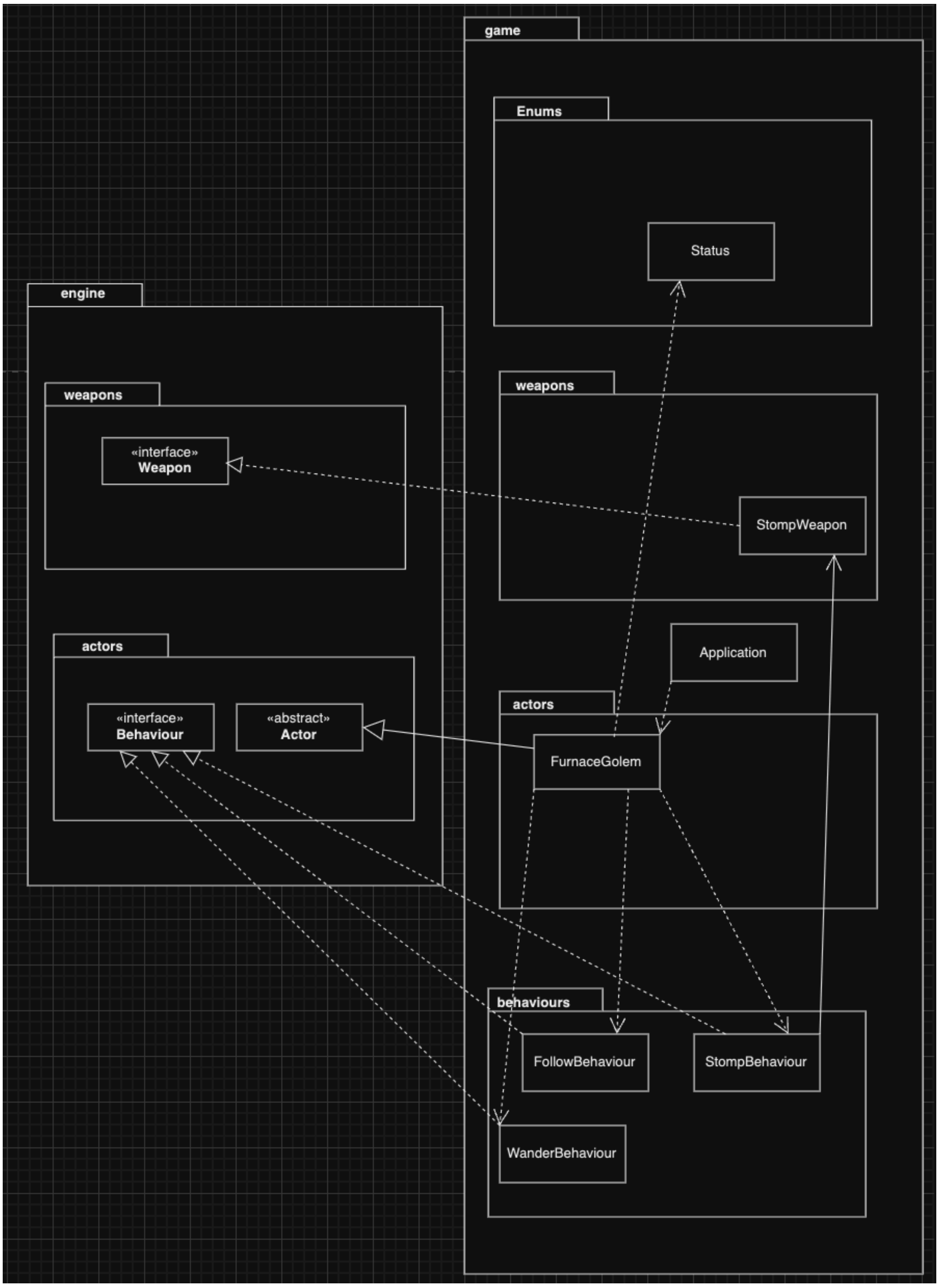
REQUIREMENT 3



Classes Involved:

- **ConsumableItem (Interface)**: Defines the `consume()` method for any item or entity that can be consumed. It ensures flexibility by allowing any consumable to define its own behavior when used.
- **FlaskOfHealing (Class)**: Heals the Tarnished by 150 HP, with 5 charges. When all charges are depleted, it can still be consumed, providing feedback that it's empty.
- **FlaskOfRejuvenation (Class)**: Restores 100 mana points with 3 charges. Similar to FlaskOfHealing, it can still be consumed when empty, showing the player it's out of charges.
- **ShadowTreeFragment (Class)**: A one-time consumable that increases max HP by 50, max mana by 25, and strength by 5. Once consumed, it's removed from the player's inventory.
- **ConsumeAction (Class)**: Handles the action of consuming items, interacting with the `consume()` method in the ConsumableItem interface, making it flexible for various items.

Roles and Responsibilities:

- **ConsumableItem (Interface)**: Provides a contract for any consumable item, ensuring all consumables implement the `consume()` method. It follows the **Interface Segregation Principle (ISP)**, allowing flexibility for each item's behavior.
- **FlaskOfHealing & FlaskOfRejuvenation**: These manage healing and mana restoration with charge limits and handle feedback when empty. They adhere to the **Single Responsibility Principle (SRP)** by focusing solely on their respective effects.
- **ShadowTreeFragment**: Increases stats and is removed after consumption, also adhering to **SRP** by focusing on a one-time boost.
- **ConsumeAction**: Manages item consumption and invokes the appropriate `consume()` method. It follows the **Open/Closed Principle (OCP)**, allowing future consumables to work without modifying this class.

Class Interactions:

- **Player and ConsumableItem**: The player interacts with items through the ConsumableItem interface. Depending on the item, it may heal, boost stats, or do nothing if empty.
- **Flasks**: Both flasks track charges and inform the player when empty, without removing themselves from the game.
- **ShadowTreeFragment**: Once used, it is permanently removed from the inventory.

Coupling and Cohesion:

- **Low Coupling**: The ConsumableItem interface ensures flexibility by allowing new consumable items to be introduced without altering existing code.
- **High Cohesion**: Each class has a single focus (e.g., healing, boosting stats, managing charges), which simplifies maintenance and extension.

SOLID Principles:

- **SRP**: Each class has a clear, singular responsibility (e.g., FlaskOfHealing handles healing only).
- **OCP**: New consumables can be added without modifying existing code.
- **LSP**: Any consumable item can replace another where the ConsumableItem interface is used.
- **ISP**: ConsumableItem defines only the `consume()` method, ensuring focused responsibilities.
- **DIP**: High-level components (like the player) rely on abstractions (ConsumableItem), not specific implementations.

Scalability and Future Extensions:

- **Future Consumables**: Easily add new items like poison or stamina potions by implementing ConsumableItem.
- **Beyond Healing**: The design allows for different effects like poisoning, buffs, or stamina restoration.
- **Non-Item Consumables**: The interface could be applied to environment-based consumption, such as drinking from a fountain.
    -
