Design Rationale

REQ3:

Crater

The crater can spawn HuntsmanSpider, so it should implement Spawner interface. But with considering there might be other spawners in the future that can spawn NPC, another layer of abstraction is added, which is the abstract class NpcSpawner. This class enables NPC spawner to perform certain operations to initialize and spawn NPC. This abstract class will implement Spawner interface and Crater will extend this abstract class instead. This makes adding more NPC spawner class easier in the future and dependency inversion can be achieved as client classes can communicate with the concrete classes of NpcSpawner through the abstract class. The downside of this implementation is that in the constructor of the concrete class, we need to set the NPC to be spawn using a setter, this is a dependency injection via setter, which prompt the possibility of the abstract class not properly initialised until the NPC has been set through the setter.

Huntsman Spider Wander Behaviour and Attack Behaviour

For the NPC behaviour, I categorized them into 2 different categories, 1 is for hostile NPCs and another 1 is for docile NPCs. Thus, I make WanderBehaviour and AttackBehaviour the abstract classes, and the hostile behaviours of the respecting behaviour will extend these abstract classes. So, HuntsmanSpider will have HostileNPCAttackBehaviour and HostileNPCWanderBehaviour. This implementation allows us to implement NPC behaviour based on their nature. The cons of this is if there are differences in behaviour among hostile/docile NPCs, another behaviour class will need to be implemented and extend the existing behaviour class, this could result in too many levels of extension created.