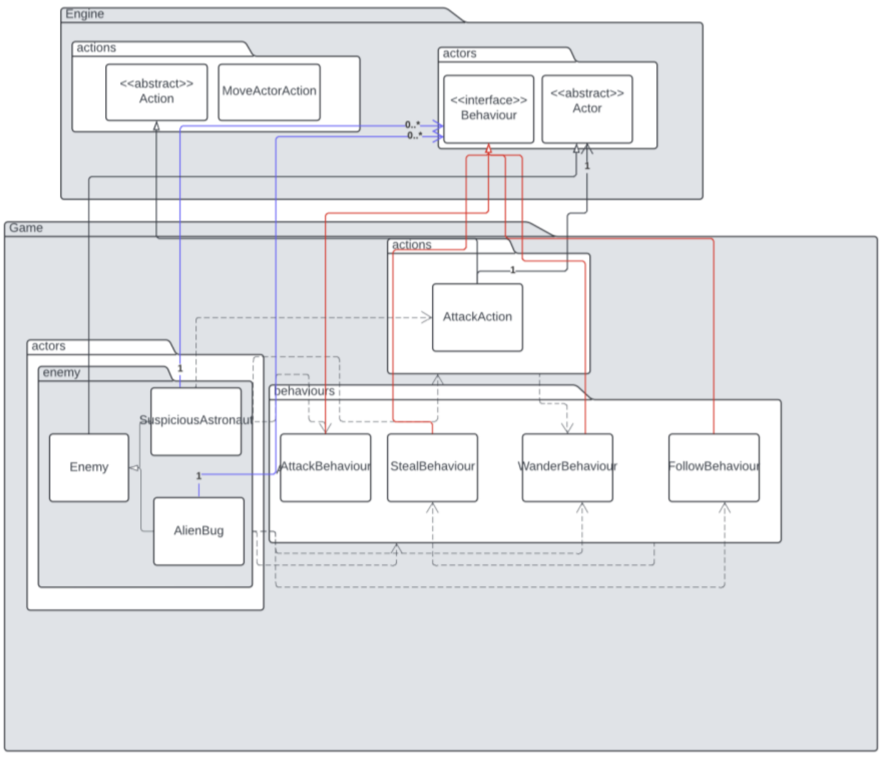
FIT2099 Assignment 2 Design Rationale

Requirement 2: The Imposter Among Us

Design Goal:

The primary design goal for Requirement 2 is to enhance the game by introducing a dynamic enemy interaction with the "Alien Bug" and the "Suspicious Astronaut." These characters add a layer of strategy and suspense to the game, focusing on stealth and resource management. The design emphasizes adherence to good coding practices such as DRY, SOLID principles, and managing connascence to ensure maintainability and extensibility.



Design Decision:

The design integrates two main characters, the "Alien Bug" and "Suspicious Astronaut," using a blend of behaviors that define their interactions in the game environment:

Alien Bug: This character can steal scraps and follow the player. It utilizes behaviors like StealBehaviour, FollowBehaviour, and WanderBehaviour to dynamically interact with the game environment.

Suspicious Astronaut: Acts as a deceptive threat, mimicking an astronaut to blend into the environment. It is mainly passive unless provoked.

Both characters are derived from an Enemy abstract class, ensuring they adhere to the game's enemy mechanics and inherit common properties and methods, promoting reusability and reducing redundancy.

Advantages and Disadvantages of the Design:

Pros:

Extensibility (OCP): New enemy types can be added without modifying existing code. They simply need to extend the Enemy class and implement required behaviours.

Maintainability (SRP): Each class and behaviour have a single responsibility, making the system easier to understand and maintain.

Flexibility (LSP): The design allows substituting any class that extends Actor with its subclasses without affecting the system's functionality.

Cons:

Complexity: The introduction of multiple behaviours and their interactions can increase system complexity, potentially raising the learning curve for new developers.

Performance Concerns: More dynamic interactions and behaviours might impact performance, especially if not well-managed or optimized.

Reasons for Current Design Choices:

DRY Principle: By using behaviours like StealBehaviour and FollowBehaviour, which are encapsulated and reusable across different enemy types, the design avoids code duplication.

Connascence: The system manages connascence by clearly defining interfaces and expected behaviours, which minimizes the dependency on specific implementations.

SOLID Principles: The design strictly follows SOLID principles, ensuring that each class is extendable, modular, and has a single responsibility.

Extension Potential:

The current design is well-prepared for future extensions:

New behaviours can be added to introduce different enemy interactions.

New enemy types can seamlessly integrate into the existing structure by extending the Enemy class.

Changes to game mechanics or enemy behaviours can be made with minimal impact on existing code.

Conclusion:

The design for Requirement 2 provides a robust and flexible framework that enhances the gameplay experience while adhering to best coding practices. This approach not only ensures code quality but also supports future game development and feature integration, paving the way for a scalable and maintainable codebase.