REQ 4

A. Classes Combat Archetypes

“PlayerRoleFactory” is a new interface we have made in requirement 3 to make the classes classes be used by allies and invaders. This allows us to easily create or give classes to our respective actors. (We have removed the extension from respective class classes to accommodate this new feature.)

Four classes are added to represent three different starting classes; “Samurai”, “Wretch”, “Bandit” and “classes”. The classes will be surrounded by a Class package for better organisation.

These classes implement the “PlayerRoleFactory” class therefore implying that we have successfully implemented the Don’t Repeat Yourself (DRY) rule. We can easily extend the amount of classes by just having a new class for said class to implement PlayerRoleFactory to adopt its methods.

The User will select which class they would like to be at the start of the game via the menu in the console. This is done by the “ChooseClassMenu”. Depending on which class the user picks the menu will invoke the “chosePlayerRole” override method that resides within the respective classes. This means that the “ChooseClassMenu” will have a dependency to the “PlayerRoleFactory”. This way of implementation follows the Single Role Principle (SRP) as we have a separate class for choosing roles using a menu. The “ChooseClassMenu” is located in the Utils class.

The downside of this approach is that we have to manually create new classes and it is not automated. This can be proven tedious and problematic as it will be hard to keep track of, for example, 1000 classes.

B. Weapons

Three Weapons, the “Uchigatana”, “Club” and “GreatKnife” are added to the “Weapon” package for better organisation. These weapon classes are depended on by their respective classes;

a. “Samurai” and “Astrologer” have a dependency to “Uchigatana”.
b. “Wretch” has a dependency to “Club”.
c. “Bandit” has a dependency to “GreatKnife”.

Each weapon extends the “WeaponItem” abstract class, so that they can share common methods (DRY rule) .

The “Uchigatana” has a special skill known as Unsheathe. Thus we have created an “UnsheatheAction” class in the “Action” package. The “Uchigatana” class has a dependency to the “UnsheatheAction” class. While the “UnsheatheAction” class extends the “Action” abstract class allowing it to share their common methods (DRY rule).

The “UnsheatheAction” class allows the user to do double damage of the weapon that wields the skill at a 60% hit rate. This is accomplished by having an inheritance relationship with “Actor”, to know the target we are attacking, the “WeaponItem”, So we know which weapon is using the skill, and a dependency relationship to “RandomNumberGenerator” so we know if the attack has missed or not.

The “GreatKnife” has a special skill known as Quick Step. Thus we have create a “QuickStepAction” class in the “Action” package. The “GreatKnife” class has a dependency to the “QuickStepAction” class. While the “QuickStepAction” class extends the “Action” abstract class allowing it to share their common methods (DRY rule).

The “QuickStepAction” class allows the user to do the same damage as the weapon that wields the skill at the same rate. However, It also allows the wielder to immediately move to an adjacent location right after the attack. This is accomplished by having an inheritance relationship with “Actor”, to know the target we are attacking and the player we are moving, the “WeaponItem”, so we know which item will be using the skill, the “Location”, so we know where the player is currently at, the “Exit”, so we know the possible exits that the player may move and if there is an actor in an adjacent location, and finally a dependency relationship to “RandomNumberGenerator”, so we know if the attack has missed or not. (Note: the player will move regardless if the attack has hit)

The downside of this approach is that we have to manually create new Weapons and skills is not automated. This can be proven tedious and problematic as it will be hard to keep track of, for example, 1000 weapons with 1000 different skills.