Assignment 3 REQ-3 Design Rationale

Implementations:

- **Communicative (interface):** This interface is for game entities that can talk or send messages (in Astley's case a monologue).

- **Astley (concrete class):** This is a new item you can buy from the computer terminal. It extends the Item abstract class and implements the Purchasable and Communicative interfaces. It costs 50 credits and shows up on the map as the character 'z'. Astley can speak to the actor if the actor pays a subscription fee of 1 credit every 5 ticks. The subscription stops when Astley is dropped and starts again when picked up. This logic is handled directly in overriding the tick methods. Astley has six possible monologues, three available from the start and three more that need specific conditions. The method speak(Actor actor) handles this.

- **MonologueAction (concrete class):** This class extends the Action abstract class. It allows entities to send a message when the action is executed.

Modifications (for previous implementations):

- **Enemy abstract classes and subclasses:** Removed default AggressiveBehaviour. This behavior is now added manually based on the subclass's behavior, making sure subclasses can be used without unexpected behavior. Also replaced "magic numbers" with constants for behavior priorities.

- **Application:** Added Astley to the list of purchasable items you can buy from the computer terminal.

Implementation Approach:

The goal is to create Astley, an AI device that speaks to the actor. Astley can be bought from the computer terminal for 50 credits. The Purchasable interface handles the buying logic as previously done on Assignment 2's REQ-4 which implements purchasable items from the shop. So, Astley simply implements the

Purchasable interface and implements its own purchase logic in the purchase method from implementing the interface in which it's just a normal transaction as per the requirement. All we need to do is add Astley into the list of purchasable items in Application.

Reasoning:

- Reusability: By using the Purchasable interface, we can reuse the existing logic for purchasing items, ensuring consistency and reducing code duplication.

- Consistency: Implementing the Purchasable interface ensures that all purchasable items follow the same pattern, making the codebase more uniform and easier to understand.

Astley requires the actor to pay a subscription fee of 1 credit every 5 ticks to use it. If the actor can't pay, they lose access to Astley's functionality and can no longer listen to its monologues. The subscription logic is directly in Astley's tick method for when the item is on an actor's inventory. When dropped, the subscription pauses and resumes when picked up which is handled by overriding the tick method for when its on the ground to pause it. During each turn in order to check whether the actor has paid subscription in order to continue using Astley, we added a boolean check on Astley's allowableActions. We'd add a MonologueAction into possible actions if has paid and just returns an empty list of possible actions otherwise if hasn't paid. Despite not implementing a new interface to help for extension, I believe that this is a better trade-off than making the code too complex and handling too much may mess up some logic as per TA feedback and subscription seems exclusive to Astley and can simply be handled by overriding the tick methods for unique subscription logic.

Reasoning:

- Simplicity: Handling the subscription logic directly within the tick method keeps the code straightforward and localized, making it easier to manage.

- Maintainability: By centralizing the subscription logic in one place, any changes or bug fixes can be made more efficiently without affecting other parts of the code.

Astley can say six different lines, three available from the start and three more under specific conditions. This is managed in the speak(Actor actor) method from the implementing Communicative interface. An interface was made so that our design is

capable of handling future grounds, actors, and other items which can speak so that they simply just need to implement this interface promoting extendibility. This is also the case for a previous implementation which is the consumable interface which allows puddles and consumable items to handle it's own unique consumption logic.

Reasoning:

- Extensibility: The Communicative interface allows for easy addition of new entities that can speak. Future items or actors only need to implement this interface to gain communication capabilities.

- Modularity: Interfaces promote modularity, making the codebase more organized and components more interchangeable.

- Consistency and Reuse: Similar to the Consumable interface, the Communicative interface ensures that all speaking entities follow a consistent pattern, facilitating code reuse and reducing the likelihood of errors.

OOP Principles Applied:

Single Responsibility Principle (SRP):

- The Communicative interface is only for talking. This ensures that any class implementing it only needs to handle communication, promoting a clear separation of concerns.
- The Astley class handles subscription logic and speaking without mixing other responsibilities. This keeps the class focused and the code easier to manage and understand.

Open/Closed Principle (OCP):

- The Communicative interface lets us add new talking items without changing old code. This makes the system flexible and easy to extend with new types of communicative items.
- The Astley class extends Item and uses Purchasable and Communicative without altering existing code. This allows for adding new features to Astley without modifying the existing codebase, ensuring stability and reliability.

Liskov Substitution Principle (LSP):

- The Astley class works wherever a Purchasable or Communicative item is needed, ensuring consistency. This means that any system using these interfaces can work with Astley without any issues.
- Removing default AggressiveBehaviour from Enemy lets subclasses define behavior clearly, ensuring they work correctly when substituted. This makes the behavior of enemies more predictable and customizable according to their specific needs.

Interface Segregation Principle (ISP):

- The Communicative interface only includes methods for talking. This ensures that implementing classes are not burdened with methods they don't need, keeping implementations simple and focused.
- Astley implements only the methods it needs for being bought and talking. This keeps the class lightweight and ensures it only contains relevant functionality.

Dependency Inversion Principle (DIP):

- The Communicative interface is an abstract idea that Astley uses, making the code flexible and easier to maintain. This allows the system to rely on abstractions, which can be implemented by various concrete classes, promoting flexibility and reusability.

DRY Principle:

- Subscription logic is in Astley's tick method and speaking logic in speak, ensuring changes are made in one place, reducing errors. This prevents code duplication, making the system easier to maintain and less prone to bugs.