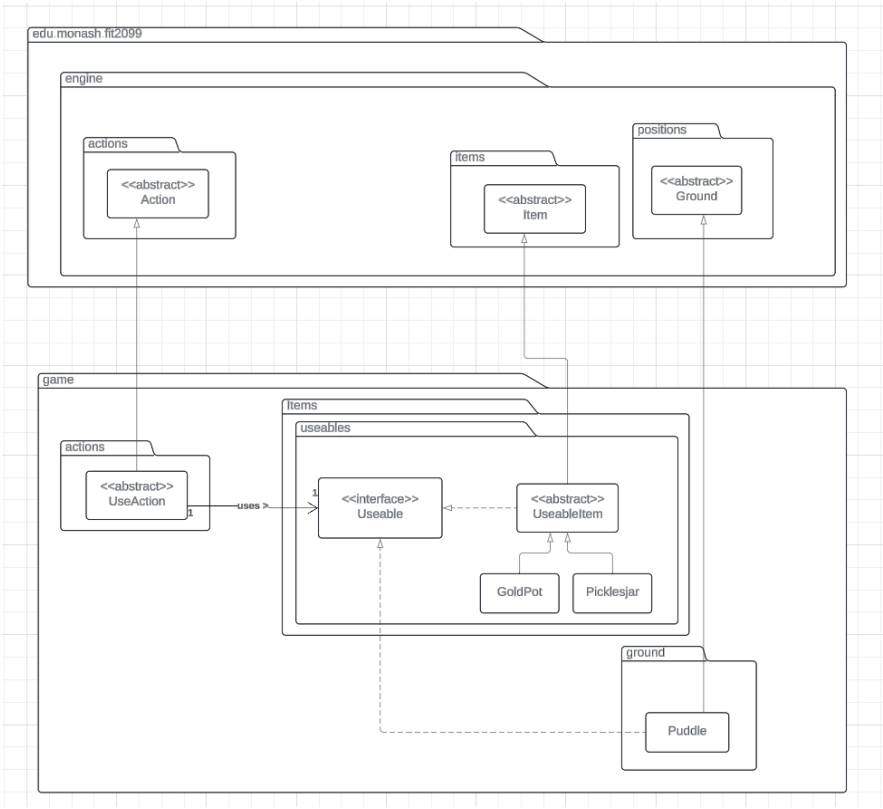

The design diagram represents the implementation of the Puddle, GoldPot and PicklesJar class that incorporates the capability for user interactions with objects.

This design differs from the one proposed in Assignment 1. In Assignment 1, a ConsumeAction class was created to allow for users to eat items. In Assignment 2, this class has been refactored to UseAction so that the class can handle different types of player interactions with game entities (eating, drinking, taking from etc.), therefore increasing reusability and coherence of the code.

The UseableItem abstract class extends from Item and implements the Useable interface. Items that the users can interact with share a common attribute: value, which denotes the player's potential gain or loss upon their use, and they also share the same methods. Thus, it was logical to create an abstract class that useable items can extend from to avoid code repetition (DRY). This design choice also allows for extensibility as new usable items can extend from the abstract parent class without modifying previous code (Open-closed Principle). Likewise, if the game were to later introduce more ground or actor entities that could be used, we can implement the same design to accommodate for it. However, a disadvantage of this design might be that it introduces complexity into the code as it adds another level of inheritance.

Through abstraction, the UseAction associates with the Useable interface rather than individual useable objects (Dependency Inversion Principle). This promotes extensibility and reusability as different types of entities; actors, items and ground can all use the UseAction without any adjustments needing to be made.