Design Rationale for Requirement 5

Overview of Abxervyer

Abxervyer is a resident of the Ancient Woods and therefore has the RESIDENT_ANCIENT_WOODS status added during instantiation. Additionally, if wandering on a void, it would not be hurt, therefore, a new Ability was added "NOT_HURT_BY_VOID". This was accounted for in the tick method of the BottomlessPit class. Any actor who did not possess this ability would be hurt. Using an ability allows for extension in future, abiding with the Open/Closed Principle.

Rainy Weather and Sunny Weather

Since the weather on the maps is constantly changing and is controlled by Abxervyer, the playTurn method in the Abxervyer class was used to keep track of time and change the weather accordingly.

The weather has been implemented using the observer principle, the Weather class is an interface with the list of entities and methods to register, deregister and notify the entities. The registering and unregistering of subjects is done in the relevant spawning grounds' tick methods, adding all instances of the actors being spawned and the grounds themselves using an instance of weather that has been passed through. This also allows us to check whether the actor is unconscious, as it would then be unregistered from the affected entities list. The sunnyWeather and rainyWeather classes implement the Weather interface.

All entities affected by weather changes implement AncientWoodEntity, which has a sunnyUpdate and rainyUpdate methods. Once the weather changes, the playTurn method calls the notifyEntity method in SunnyWeather or RainyWeather, which calls either sunnyUpdate or rainyUpdate, respectively. In each entity's respective methods, the changes which occur in different weather are made. This implementation is extensible, as any additional entities affected by the weather change would be able to implement AncientWoodEntities, hence following the Open/Closed Principle. One potential drawback stems from the use of a centralised list; if any other entities, who are not residents of the Ancient Woods, were added to the list, they would also be affected by the weather-changing pattern.

The Single Responsibility Principle (SRP) is respected since the responsibility of each observer is transferred to its sunnyUpdate or rainyUpdate method and does not lie with the Observer classes.

The Dependency Inversion Principle is satisfied as in which whoever wants to observe it must follow some rules, and in particular, the observed subject will call their sunnyUpdate()/ rainyUpdate() function instead of calling the observers' specific functions.

DRY is currently followed but will be violated if two different classes behave the same way in the same weather as code would have to be repeated in their respective methods.
The potential downside of this implementation is that there is a lot more abstraction, hence increasing the complexity.

Abxervyer's playTurn() is used in the Interaction Diagram, specifically when the weather is changed to rainy weather.

The Death of Abxervyer

The only way Abxervyer can be killed is at the hand of another actor, therefore, the appropriate unconscious method was overridden from the Enemy parent class, indicating the Open/Closed Principle was being adhered to, as extensions were added but the parent class did not need to be modified to accommodate the changes.

Upon Abxervyer's death, his last location is turned into a gate to the Ancient Woods. Therefore, to add the MoveActorAction, the map of the Ancient Woods would need to be accessible by Abxervyer. The first implementation saw the Ancient Woods map was passed in to the constructor during instantiation and initialised. This appeared advantageous, as the only possible way the map could be retrieved was from the Application class, where it was initialised. As Abxervyer is added from the Application class, where the maps are created, this was successful. However, if in future, new bosses were able to be spawned, this functionality would no longer work, which may be a violation of the Open/Closed Principle.

A second implementation, using a gate instance passed as a parameter was used. This allowed for a gate to be created in the Application, with the MoveActorAction added, before passing the gate instance through when creating Abxervyer. This proved to be a better implementation, as Abxervyer could now take any gate, which could lead anywhere, rather than having the destination created in the Abxervyer class. This would better implement the Open/Closed Principle. However, there would still be problems if, in future, a boss was able to be spawned randomly at any turn. Therefore, this implementation still needs improvement.

An alternative implementation could see the Player class add the map as an attribute which could be accessed in the unconscious method for Abxervyer. However, this would violate the Single Responsibility Principle, as the Player class should not be responsible for tracking a map which will ultimately be used by another actor. Therefore, the current implementation is likely the best option.