Second Semester Programming Project:

Important links:

- Github repository: https://github.com/musman65/Project

Project Scenario:

My project operates in a video game. It runs a turn based video game and lets the user play in the output displaying all the information needed.

Design Paradigm:

- Turn by turn based style
- Different types of enemies, all with unique abilities and weaknesses
- Different classes for the player to chose
- Items
    - Weapons
    - Potions

Expected Results:

When the application is running the user can:

- Make a player and select a class
    - Wizard
    - Warrior
- View their stats and their available moves
- Battle opponents such (e.g. Goblin, Spectre)
    - The battle is turn based
        - The player choses a move each round
        - The enemy answers with a move based its strategy
- After each turn, the output displays:
    - What moves were used
    - The health points remaining of both combatants

- The battle is won or lost when either the player or the enemy reaches 0 health points

<u>Project Explanation:</u>

There are two super classes at the top and one regular class: (Italics : abstract class)

- *Player* class (general NPC class)
    - *Human* class (user class)
        - Warrior class
        - Wizard class
    - *Enemy* class
        - Goblin class
        - Spectre class
        - Mutated Wolf class
        - The Revenant King class (boss)
- *Item* class
    - Weapon class
    - Potion class
    - Armor class
- Move class

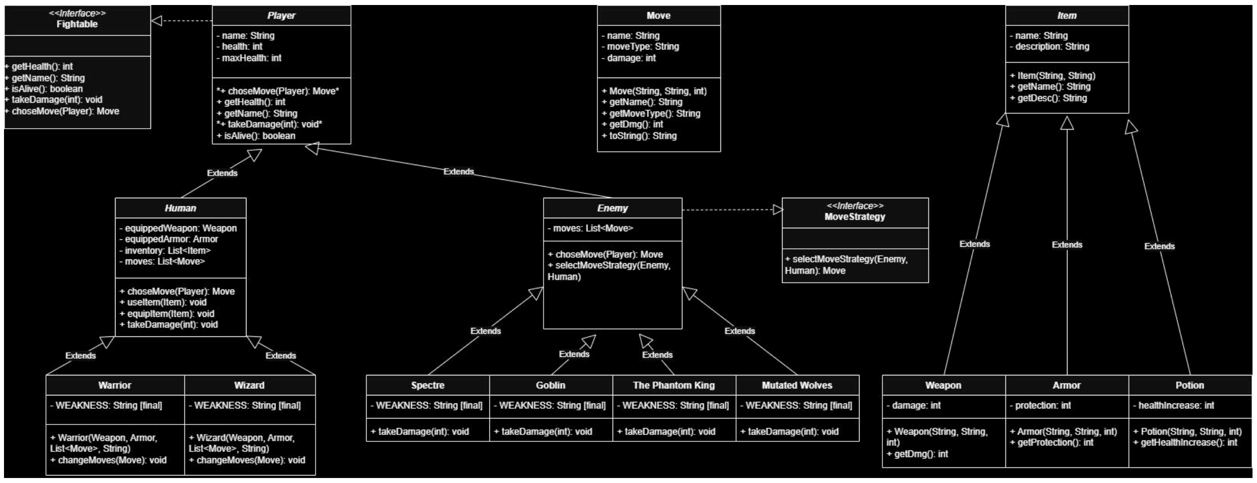
My program contains two interfaces. "Fightable" and "MoveStrategy". The first interface would make battles much easier to operate and manage. It would include methods that combatants need. This would also keep the main battle loop more organized. The second interface makes it so the enemies that are fighting the user don't only use one move but a variety of their moves depending on the situation (e.g. if they have less than a certain amount of health). All the methods involving strategy could be hidden behind the scene and I would only have to call one method that choses a move.

The method choseMove() would use polymorphism. Inside the human class, it would ask for the user input but in the enemy class, it would use predefined strategies depending on its situation.

The textIO implementation will be used in the Human class in order to save their progress. To save the progress, it will write all the information onto a .txt file and to import that save, it will read from the file to allow the person to continue their unfinished save.

There will be a Comparator implemented into the Item class to sort the items granted by defeating enemies to allow the user to sort them differently (e.g. by how much damage it does, the amount of armor it grants, etc). The Comparable will be implemented in the Enemy class to rank their difficulties by health.

Here is a class diagram for all the classes I will make:



For deliverable 2, the Player class, the enemy class and it’s subclasses (Spectre, Goblin, The Revenant King and the Mutated Wolf classes) will be finished.