Name: Hujoe Pandi Selvan

Class: CPE 203

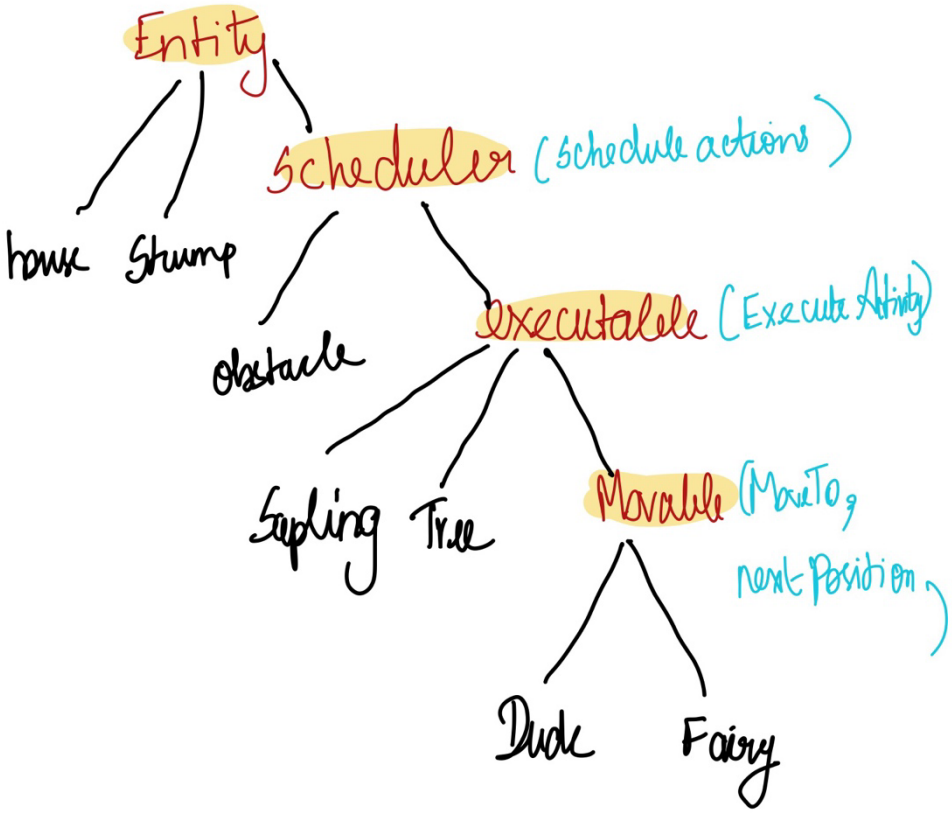
In Project 2, I initially used an entity as an interface, but I later changed it into an abstract method that sits at the highest order in the hierarchy. This change allowed me to provide a clear branching off point for the House and Stump classes, as well as the Scheduler abstract class.

I created the Scheduler abstract class to hold schedule actions that are commonly found in Obstacle and other following classes. By doing this, I was able to create a more modular codebase that was easier to manage and maintain.

To further enhance modularity, I derived an Executable abstract class from Scheduler, which in turn is being extended to the Sapling, Tree, and Movable abstract classes. The Executable abstract class contains an execute activity that is common in the following classes. By encapsulating this activity in the Executable abstract class, I was able to create a more concise and less repetitive codebase.

Finally, I created the Movable abstract class, which contains the moveTo and next position functions that are then extended to the Dudenotfull, Dudefull, and Fairy classes. This abstraction ensures that my code is more efficient and that each class is focused on its core functionality.

Overall, the hierarchy I created follows a clear and logical structure that enhances modularity and code reusability. By breaking down the code into smaller, more manageable classes, I can easily add or remove features without having to make significant changes to the overall structure. I have attached the hierarchical idea I followed bellow.


Entity
Scheduler
(schedule actions)
house
Stump
Obstacle
Executable
(Execute Activity)
Sapling
Tree
Movable
(MoveTo, nextPosition)
Dude
Fairy