Name	Cole Paulik	Team	Spaghetti Studios	TL	3	Date	4/14/2025	Time	9:30

Fill in the underlined areas (and the boxes above), now but don't write on the remainder of this form.

Contribution: Briefly describe what your feature(s) is/are: **Items/Power ups for increasing the player's health and player speed. _____** Walk me through your Gantt chart. How long did this take? How long did you estimate it would take? What did you learn about your skill as an estimator? At the time of writing this it has taken me 28 hours out of my projected 48 hours. When estimating these I thought I was underestimating how much time I was giving myself and now that I have done the project and am at the end I realized that I still had a bit of time leftover. Run your game and point out places where your code is called and run. (I will cycle through asking you this question and the next one until you either run out of interesting things to talk about or it is clear that you have made an above average contribution.) Show the C++/C# code that was run. Walk me through the methods called from the time it enters your section of code.	/10
Technical: Walk me through your test plan. Give an example where a test case later found a bug in your code by things a teammate added later. (Or explain why you chose a test case specifically because you wanted to ensure that a teammate would know if they broke your code.) _____ Pick a Prefab you have created that is documented well in a separate readme file. (I will point to several places in your code documentation and ask) What question where you trying to answer here? Who do you anticipate would be asking that question? What other questions might this person need the answers to? Prefab Name: HealingItem_____ Show me a class in your code where there could be either static or dynamic binding. Write some mock code on this paper showing how you would set the static type and dynamic type of a variable. Super Class: IPowerUpEffect_____ Sub Class: BasicHealthBoost_____ Virtual Function: Apply(GameObject player)_____	/4 /3 /3

Choose a dynamically bound method. What method gets called now?
Change the dynamic type. What method gets called now?
Pick a statically bound method. Which one would be called in each of the two previous cases?

```
public class PowerUp : MonoBehaviour
{
    public virtual void ApplyEffect(GameObject player)
    {
        Debug.Log("PowerUp applied to " + player.name);
    }
}


    public override void ApplyEffect(GameObject player)
    {
        effect.Apply(player);
        Destroy(gameObject);
    }
```

/4

Show me an example of reuse in your code where you violate copyright law.
How does it violate copyright? Regeneration Potion from Minecraft__
What did you have to do to integrate it with the code you wrote? What are the legal implications if you market your code with the re-used portion? Use fair use argue that you can use this anyway.

/4

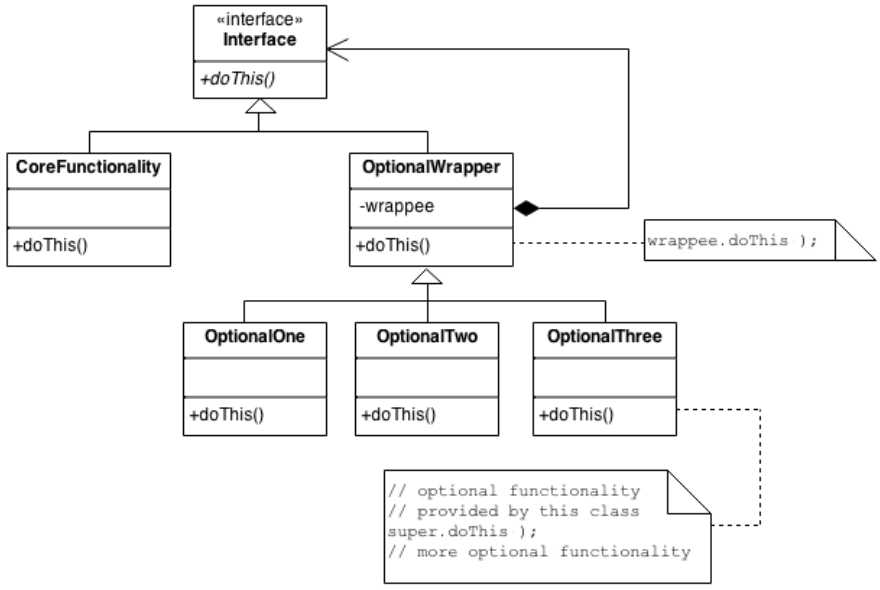
I used the regeneration potion from Minecraft as the sprite for my regeneration power up. If I was a prosecutor, I would argue that this potion is one of the most famous ones in the game and that it is a game mechanic that has been around and is easily recognizable. As a defense attorney I don't deem the potion to be an integral part of the game and there are other potions in the game that look extremely similar. I don't think it would hurt the potential marketing for Minecraft so I think I could argue that I have the write to use it.

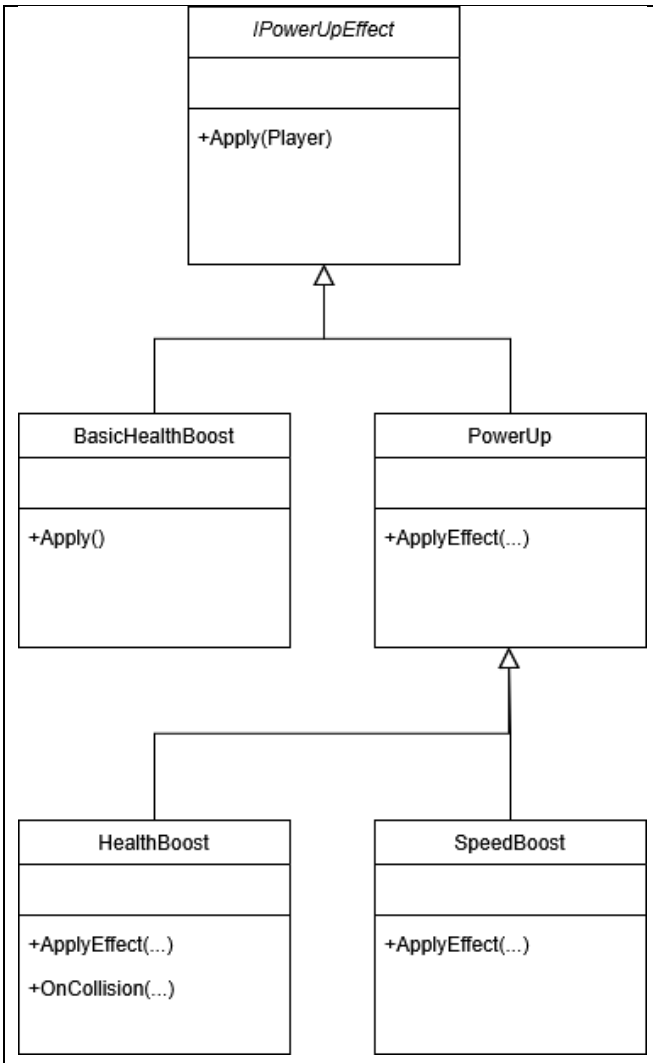
4. One big or two small, well-chosen patterns.
Small Patterns = {Singleton, Private Class Data}
Which patterns did you choose?
1. Decorator

2.

Why did you choose each pattern? (Justify your use of it).
I chose this pattern because it is good for power-ups because it allows adding new behaviors or functionalities to objects dynamically.

Draw the class diagram for your pattern(s).





Would something else have worked as well or better than this pattern? When would be a bad time to use this pattern?

A Factory patter could also work because it allows the creation of objects without exposing instantiation logic to the client. It would be bad to use this pattern with few power-ups, which is what I have because I only have a handful of power-ups created, which would be overkill.