REQUIREMENT 3 - Design Rationale

Classes Modified/Created	Roles/Responsibilities	Rationale
Memento (Created)	When Memento is activated there is a 50% chance that the wielder's health, mana, and strength will be remembered as a state. The other 50% chance will cause the wielder to be restored to the last remembered state. Memento will consume 5 hit points from the wielder every time it is used unless the wielder has 5 or less current hit points in which Memento is not activated.	**<u>Solution:</u>** For Memento class like the other weapon arts it has the method activate which implements what the weapon art does when an actor uses a weapon with Memento attached to it on another actor. It has two abilities which are to save the wielder's current attributes or restore the last state saved using Memento; only one of these will occur when the weapon art is activated. Which are handled by the ActorState class. **Alternatively,** I could have implemented the two methods into the Memento class but due to design considerations and SOLID principles these methods were implemented in the ActorState class. For example, one of the design considerations states that in the future Memento can save the wielder's location and when restored teleport the wielder to that saved location. With the ActorState class it makes it so the Memento class does not need to be modified since functionalities for saving and restoring the location of the wielder can be implemented in the ActorState class. **<u>RELEVANT SOLID PRINCIPLES</u>** **Single Responsibility Principle (SRP)** is followed in Memento class since it has a single responsibility of saving, restoring and health cost logic without concern of the functionalities of other weapon arts. **Open/Closed Principle (OCP)** is adhered to, as mentioned before the class is open for extension but closed for modification. This is due to the ActorState class which saves and restores the states of the actor and

		allows for new attributes such as stamina or location to be implemented without having to modify the Memento class. **Liskov Substitution Principle (LSP)** is also followed as the Memento class can be substituted for any other class implementing the WeaponArt interface. Since all weapon art classes fulfil the activate method. **Interface Segregation Principle (ISP)** is adopted in Memento class due to the WeaponArt interface that it implements as it is only required to implement the activate method and no other unnecessary methods. **Dependency Inversion Principle (DIP)** is also promoted by the design as it allows for higher-level modules (such as weapon or actor classes) to depend on the abstraction provided by the WeaponArt interface rather than concrete implementations provided in the Memento class.
ActorState (Created)	The ActorState class manages the state of an actor at a particular moment in time by storing specific attributes (health,mana and strength). It also creates and returns new states which saves the players current health, mana and strength. Additionally, it can restore states by accessing the attributes saved from creating new states and using these to replace the player's current attributes.	<u>**Solution:**</u> The ActorState class was created to make the design more extensible and flexible since it can be easily reused and modified for future design considerations. <u>**RELEVANT SOLID PRINCIPLES**</u> **Single Responsibility Principle (SRP)** is adhered to as the ActorState class only has one clear responsibility of managing the saving and restoring of an actor's attributes. Whilst the other responsibilities such as when to call these methods are handled in the Memento activate method. **Open/Closed Principle (OCP)** is open for extension as mentioned previously as future attributes and other functionalities such as remembering new attributes (stamina or location) or even remembering attributes of a target actor.
WeaponArt (Modified)	Interface for weapon arts	<u>**Solution:**</u> Originally in A2 this was an abstract

		class in which the weapon arts extended from, however for A3 we decided to change it to an interface with a single abstract method where all weapon arts will implement this interface. There is only one method that is common in all the weapon arts which is the activate method, however the functionality is completely different in all weapon arts so there is no need for default implementation that the abstract class can provide. Additionally, the Memento class has a health cost which is not common in the other weapon art classes which prior to A3 had either no attribute cost or a mana cost. Hence, there are no attributes such as manaCost shared amongst the weapon arts which is another reason why an abstract class is redundant. **<u>RELEVANT SOLID PRINCIPLES</u>** Using a WeaponArt interface, follows the **Single Responsibility Principle (SRP)** since it only defines a contract for weapon arts that can be activated and doesn't handle any of the other functionality leaving it for each implementing class to focus on the specific behaviours. **Open/Closed Principle (OCP)** is also applied since the interface allows for easy extension since creating new weapon arts in the future can be done by simply implementing the interface in the class without having to modify any of the existing code. Since the Weapon Art interface is focused and minimal it addresses the **Interface Segregation Principle (ISP)** by only requiring the implementation of the activate method. In doing so, it ensures that implementing classes are not forced to implement unnecessary methods. **Dependency Inversion Principle (DIP)** this design allows for higher-level modules (such as weapon or actor

		classes) to depend on the abstraction provided by the WeaponArt interface rather than concrete implementations. Consequently, promoting loose-coupling as weapon items and actors only need to interact with the interface without the details of the specific weapon art.