Usman Muhammad

RPG_Project

420-SF2-RE DATA STRUCTURES AND OBJECT ORIENTED PROGRAMMING section

00001

<u>Table Of Contents:</u>

<u>Project Description:</u>

<u>Project Scenario:</u>

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.

<u>Design Paradigm:</u>

- 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

<u>Expected Results:</u>

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 Phantom 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.

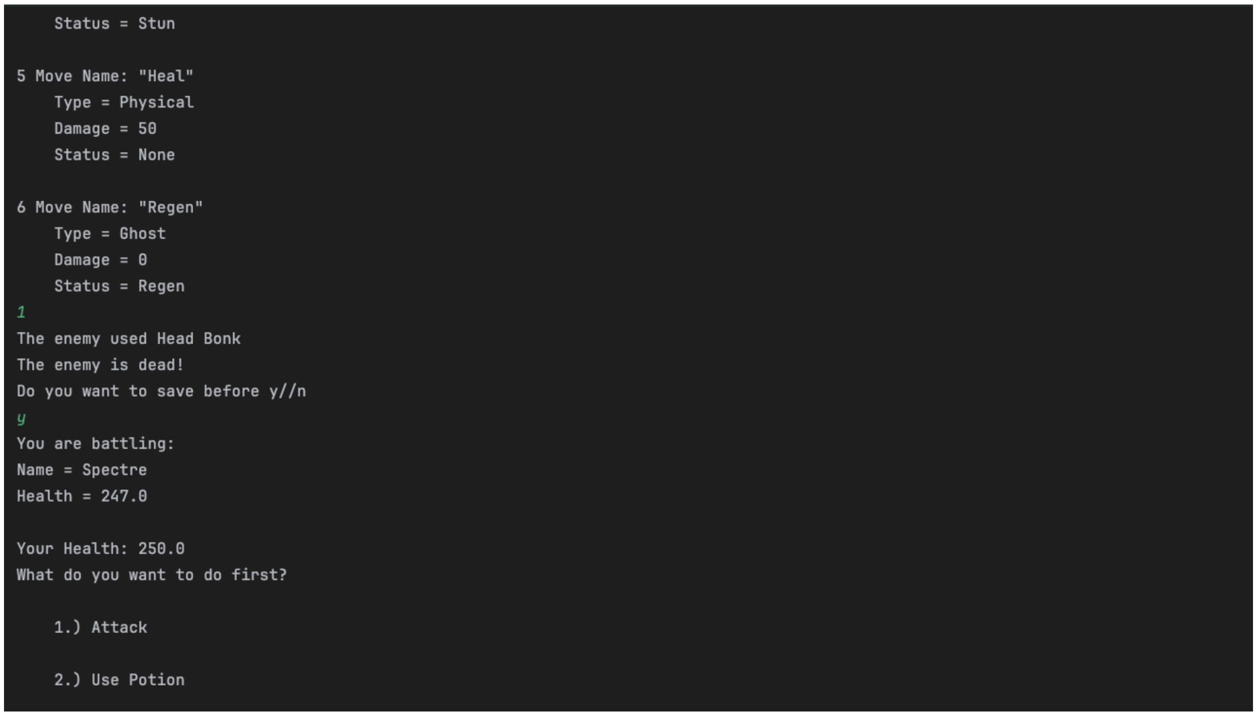
The method choseMove() would use polymorphism. Inside the human class, it would ask for the user input but in the enemy class, it would use predefined strategies depending on its situation.

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.

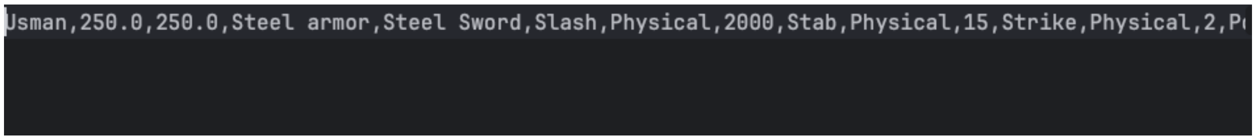
There will be a Comparator implemented into the Item class to sort the items granted by defeating enemies to allow the user to sort them differently (e.g. by how much damage it does, the amount of armor it grants, etc). The Comparable will be implemented in the Enemy class to rank their difficulties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

Program Features:

You can fight enemies with different movesets and built-in strategy in a turn based game. You can plan strategically your next move in order to prevail against your foes.
You are also able to save your progress after every battle!:

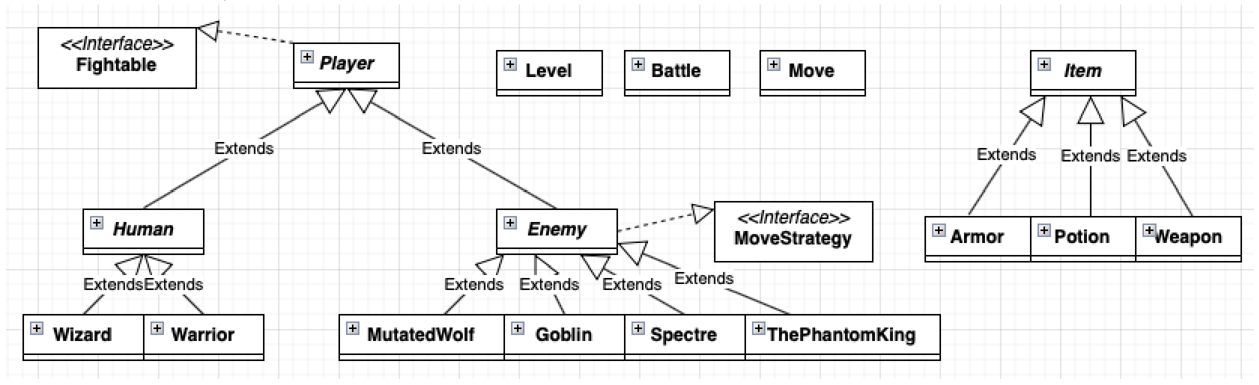


```
    Status = Stun

5 Move Name: "Heal"
    Type = Physical
    Damage = 50
    Status = None

6 Move Name: "Regen"
    Type = Ghost
    Damage = 0
    Status = Regen
1
The enemy used Head Bonk
The enemy is dead!
Do you want to save before y//n
y
You are battling:
Name = Spectre
Health = 247.0

Your Health: 250.0
What do you want to do first?

    1.) Attack

    2.) Use Potion
```



```
Usman,250.0,250.0,Steel armor,Steel Sword,Slash,Physical,2000,Stab,Physical,15,Strike,Physical,2,P
```

Here is also an updated version of my class diagram (only with the names of the classes/interfaces):



Challenges:

There were several challenges I faced during the creation of my project. The main one was the amount of time I thought it would take. I underestimated how much time I would take to finish the project. I also added way too much stuff after deliverable 2. I originally planned to only have a handful of mechanics in the battle but then I decided to add status effects which increased the workload much more. I had to take into account the user attacks and gets attacked and check for the status effects in both. I could have planned it better and maybe optimized it more. Because of this, I could not implement all the features that I wanted to, such as importing your save and I also left out some JavaDoc on some helper methods.

Learning Outcome:

I learned that I should not fly too close to the sun. I should not add any new features before having finished everything that is required and the barebones version of my project. In the next programming project that I will do, I will organize my planning in a much more efficient way. Even if I was not able to finish everything in my project I still enjoyed making it very much. It was more of a fun hobby to me than a chore. I also learned some valuable lessons (like the one I just mentioned) that will stick with me for a very long time. At the end of the day, isn't that what it's all about?