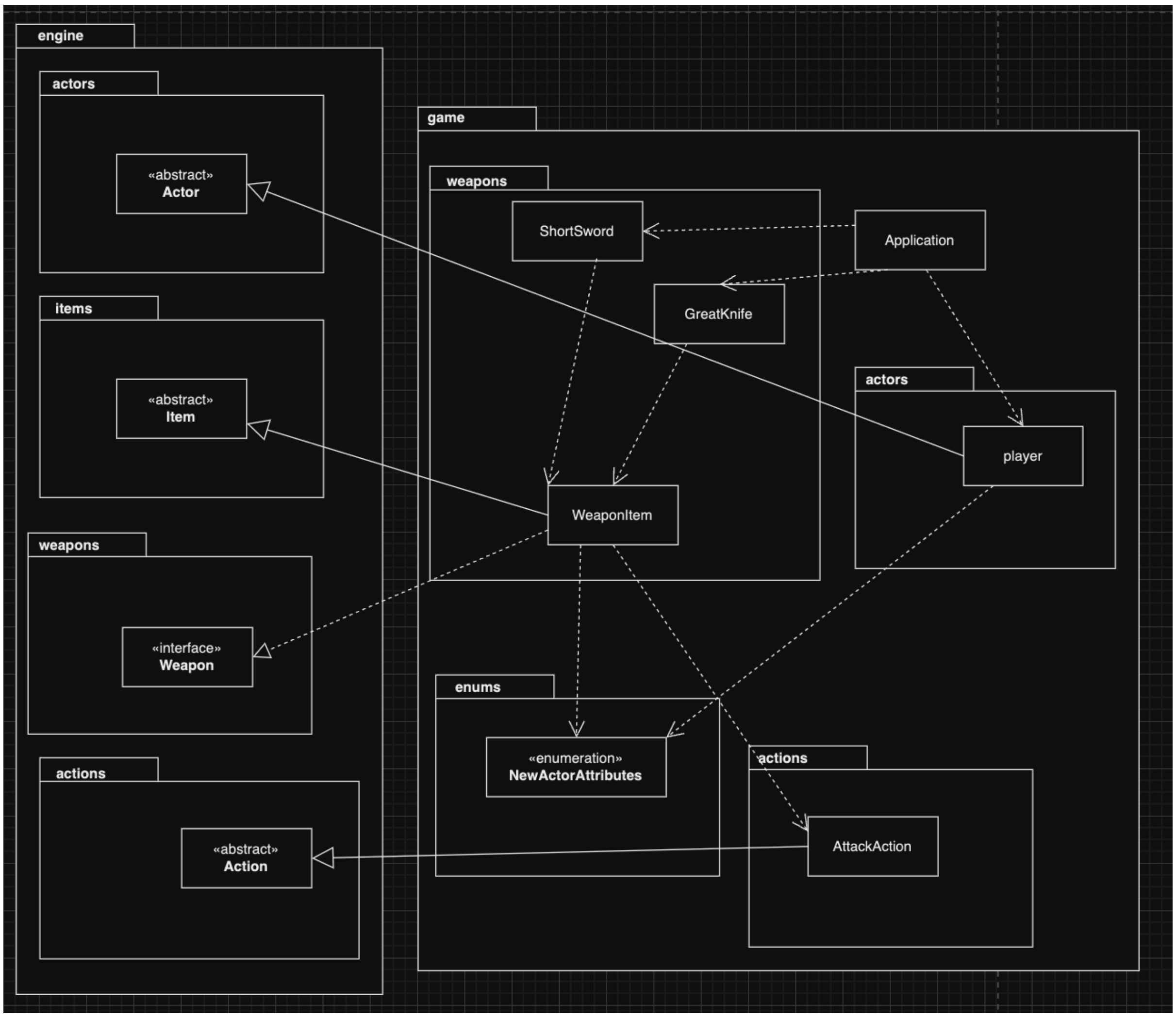
REQUIREMENT 1



Classes involved:

1. Player (Tarnished):
    - This class represents the playable character, the Tarnished, who starts with 150 hit points, 100 mana, and 5 strength.
    - The Tarnished will display its attributes (hit points, mana, and strength) at every game tick.
    - The player can pick up multiple weapons (e.g., Short Sword and Great Knife) and choose which weapon to attack with.
2. WeaponItem (Short Sword and Great Knife):
    - Represents items that can be used as weapons in the game.
    - Weapons have damage, hit rates, and strength requirements. These attributes are crucial for whether a player can pick them up.
    - The getPickUpAction method checks whether the Tarnished has sufficient strength to pick up the weapon. Once picked up, the weapon is stored in the player's inventory.

3. ShortSword (extends WeaponItem):
    - Represents the short sword weapon with predefined attributes.
    - Requires at least 10 strength points to be picked up and deals 100 damage with a 75% hit rate.
4. GreatKnife (extends WeaponItem):
    - Represents the great knife weapon with predefined attributes.
    - Requires at least 5 strength points to be picked up and deals 75 damage with a 60% hit rate.
5. BareFist (extends WeaponItem):
    - Represents the default weapon when no other weapons are used.
    - Deals 25 damage with a 50% hit rate.

Roles and Responsibilities of New/Modified Classes:

1. Player (Tarnished):
    - Role: Central actor controlled by the player, capable of attacking enemies, picking up weapons, and displaying vital stats each tick.
    - Responsibilities:
        - Manage multiple weapons.
        - Display health, mana, and strength at each tick (as per Alexander's quote).
        - Allow the player to pick up weapons, attack with them, and drop them if necessary.
2. WeaponItem:
    - Role: Base class for all weapons, defining core attributes like damage, hit rate, and strength requirements.
    - Responsibilities:
        - Handle weapon pickups and drops.
        - Manage the attack logic (e.g., calculating hit success and applying damage).
        - Ensure only players with sufficient strength can pick up weapons.
3. ShortSword & GreatKnife:
    - Role: Specific weapon classes that extend WeaponItem with unique attributes.
    - Responsibilities:
        - Predefine damage, hit rate, and strength requirements for the specific weapon type.

Interactions between Classes:

1. Player and WeaponItem:
    - The Player class can pick up multiple weapons (e.g., ShortSword, GreatKnife), store them in its inventory, and select any to attack with during combat.

    - The getPickUpAction in WeaponItem checks the player's strength to see if they can pick up the weapon. If the strength requirement is met, the weapon is added to the player's inventory.
    - During each turn, the player can decide which weapon to use, leveraging the allowableActions method from WeaponItem to display available attack options.
2. Player Display:
    - At every tick, the player's attributes (health, mana, and strength) will be displayed to the player via the playTurn method.

Coupling and Cohesion:

- Low Coupling: Each class has a clear responsibility, with minimal overlap. For instance, the Player class handles character attributes and actions, while the WeaponItem class is solely responsible for weapon-specific logic. This reduces interdependencies, making it easier to modify or extend the system without affecting unrelated classes.
- High Cohesion: Classes are focused on single, well-defined tasks. The Player class manages the player's state and decisions, while WeaponItem is responsible for all weapon-related logic. This ensures that each class performs its role effectively without unnecessary complexity.

Pros and Cons of the Design:

Pros:

- Reusability: The WeaponItem class can be easily extended to add new weapons without modifying the core system. Each weapon only needs to define its damage, hit rate, and strength requirements.
- Scalability: The system can be extended to include new features (e.g., new weapons, abilities, or maps) with minimal changes to existing classes.
- Maintainability: By following principles like DRY (Don't Repeat Yourself) and separating responsibilities, the codebase is easier to maintain. Any updates to the WeaponItem class, for example, will automatically apply to all weapons that extend it.

Cons:

- Complexity in Weapon Selection: Managing multiple weapons in the player's inventory may require additional logic to handle scenarios where the player has many items, potentially complicating the game menu.
- Tight Dependency on WeaponItem: Since both the ShortSword and GreatKnife rely heavily on the base WeaponItem class, changes to this base class could ripple across all weapon types.

Delivering Required Functionality:

1. Weapon Management:
    - Players can pick up multiple weapons and choose between them to attack enemies. Strength requirements ensure that players can't pick up certain weapons until they are strong enough.
2. Player Vital Stats:
    - Health, mana, and strength are displayed at every turn to the player, fulfilling the requirement from the game narrative.
    - Strength has its own “NewActorAttributes” class to accommodate for its absence in the “BaseActorAttributes” class
3. Weapon Damage and Hit Rates:
    - Each weapon has a predefined damage value and hit rate, which are factored into attacks against enemies.