REQ4

Design Goals:

1. Implement a player
2. Implement combat archetypes of the player
3. Implement weapons and corresponding actions for each archetypes

To achieve design goal 1:

- We made the Player class abstract since the user must choose one of the archetypes available when playing the game
- The Player inherits Actor class. They share most of the functionalities, but Player requires Menu object to accept user input for action choice.
- The player is classified by an enumerable ActorStatus. It always has “hostile_to_enemy” status which can be used when using alloableActions of other actors.

To achieve design goal 2:

- We made a class for each archetype, Samurai, Bandit, and Wretch. They inherit the Player abstract class and have different constructors to instantiate their corresponding weapons, health, other attributes.
- Although each archetype player can only have one weapon initially, they can pick up and buy weapons from the trader. This is why the multiplicity is 0..” to 1..*.
- Since we need to apply different actions for the player depending on weapons available, each player type comes with WeaponStatus.

To achieve design goal 3:

- We made a class for each weapon: Club, GreatKnife, Uchigatana. This is because they all require different implementation.
- Each of them inherit WeaponItem class. At this point, there will not be any extra methods in these classes, but the attributes (such as damage, hitRate, verb, etc) will be initialized with a constant value inside the constructor, to avoid using literals many times.
- For each action that player can make using these weapons, we made a class: AttackAction, UnsheathAction, QuickstepAction. They all inherit the Action abstract class. Since, QuickstepAction can be considered as AttackAction + MoveActor, it has dependency with both of them. This ensures that any actions related to weapons will be executed through Action class, which is consistent with other parts of the system.
- Although not shown in the diagram, these actions will be instantiated in each enemy’s allowableActions method. In the allowableActions(), we use the WeaponStatus to classify a given player type and instantiate action objects (described above) corresponding to that class. Hence,

REQ5

Design Goal:

1. Splitting the map into West and East
2. To implement more enemies required
3. To implement more weapons

To achieve design goal 1:

- The relationship for the Environment class is the same as REQ1 (with the same environments) but this class further shows dependency with the Locations interface which contains West and East.
- Enemies that spawn from the West side are Heavy Skeletal Swordsman, Lone Wolf and Giant Crab, and enemies from the East side are Skeletal Bandit Giant Dog and Giant Crayfish and hence dependency is shown.

To achieve design goal 2:

- Using the abstract class Actor from engine, we extend an abstract class Enemy, which further extends to three abstract classes named Ocean, Wind and Skeleton.
    - Ocean extends two concrete classes Giant Crayfish and Giant Crab.
    - Wind extends two concrete classes Giant Dog and Lone Wolf.
    - Skeleton extends two concrete classes Heavy Skeletal Swordsman and Skeletal Bandit.
- Heavy Skeletal Swordsman, Lone Wolf and Giant Crab spawn from Graveyard, Gust Of Wind and Puddle Of Water respectively on the West side of the map whereas Skeletal Bandit, Giant Dog and Giant Crayfish spawn from Graveyard, Gust Of Wind and Puddle Of Water respectively on the East side of the map and hence shows dependency. These enemies can also Despawn (implemented in the action class).
- They each also have certain behaviours, same as REQ1, the only difference being, that the Heavy Skeletal Swordsman and Skeletal Bandit could use spinning attack as a behaviour and Giant Crab and Giant Crayfish could use slam as a behaviour.
- Each of these enemies can attack other enemies and players, except Skeletal Bandit (when using the SpinningAttack with the Scimitar as seen below) and Giant Crayfish (when using Slam), which uses AreaAttackAction (damaging everyone in the area).
- The Skeletal bandit has similar as the Heavy Skeletal Swordsman which has a special ability where when it is dead (using DeathAction), it turns into a Bone Pile that can be turned back into a Heavy Skeletal Swordsman (by using the action classes TurnIntoHeavySkeletalSwordsman and TurnIntoPileOfBones).
- DeathAction is also used when a player/enemy dies/is killed.

To achieve goal 3:

- A new weapon name Scimitar is added that extends from the abstract class WeaponItem from engine. This is used by the Skeletal Bandit.
- This weapon uses AreaAttackAction to inflict damage on anyone in its surroundings, only when SpinningAttack is used (implemented in behaviours class)
- The Scimitar can be purchased and sold (using Sell and Purchase from the abstract class Action) using RuneManager (that uses Runes), whereas the Grossmesser can only be sold not purchased.