Design Rationale for Requirement 3

The diagram represents an object-oriented system for the Flask of Crimson Tears item, Site of Lost Grace ground, Game Reset feature in the game.

The Flask of Crimson Tears item is represented by its own class, FlaskOfCrimsonTears, which extends the Item abstract class. Because the player starts with it, it has a 1 to 1 association relationship from Player to FlaskOfCrimsonTears.

The Site of Lost Grace ground is represented by its own class SiteOfLostGrace, which extends the Ground abstract class. It allows the player to rest on it, represented by the RestAction class, which extends the Action abstract class.

The Game Reset feature is implemented using the given ResetManager class and Resettable interface. Classes which are “resettable” implement the interface. This includes FlaskOfCrimsonTears, Player, and Enemy. The ResetManager handles resetting all the objects which implement the Resettable interface, represented by the 1 to many association between ResetManager and Resettable. It resets the game when the player performs the Rest action at a Site of Lost Grace ground, represented by the 1 to 1 association between ResetManager and RestAction.

When the player dies, their runes are dropped, represented as a DroppedRunes class, which extends the Actor abstract class. The player can recover the Runes, an action represented by the RecoverRunesAction class, which extends the Action abstract class. RecoverRunesAction uses the RuneManager class to update the player’s rune count.

The use of abstract classes to represent the Flask of Crimson Tears item, Site of Lost Grace ground, Dropped Runes actor, and the two Rest and RecoverRunes actions adheres to the Open-closed Principle, by not modifying existing code. It also adheres to the DRY principle by reusing code through inheritance.

Adding functionality for classes to reset their state whenever the game resets through the use of the Resettable interface adheres to the Dependency Inversion principle, by not having concrete classes depend on each other, and using an abstract interface for all “resettable” classes.

A slight disadvantage to the approach of creating and implementing the interface is that more code has to be written. It can also cause issues if implemented poorly, forcing subclasses to implement unneeded methods, if the Interface Segregation Principle is not followed.

Changes in Assignment 2

Player has a dependency instead of a 1 to 1 association with FlaskOfCrimsonTears, as the private attribute was not necessary.

Using the Flask of Crimson Tears is implemented as the DrinkFlaskAction action, which has a 1 to 1 association with the flask it is tied to.

The “The First Step” Site of Lost Grace is implemented as its own class, extending the SiteOfLostGrace abstract class, adhering to the DRY principle through reusing code by inheritance. An advantage to this approach is that it allows future Sites of Lost Grace to be easily implemented. One disadvantage is that the code and logic

The ResetManager class uses a singleton constructor, ensuring only one instance of the class ever exists at once. The dependency relationships represent the various resettable classes which add themselves to the list of resettable objects, as well as actually causing the game to reset (GraceResetAction, DeathAction). This adheres to the Single Responsibility principle, by having the ResetManager class be responsible for resetting all resettable objects. An advantage of this approach is that the ResetManager instance can be accessed by any other class. A disadvantage is that the implementation (the singleton constructor) is more complex, and can cause problems if done incorrectly.

Similarly, the RuneManager class also uses a singleton constructor to ensure a single instance. It keeps track of the instance of DroppedRunes, represented by a 1 to 1 association, to appropriately remove whenever the runes are recovered, or if the player dies once more. An advantage of this approach is that the RuneManager instance can be accessed by any other class. A disadvantage is that the implementation (the singleton constructor) is more complex, and can cause problems if done incorrectly.