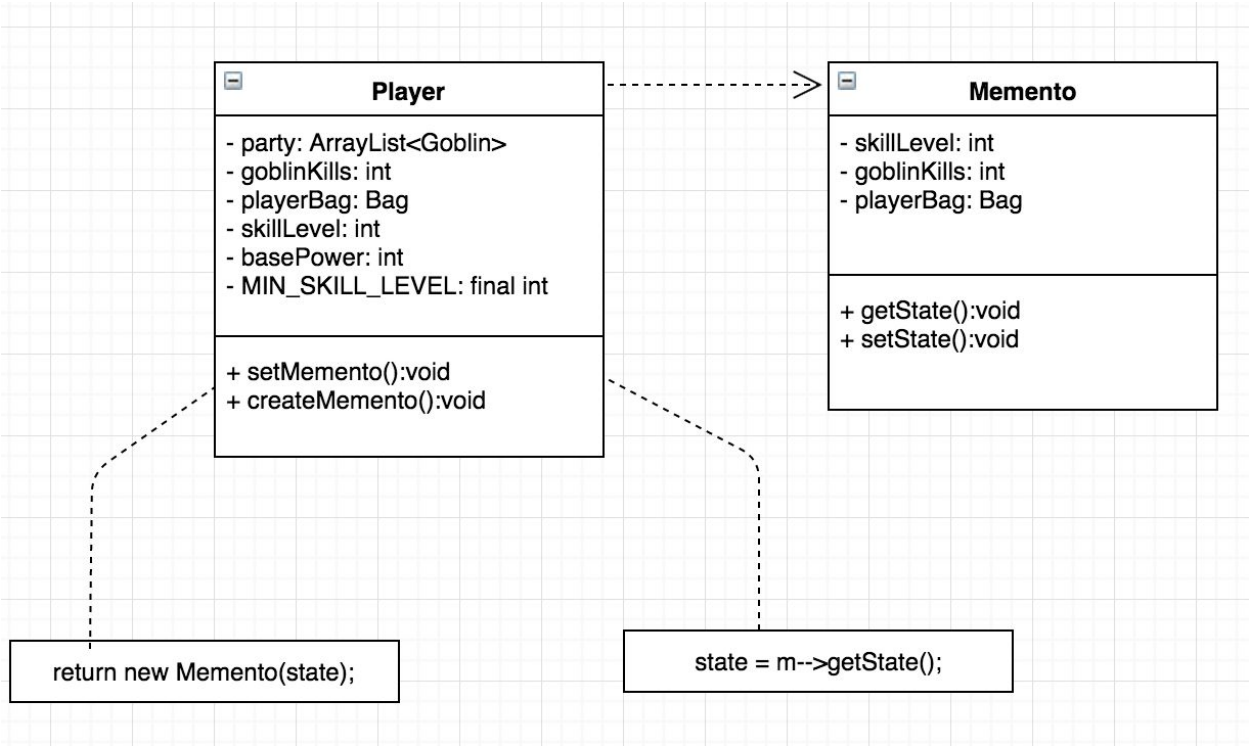
No we did not implement any design patterns in our project. We could have implemented the Flyweight design pattern to create and manipulate the Goblin objects. For this game, the Bad

Goblins have type “Bad” and good Goblins have type “Blue”, “Green”, or “Purple”. This property “type” is an intrinsic value of each instance of Goblin that is in the Goblin cache.
Objects of type Player, at any point in the game, will have some game level that they are at. Based off of this, and other factors possibly if the game was expanded, the Goblins extrinsic properties: health level and strength level can be set. The name of the Goblin is also an extrinsic property of the Goblin that can be set by the user to avoid repetition of names. Using this design pattern, we only ever have to create four Goblins for the entire life of the game and can reuse and redecorate them according to the specific scenario.




Flyweight Diagram

Another design pattern that could have been implemented was memento. Memento would allow us to externalize the internal state of objects, in this case the state of attributes of the player. Implementing this would allow us to restore the game to previous states, allowing users to continue on their quest with the same items in their Bag inventory, Weapon Inventory, health, skill level, and Goblin Kills. This would have been helpful if we scaled our game to being larger, but as written a single quest can be played satisfactorily in one sitting.




Memento Diagram

## *5. What have you learned about the process of analysis and design now that you have stepped through the process to create, design and implement a system?*

We have learned that planning ahead and carefully designing the system out before implementing can lead to more efficient implementation. Designing out a system properly in advance can be motivating to developers as a good design can give a clear path forward for them to work on. It also is helpful as it is somewhat a contract between developers if done right. The whole team can decide on an implementation where there is little to no miscommunication between developers because it is written out and connections are defined. Class diagrams are especially helpful for delegating work out to members of the team. People can clearly see what they need to implement and how it fits into the system.

In the absence of a dedicated project lead, or scrum master, it has been great to have dedicated documents we can all look at - both individually and as a team - to guide the process of developing our project. It has been helpful to be able to have agreed upon documents that are easily referenceable as representations of what we have been working towards.