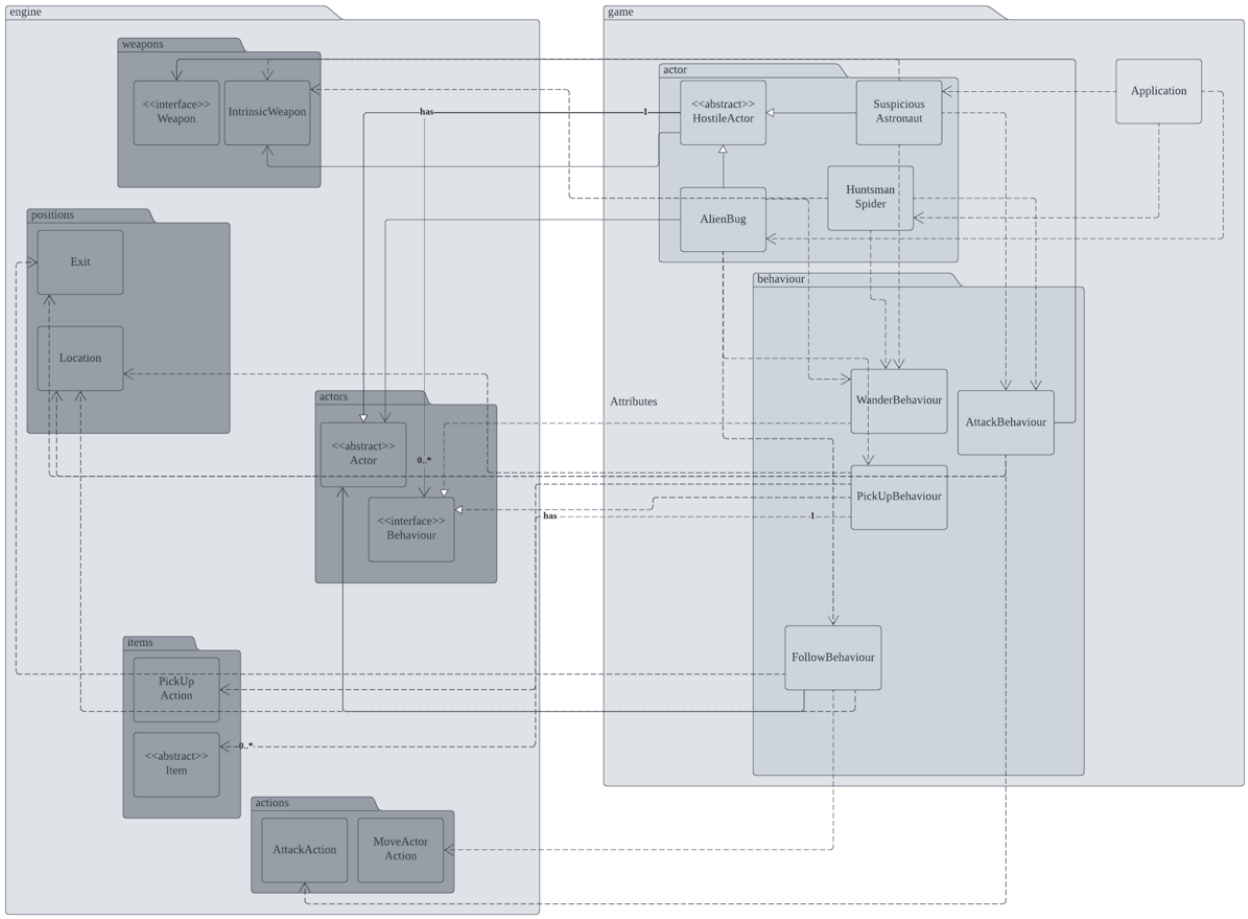
Assignment 2 Design Rationale

Requirement 2: The imposter among us

UML Diagram:



Design Goal:

The design goal for this requirement is to implement the alien bug and suspicious astronaut NPC and its required behaviours while adhering closely to relevant design principles and best practices.

Design Decision:

In implementing Requirement 2, the decision was made to have the alien bug and suspicious astronaut extend from the hostile actor abstract class. This decision was made as the alien bug and suspicious astronaut are both hostile to the player character, hence they are made as children of the hostile actor abstract class. Most of the behaviour decision making is made in the hostile actor class itself, so no modification has to be made to introduce new hostile actors to the game.

Alternative Design:

One alternative approach could involve having all the hostile actors in the future extend from the actor class directly. However, if the approach were to be chose, future implementations will need to be heavily modified to fit within the new system. This can cause bugs and frequent typos to implement these new actors, which can slow down the development process.

Analysis of Alternative Design:
The alternative design is not ideal because it violates various Design and SOLID principles:

1. **[DRY]:**
    - The alternative design would require repeats of the behaviour decision making process, which will violate the Don't Repeat Yourself principle.
2. **[OCP]:**
    - To add new hostile actors, the implementation of these new actors would need to be heavily to work within the new system.

Final Design:
In our design, we closely adhere to the DRY and SOLID principles.

1. **[DRY]:**
    - **Application:** In the chosen diagram, the behaviour decision making process has moved to the hostile actor class, instead of being implemented in each of its subclasses
    - **Why:** By adhering to the DRY principle, the code is kept short and is easily readable and maintainable.
    - **Pros/Cons:** The code is encapsulated in its respective classes, which makes it easier to read and maintain. However, we have more classes in the final design which may look confusing at first glance but is easier to understand.
2. **[OCP]:**
    - **Application:** The design adheres to the OCP as the hostile actor class allows for easy extension, with no modifications required to add new hostile actors.
    - **Why:** As the game is more developed, more hostile actors will need to be added in the future. By adhering to this principle, more hostile actors can be added easily.
    - **Pros/Cons:** The main benefit to this design is that it allows for easier extendibility for future hostile actors.

Conclusion:

Overall, our chosen design provides a robust framework for the completion of requirement 2. It adheres to the design principles that were taught in the unit, such as DRY and OCP. The design we used here is efficient, easily maintainable and understandable by developers working on the project, which helps to reduce the development time of the project.