Design Rationale for Requirement 4

GreatKnife & GiantHammer

The initial implementations of the GreatKnife and GiantHammer worked similarly with the Broadsword in Assignment 1, implementing the AbleToActivateSkill interface. However, this showed a code smell in which this interface handled too many skills in a single declaration. Thus, to approach this problem, an application of the Interface Segregation Principle was used to separate these declarations into smaller interfaces, namely FocusCapable, StabAndStepCapable, and GreatSlamCapable, to ensure the weapon items only implement the skills they can use. For example, GreatKnife and GiantHammer implemented StabAndStepCapable and GreatSlamCapable, respectively, and within the interface, there was a method returning the relevant Action to take place. This also satisfied the Open/Closed Principle, as new weapon items that use similar skills can also implement the created skill interfaces.

The allowableActions method allows the item's owner to perform an Action on another actor if the actor is hostile to the enemy. This method will add the relevant action list to the actions list, which the Actor can perform.

Instead of overriding the allowableActions in the Enemy class, created in Assignment 1, to create the AttackAction instances, this action had been moved to individual WeaponItem concrete classes. This occurs in the allowableActions method in the respective classes. This removed downcasting to WeaponItem when looping through the player's inventory.

The allowableActions method of GreatKnife was chosen to be visualised in the sequential diagram, displaying the new ability to add an AttackAction, StabAndStepAction, and SellAction to the actions list as it is the most complex function.

StabAndStepAction & GreatSlamAction

These classes extend from the abstract Action class and accept an Actor, WeaponItem, Location, and staminaDecreasePercentage integer instances as parameters for their constructors. The passed-in Actor was renamed 'target' to differentiate from the actor acting.

Moreover, these action classes also implement the StaminaConsumable interface. This indicates the skills could consume the actor's stamina when being activated. The use of an abstract class, StaminaConsumableAction, instead of an interface, although it would remove repetition and align with the DRY principle, would still violate the Liskov Substitution and Dependency Inversion Principle and, hence, was not implemented since any classes extending from this would be too abstracted. By introducing an interface instead, only the actions that needed to consume stamina to activate would need to be implemented and would meet both of these principles.

Furthermore, the execute method for the StabAndStepAction first checks whether the Actor has enough stamina to trigger the skill. If yes, the actor attacks and moves away to the first

available exit; however, there remains a chance that the Actor misses while attacking the target.

On the other hand, the execute method of the GreatSlamAction checks whether the Actor has enough stamina to trigger the skill. If not, the skills wouldn't be activated. Following that, the target would be hit at a damage multiplier of 100%, and both the actor and the surrounding enemies would be hit at a damage multiplier of 50%. However, there is a chance that the actor misses the target due to the weapon's chance-to-hit index.