Design Rationale for Requirement 4

Implementing monologues

Traveller now implements the Talkable interface which defines talked() method, ensuring that the Traveller class can be listened to by the player. ListenAction is also added to their allowable actions list. Since ListenAction accepts only classes which implement Talkable interface, this aligns with the Interface Segregation Principle.

The code implements the talked() in a way similar to the Blacksmith class from Requirement 3. Since both talked() majorly do the same thing, a new method in the Utility class is added. This method makes a list of phrases based on the conditions as well as randomly chooses a phrase. This way similar code is reduced, and the code follows the Don't Repeat Yourself principle. It also ensures that this functionality is consistent and easy to maintain across Talkable classes.

Except for the creation of a new Utility method, ListenAction and Talkable interface both follow the Open Closed principle, because the new traveller's feature was easily integrated to the existing system without changing much.

Also Traveller's dialogues have multiple conditions while Blacksmith from Requirement 3 has only 1. So in order to handle multiple conditions, monologues now have a list of conditions rather than just one. This was the second adjustment.

Overall, implementing monologues follow the Open Closed principle and now after some refinements, it has become more scalable.

Adding a ListenAction to the allowable actions list as well as executing this action is shown in the Interaction Diagram for the Requirement 4.