Design Rationale REQ_4

Design Rationale

Implementation & Why:

Traveller:

- Traveller takes in the abxervyer object during instantiation to be used in the condition checking.
- Implements the Monologue interface (Interface Segregation Principle)
- AllowableActions calls the monologue() method that adds the initial monologue not affected by conditions to the monologueOptions arraylist.
- The rest of the monologue is added after the conditions are checked.

Similar implementations as REQ_3, reuse all the implemented interface and functions to adhere to DRY principle and code reusability. The conditions for each required monologue are changed and reimplemented in the Traveller's allowableActions().

Pros:

HasItem utility class follows the single responsibility principle as the class exhaustively takes care of checking if an actor has the item in their inventory. The class method actorHasItem can be used by all actors in the game to find any Item by utilising the getClass() method that gets the root class of the item.

Our implementation of adding and removing from the monologueOptions arraylist makes sure that all the monologue String(s) in the arrayList always fulfils the conditions and can be printed out.

Cons:

The checking for the monologue conditions as well as adding/removing from the list may seem repetitive at times.

Extendability:

The use of the Monologue interface allows future actors that would have the option to have monologue interactions to just implement the interface.

MonologueConditions handles all the conditions and adding/removing of the condition based monologues that allows for high level customization.