Final Exam Explanation

Saad Khan - 100829159 (Prime number: 23)

Game: Mega Man 6
Scene: Blizzard Man Fight

Implementation

Command

Unfortunately, I couldn't get it to function properly in time, however, the theory was that the command design pattern was used to keep track of player inputs (i.e. moving left, moving right, jumping). To get it working with the input, you'd need an invoker that acts as a middle man and will keep track of the commands, the input handler would then review the commands and run specific methods depending on which command you've invoked. I chose to use commands for inputs because it's a good way to encapsulate the player's movement into objects in the event you want to undo them.

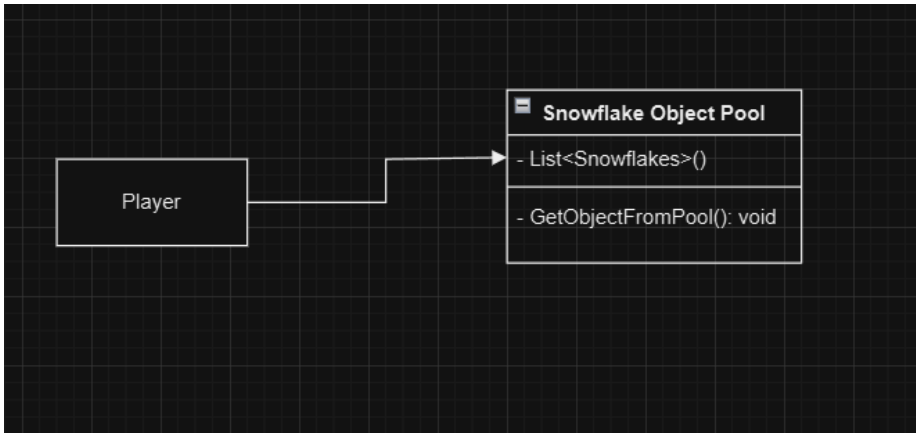
Singleton

Unfortunately, the singleton doesn't function, however, I would use it to record the player's health. It's a simple yet effective singular implementation that can work with the UI system to display the health of Mega Man. In tandem with an observer, I can read the value and display it in text on the UI's canvas.

Object Pooling

Unfortunately, Object Pooling wasn't complete in time, however, I can still discuss what I would've wanted to happen. As the instructions suggest, I would use it to manage the snowflakes on the scene. When they touch the ground or hit Mega Man, rather than destroying them, I can simply deactivate them and add them to the pool. Once they instantiate again, I can dequeue them from the pool. It's a good way to maintain the amount of snowflakes on screen and can help with performance issues. Speaking of performance issues, if I was able to implement the object pool in time, I could use the profiler to see the performance difference between the pooled spawner and an unoptimized pool spawner. Depending on the amount of snowflakes I'm spawning in, the performance gain will increase accordingly.

Diagram



This is the diagram for the supposed Object Pool for the snowflake.