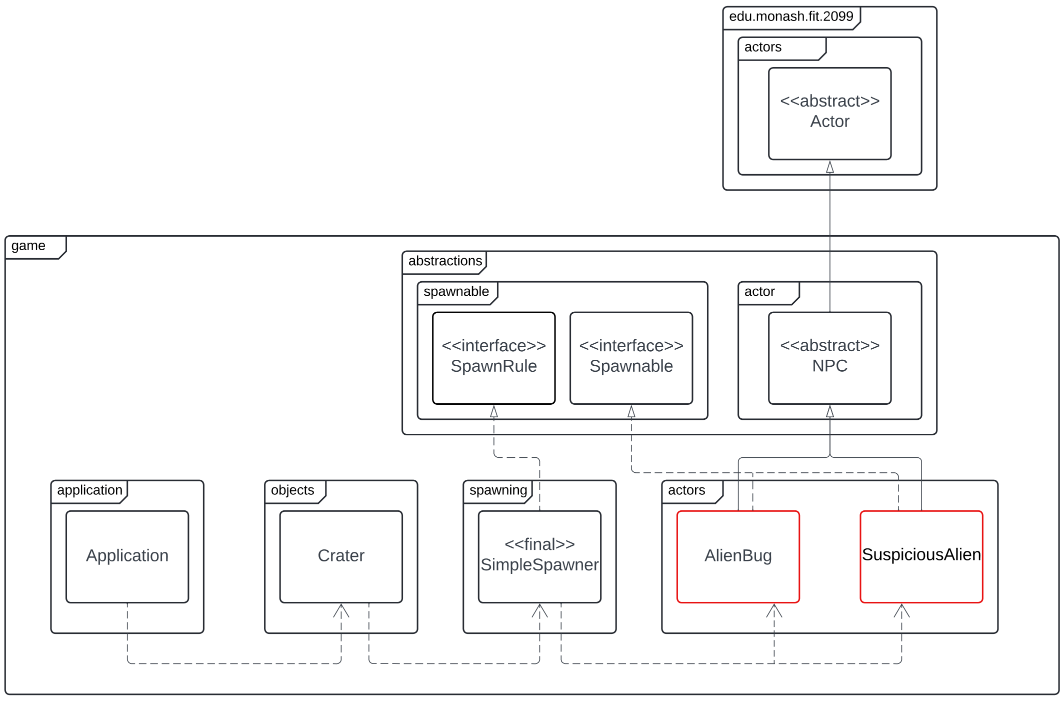
Assignment 2 Design Rationale

Requirement 1: The moon's (hostile) fauna II: The moon strikes



back

Design Goal

The design intention of extending the 'Static Factory' game was to build upon the existing player interaction within the game by introducing new non-Player characters (represented by the red outline); similar to the existing functionality of the HuntsmanSpider class, AlienBug and SuspiciousAlien should implement their respective behaviours (hostile or not) and their attributes (health, attack logic if any). In doing so, the games spawn logic is to be preserved, where a given concrete class (inheriting Actor or Item) implements interface Spawnable allowing the instance to be spawned (if permitted).

Design Decision

To adhere to SOLID Principles, the concrete classes (AlienBug, SuspiciousAlien) extend abstract class NPC (extending Actor) allowing for behaviour attribution specific to the creature; as per the requirement, AlienBug is not hostile to the Intern, and therefore only has a 'Follow and Wander' behaviour whereas SuspiciousAlien possess hostile behaviour. Additionally, the two creatures both have spawning ability specific to their 'spawn chance' at each tick of the game; to implement this, the entity and its respective spawn chance is passed to the final class SimpleSpawner, which handles the spawning logic through the implementation of interface SpawnRule.

Alternative Design

An alternative design to implementing the creature classes and their respective spawning attributes may see the NPC classes have their ‘spawn chance’ as an attribute of the class. However, the use of magic number (spawn chance) leads to obscurity in design intention and makes it difficult for future extension, thus this design approach was not chosen.

```
public class AlienBug extends NPC implements Spawnable {

        private final double spawnChance = 0.1;

        public AlienBug() { super ( “Alien Bug”, ‘a’, 2 ); }

}
```

Analysis of Alternative Design

1. **Single Responsibility Principle**
   - The spawn logic should be delegated to a designated spawn interface that handles the responsibility as opposed to the creature class.

Final Design

The chosen design better adheres to SOLID Principles while maintaining a roust codebase and ensuring multiple inheritance for easier future extension.

1. **DRY**
   - As AlienBug and SuspiciousAlien extend NPC, the codebase is more robust through extending these identities maintaining readability. This allows the creatures share the same base functionality but with different behaviours.
2. **Open – Closed**
   - Through the abstraction of the spawning logic, the addition of ‘spawnable’ creatures demonstrates extending the existing implementation, without further modification.
3. **Liskov – Substitution**
   - As the creature classes implement Spawnable, the upcasting of NPC objects when passing to a Crater object allows for different creatures to be spawned by Craters during the game. This allows for flexibility in future extensions as there may be various creatures that can spawn from a given ground object.
4. **Single – Responsibility**
   - Utilising an interface to delegate spawning logic; as a result creature instances are not responsible their spawning implementation. This ensures the code is more scalable and extensible in future iterations.