Design Rationale for Requirement 4

The diagram represents an object-oriented system for player classes, the respective starting weapons, and their special skills.

The three player Classes(capitalised to reduce confusion), Wretch, Bandit, and Samurai are represented as classes of the same name. The Player class has been changed to an abstract class to allow for these Class classes to extend and implement.

The three associated weapons of each class, Club, GreatKnife and Uchigatana are also represented as their own classes Club, GreatKnife and Uchigatana respectively. They all extend the WeaponItem abstract class, and have dependency from the Class to the weapon to represent that each Class starts with their respective weapon.

The Great Knife weapon’s Quickstep skill is represented as its own class, QuickStepAction, which extends the AttackAction abstract class, as it is a special attack. The dependency from the weapon to QuickStepAction shows that the Great Knife weapons allows its wielder to use QuickStepAction. QuickStepAction has a dependency between Location in engine to allow player to check the surrounding in map to move.

There is a class call UnsheatheWeapon which is extend the WeaponItem abstract class. It is a weapon for Uchigatana special attack(UnsheatheAction). There are dependency between UnsheatheWeapon and UnsheatheAction shows that the UnsheatheAction class will use the UnsheatheWeapon.

The use of abstract classes to represent the player Classes and weapons adheres to the Open-closed Principle, by not modifying existing code. It allows for new Classes and weapons to be added in the future with minimal risk. It also adheres to the DRY principle by reusing code through inheritance.

A downside to this approach is that the code and logic for player Classes and weapons is spread across multiple classes rather than in one place, which can make it harder to understand. It also results in larger code.

Changes in Assignment 2

The relationships between GreatKnife and QuickstepAction, and Uchigatana and UnsheatheAction, are now dependencies.

UnsheatheAction now creates a new instance of UnsheatheWeapon which is used as a temporary weapon to pass into the AttackAction, to achieve the x2 damage and 60% accuracy modifiers.