Recommendation of engine code

Throughout the assignments that we have done, discussion about the engine package was great and understandable, hence, we are satisfied with the engine package and does not require any modification for it as it has met the design principles that we have learned in this unit. Why do we think that the engine package is well done? In the engine package, methods from each class are properly commented, hence it makes it easier for us to extends from it and modify according to the requirement of the assignment. Below, we will be discussing about a few design principles that we have learned throughout this semester.

First and foremost, a good example of design principle is "Keep It Short and Simple (KISS)", where the codes that are written in the engine package can easily be understand as everything is well organised. Extending or implementing classes that is found from the engine package can easily be done, thumbs up for the functionality description has been well written by keeping each code short and simple.

Secondly, we will be explaining two of the SOLID principle that has been applied in this engine code. The two principles are Single Responsibility Principle (SRP) and Open/Closed Principle (OCP).

Single Responsibility Principle (SRP) tells us that each class should only have one responsibility and not overloaded with more than that. This principle can ensure that the code is cleaner and more organised, also easier to maintain the system. By giving an example, PickUpItemAction, DropItemAction is not implemented in the Actor class as picking up item / dropping an item is the responsibility of the PickUpItemAction / DropItemAction but not the Actor class's responsibility to implement or validate it.

Open/Closed Principle (OCP) allows us to be able to add features to our game package. This makes it easier for us to maintain our codes and does not break it. By taking an example of Action class, we are able to extends our application well just by extending the Action class from the engine package and implement it according to the requirements that are needed for the assignment (e.g. ChantingAction, CraftingAction and many more). Through extending the Action class, we are able to build or add new functionality for the player to have various actions. This principle also ensures backwards compatibility and prevent regressions.