Name	Ben Kopf	Team	Spaghetti Studios	TL	5	Date		Time	

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: I worked on the game's enemies and the demo mode for the game. This includes sprites and movement for the enemies and handling damage being delt to them. **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?** Most of my estimates were much longer than it actually took me to complete. I am aware that I am a poor estimator of time, so I generally try to add X amount of hours to my estimated time, that way I know I am at least not underestimating the time it will take. Though at times it still isn't enough. **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.)** For my test I conducted a lot of boundary tests, ensuring each enemy type took damage, and died as they were supposed to. I also checked enemy speed to make sure it couldn't rise above a certain value. For my stress test I tested how many enemies I could spawn before we dropped below 30fps, and it was around 300, which is more enemies than should ever be in one room. These tests helped me find bugs that got introduced once the player started damaging the enemy, I was noticing some weird issues where the enemy would take multiple instances of damage, I was able to root out the cause and fix the issue. It also led to me optimizing my enemy death triggers better and making a more robust system. **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:** Enemy	/4 /3 /3

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: EnemyHandlerBC
Sub Class: EnemyHandler
Virtual Function: SetHealth()
Choose a dynamically bound method. What method gets called now? /4
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?
Show me an example of reuse in your code where you violate copyright law.
How does it violate copyright?
I used the Goomba from the Mario games as an enemy sprite
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.
I had to find the sprite online and put it into my game, I was able to replace the red square I had previously as the sprite. To argue fair use, since the goomba isn't the main character of the Mario franchise and not a widely advertised character of our game it won't be taking revenue from the Mario franchise. /4

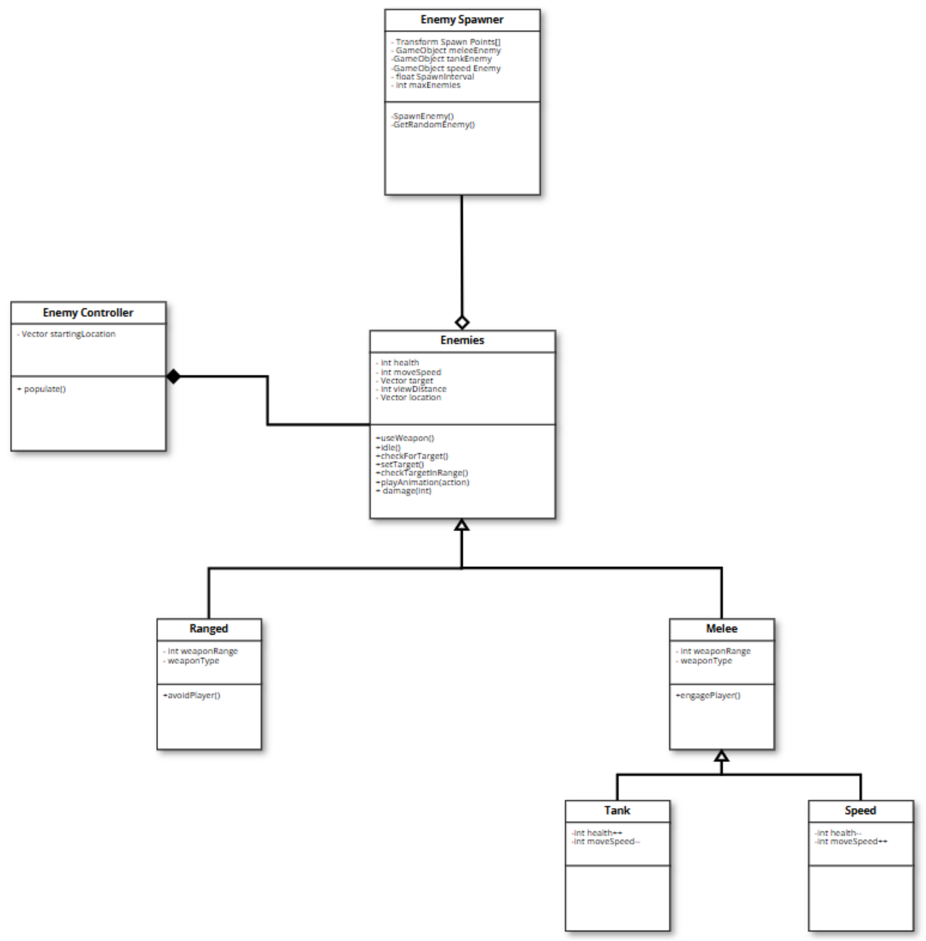
4. One big or two small, well-chosen patterns.
Small Patterns = {Singleton, Private Class Data}
Which patterns did you choose?
1.Factory

2.Decorator

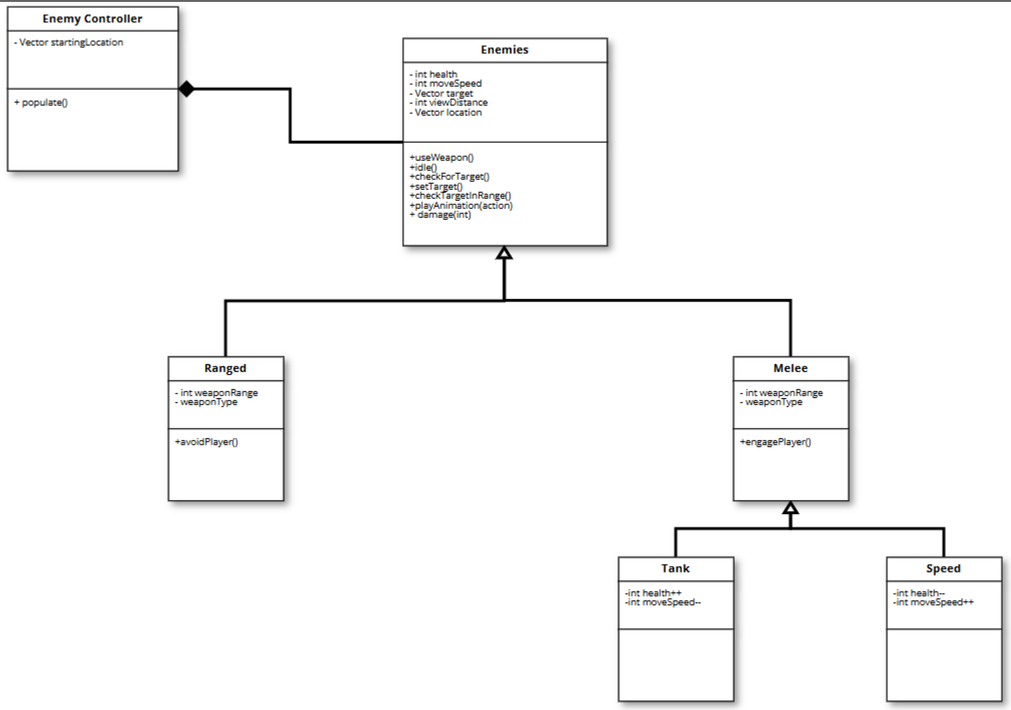
Why did you choose each pattern? (Justify your use of it).
I chose the factory because it provides me a way to spawn my enemies across rooms without individually placing each one and giving some variance in runs.
I chose to use the Decorator pattern because each of my enemies was like a slightly modified version of the base class of Enemy. So, the Tank enemy is like a regular enemy wearing a melee enemy hat with a tank shield. Sort of like accessories.

Draw the class diagram for your pattern(s).

Factory:


Enemy Spawner
- Transform Spawn Points[]
- GameObject meleeEnemy
-GameObject tankEnemy
-GameObject speed Enemy
- float SpawnInterval
- int maxEnemies
-SpawnEnemy()
-GetRandomEnemy()
Enemy Controller
- Vector startingLocation
+ populate()
Enemies
- int health
- int moveSpeed
- Vector target
- int viewDistance
- Vector location
+useWeapon()
+idle()
+checkForTarget()
+setTarget()
+checkTargetInRange()
+playAnimation(action)
+ damage(int)
Ranged
- int weaponRange
- weaponType
+avoidPlayer()
Melee
- int weaponRange
- weaponType
+engagePlayer()
Tank
-int health++
-int moveSpeed--
Speed
-int health--
-int moveSpeed++

Decorator:



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

I think the decorator pattern works really well for what we need, but another candidate would have been the template method. I think I would've aimed to do that one had my code and class diagram not already resembled the decorator pattern.

I wouldn't want to use the decorator pattern if I needed more customizability over enemy's properties. Like if I wanted a speed enemy but I didn't want to make another class, I wouldn't be able to do that with this pattern.

I like the factory method for spawning enemies a lot because it gives me the flexibility of random spawns, but I don't have such a vast array of enemies or classes of enemies that an abstract factory would be necessary. I wouldn't want to use this pattern when I want to have a very wide variety of things to spawn.