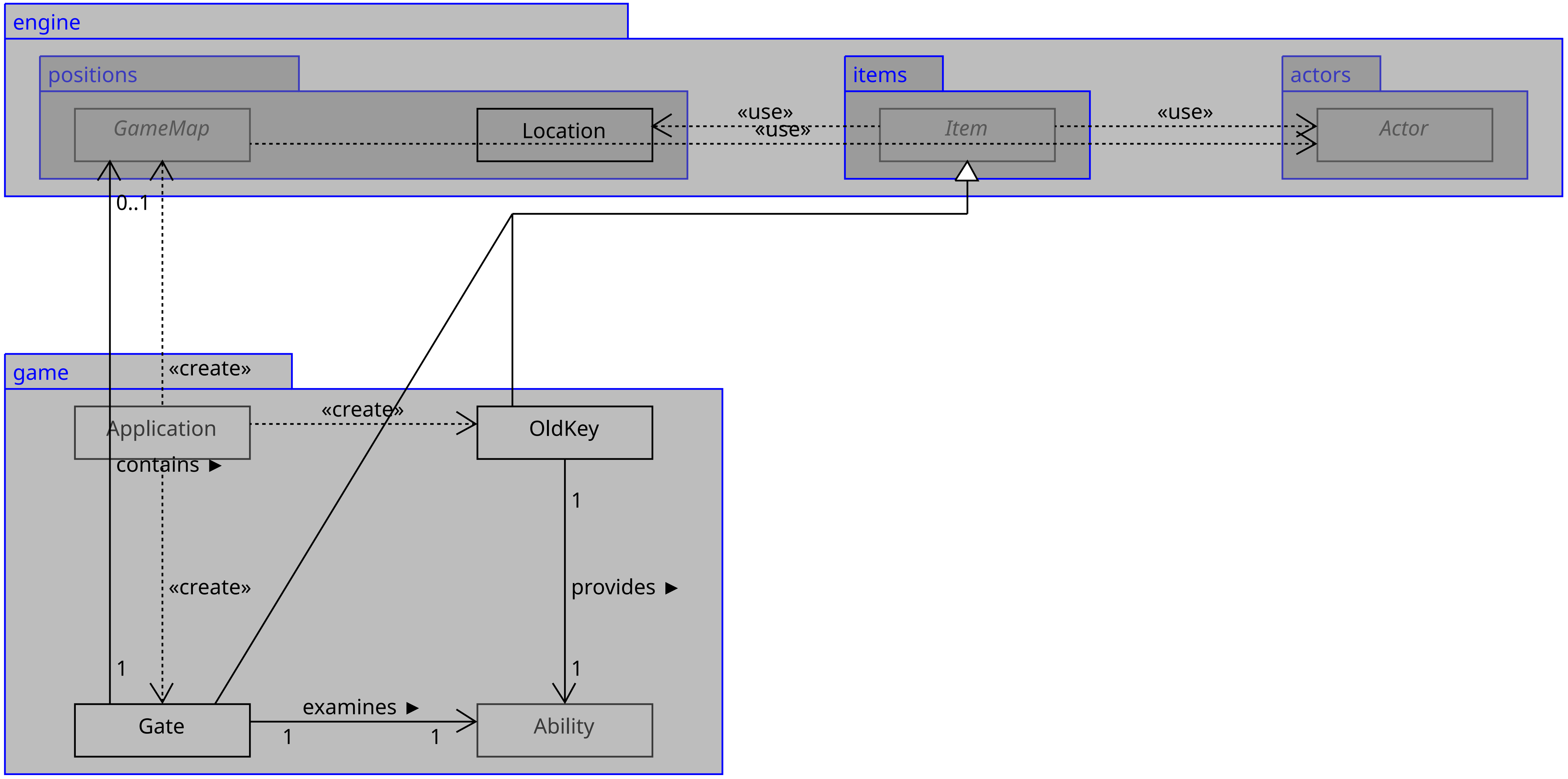
Design Rationale

The Gate class abstracts the concept of a teleportation gate between two game maps. It encapsulates the origin and destination maps, coordinates, and the ability required to use the gate. This design adheres to the SRP as the Gate class has a single reason to change: when the teleportation behavior needs modification. The Gate and OldKey classes inherit from the base Item class. This design follows the Open/Closed Principle because new types of items or gates can be added without modifying existing code. For instance, you can easily add more types of keys or gates with different abilities without altering the core logic.

The Gate class also encapsulates the teleportation logic, ensuring that it is not scattered throughout the codebase. This adheres to the DRY principle, as teleportation is defined in one place, making it easy to modify and maintain. The Ability enum is used to represent the required key to unlock a gate. This design decision promotes code readability and scalability. For instance, if more abilities are introduced in the game, you can easily extend the enum without altering existing code.

The code is designed to be easily extendable. New game maps, gates, or keys with different abilities can be added without significant changes to existing classes. This follows the Open/Closed Principle as the code is open for extension but closed for modification. It also handles scenarios where an actor attempts to use a gate without the required ability. It checks for the player's capability before teleporting, preventing unauthorized access.


engine
positions
GameMap
Location
«use»
«use»
items
Item
«use»
actors
Actor
0..1
game
«create»
Application
«create»
OldKey
contains ▶
1
«create»
provides ▶
1
1
Gate
examines ▶
1
1
Ability