Core Systems Design Analysis

These four UML diagrams for "ELDEN THING: THE VALLEY OF THE INHERITREE" reveal a well-structured game architecture organized around four interconnected systems: Non-Hostile Creatures, Plant/Environment, Seed/Planting, and Talisman/Curing. This design effectively balances inheritance hierarchies with composition patterns to create a cohesive gameplay experience while maintaining modularity.

Non-Hostile Creatures System

The creature system introduces the concept of entities that can become corrupted over time. Central to this design is:

The Rottable interface defining core rot mechanics (reset timer, check curability)
RottableActor as an abstract class extending Actor and implementing the rot timer logic.
Concrete implementations (SpiritGoat, OmenSheep) that use composition with WanderBehaviour for movement.
Status enumeration (NON_HOSTILE) identifying friendly entities.
This design demonstrates good separation of concerns, isolating movement logic from entity definition through composition. The inheritance hierarchy establishes a clear relationship between standard actors and those affected by rot mechanics, while the timer implementation in RottableActor provides a unified approach to environmental degradation.

Plant and Environment System

The environment system forms the ecological backbone of the game world:

An abstract Plant class extending Item that uses template method pattern through applyEffect.
A CurableGround interface allowing environmental healing.
Two contrasting ground states (Blight, Soil) representing corruption and health.
Two plant types with opposing effects (Inheritree for healing, Bloodrose for damage).
This system creates environmental dynamics where player actions have visible consequences, enhancing game immersion while maintaining clean separation between distinct ground types and plant behaviors.

Seed and Planting System

The seed system provides player agency to modify the game environment:

An abstract Seed class with specialized implementations (InheritreeSeed, BloodroseSeed).
A PlantAction class that encapsulates the planting behavior.
Clear relationships showing how seeds create their respective plants.
Environmental interactions (e.g., InheritreeSeed converting Blight to Soil).
This design demonstrates effective use of the command pattern through PlantAction, allowing the complex behavior of planting to be encapsulated and reused.

Talisman and Curing System

The Talisman system provides a specialized tool for combating corruption:

Talisman extends WeaponItem to function as both weapon and healing tool.
TalismanCureAction targeting both Rottable actors and CurableGround.
Stamina management for balance (applyStaminaCost).
Flexible targeting mechanism through allowableActions.
This multi-purpose tool design creates strategic choices for players while maintaining a unified interface for curing different types of corruption.

Design Principles Applied

The architecture demonstrates several sound design principles:

Interface Segregation: Small, focused interfaces (Rottable, CurableGround) serving specific purposes.
Open/Closed: Systems designed for extension without modification (new plants, seeds, creatures).
Template Method Pattern: Abstract behavior in Plant with specialized implementations.
Composition over Inheritance: Behavior sharing through composition (e.g., WanderBehaviour).
Command Pattern: Actions encapsulating complex behaviors (PlantAction, TalismanCureAction)
System Integration.
The four systems integrate cohesively:

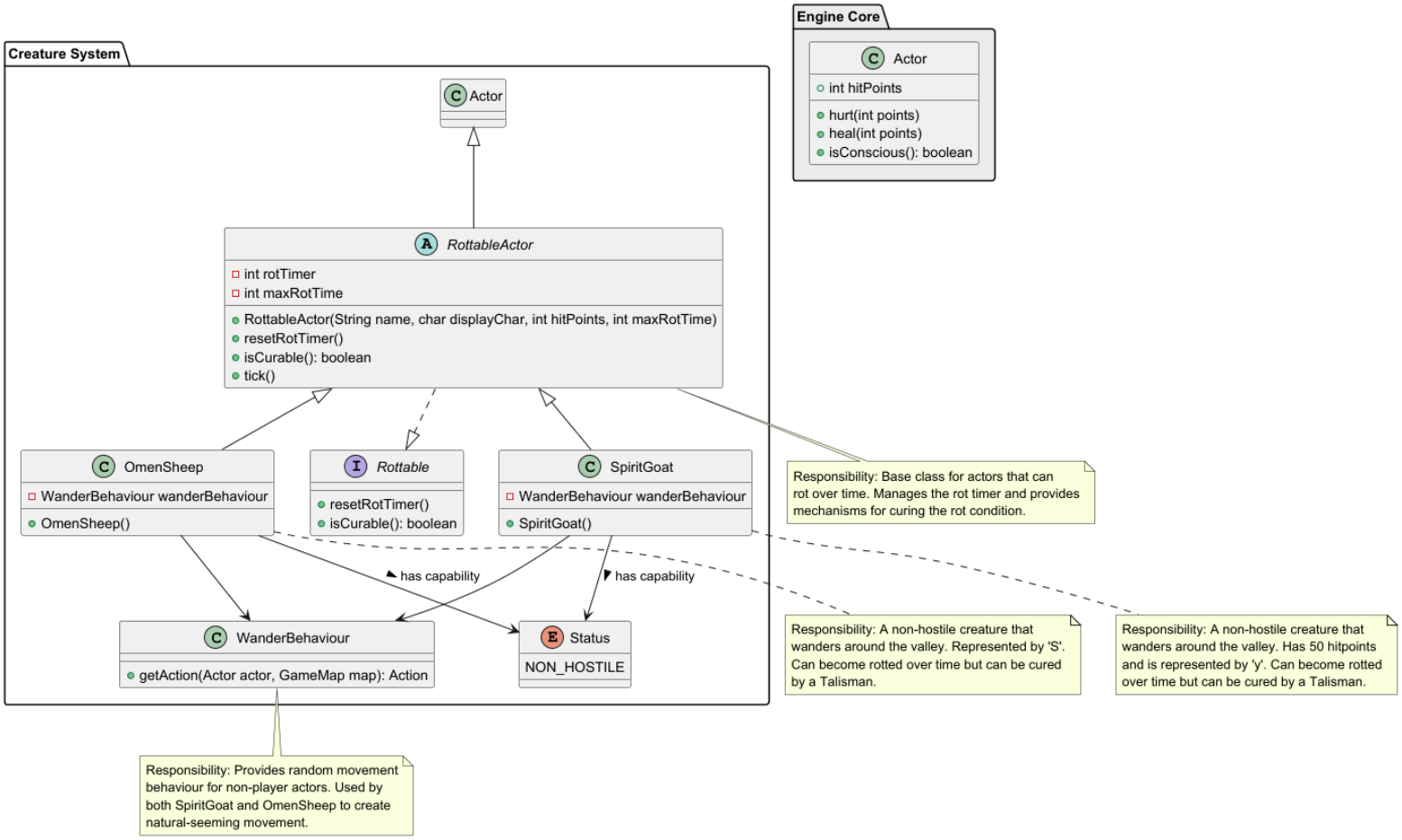
Creatures can become corrupted over time and cured by the Talisman.
Seeds create plants that modify the environment and affect creatures.
The environment can be corrupted (Blight) or healthy (Soil).
The Talisman provides a means to combat corruption in both creatures and environment.
This interconnected design creates a dynamic gameplay loop where player actions have cascading effects across all systems.

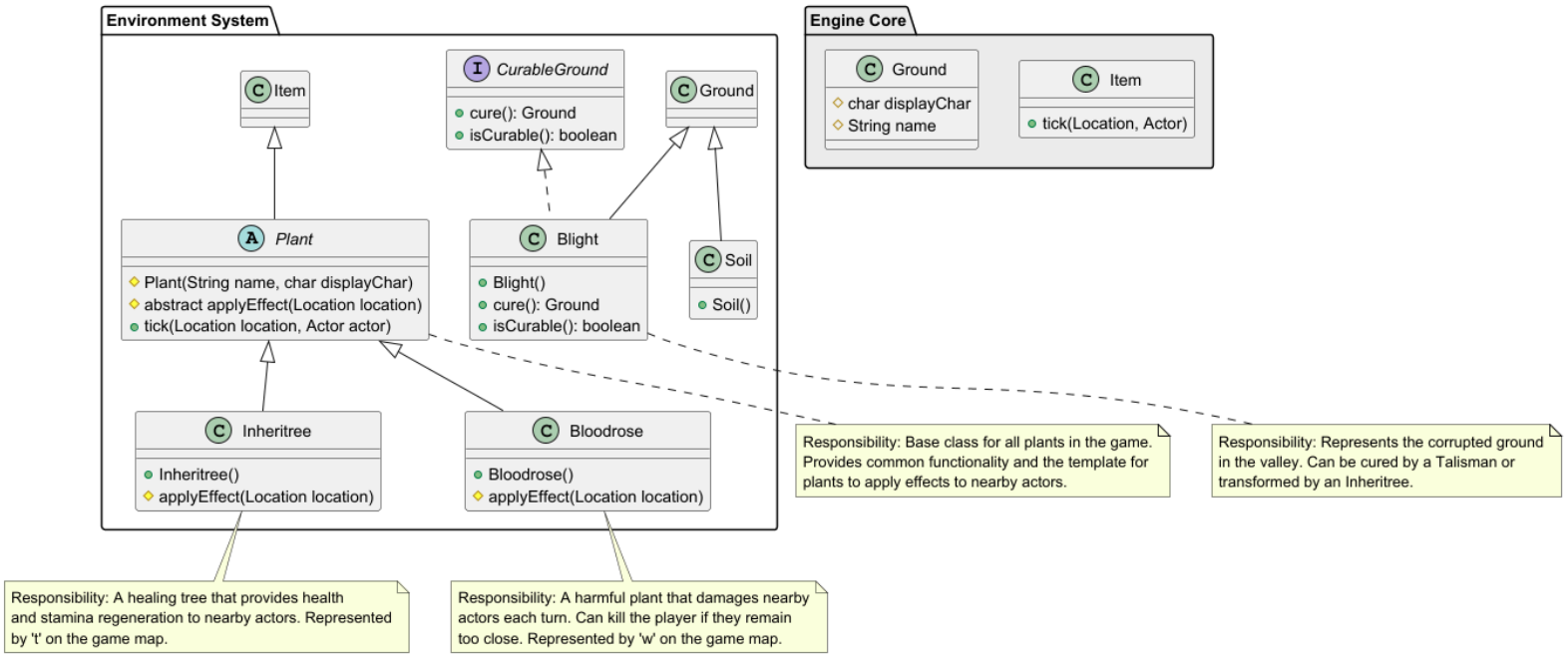
Design Evaluation

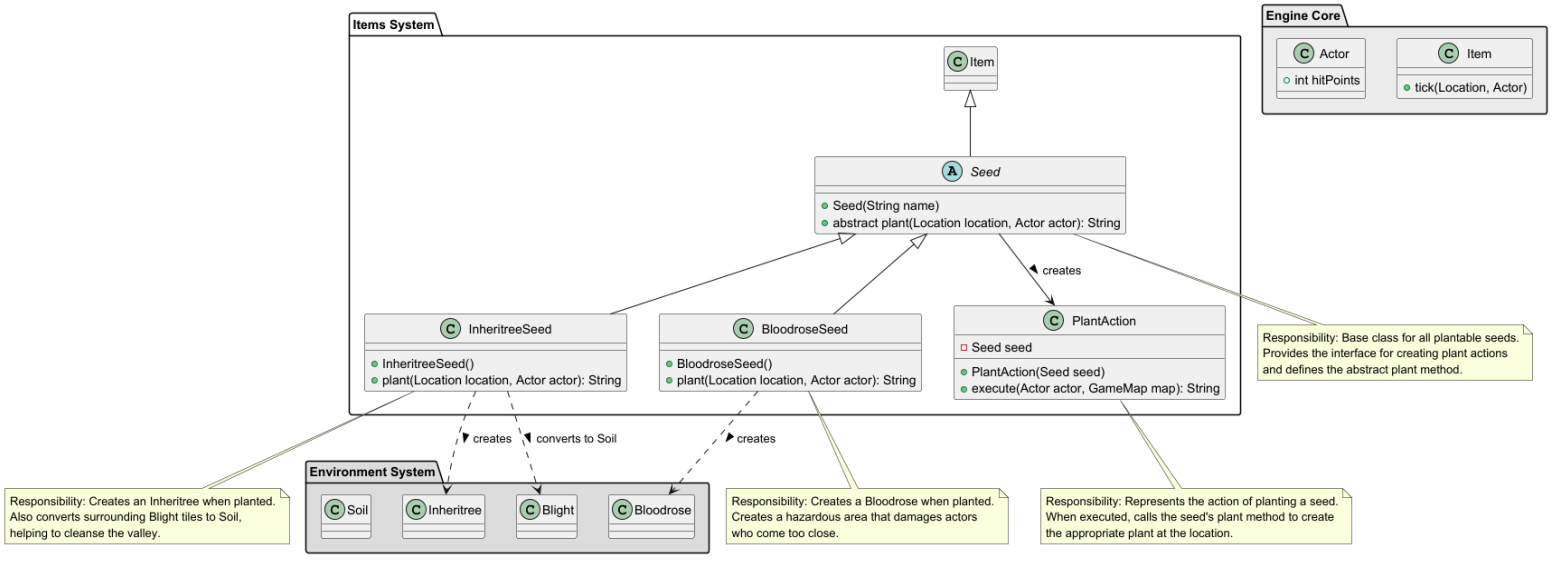
The overall architecture strikes an effective balance between flexibility and comprehensibility. The clear separation of concerns and consistent use of design

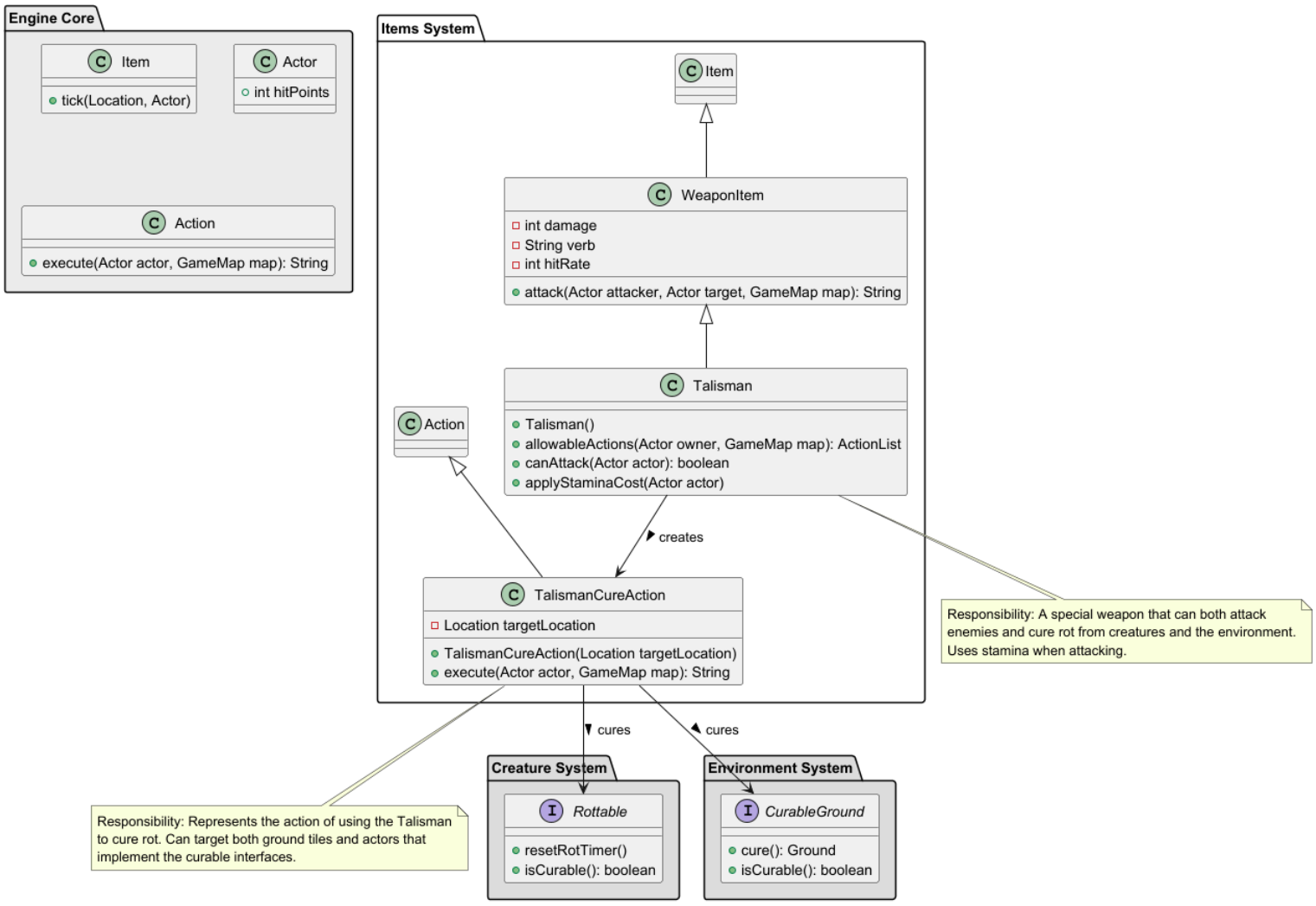
patterns creates a maintainable codebase that can evolve with additional game features. The heavy use of interfaces and abstract classes facilitates future extensions while maintaining system cohesion.

This design successfully establishes a foundation for the game's core mechanics while leaving room for expansion through additional creature types, plants, environmental effects, and player tools.




Environment System
Item
CurableGround
cure(): Ground
isCurable(): boolean
Ground
Plant
Plant(String name, char displayChar)
abstract applyEffect(Location location)
tick(Location location, Actor actor)
Blight
Blight()
cure(): Ground
isCurable(): boolean
Soil
Soil()
Inheritree
Inheritree()
applyEffect(Location location)
Bloodrose
Bloodrose()
applyEffect(Location location)
Engine Core
Ground
char displayChar
String name
Item
tick(Location, Actor)
Responsibility: Base class for all plants in the game. Provides common functionality and the template for plants to apply effects to nearby actors.
Responsibility: Represents the corrupted ground in the valley. Can be cured by a Talisman or transformed by an Inheritree.
Responsibility: A healing tree that provides health and stamina regeneration to nearby actors. Represented by 't' on the game map.
Responsibility: A harmful plant that damages nearby actors each turn. Can kill the player if they remain too close. Represented by 'w' on the game map.


Items System
Item
Seed
Seed(String name)
abstract plant(Location location, Actor actor): String
creates
InheritreeSeed
InheritreeSeed()
plant(Location location, Actor actor): String
BloodroseSeed
BloodroseSeed()
plant(Location location, Actor actor): String
PlantAction
Seed seed
PlantAction(Seed seed)
execute(Actor actor, GameMap map): String
Engine Core
Actor
int hitPoints
Item
tick(Location, Actor)
Responsibility: Base class for all plantable seeds. Provides the interface for creating plant actions and defines the abstract plant method.
creates
converts to Soil
creates
Environment System
Soil
Inheritree
Blight
Bloodrose
Responsibility: Creates an Inheritree when planted. Also converts surrounding Blight tiles to Soil, helping to cleanse the valley.
Responsibility: Creates a Bloodrose when planted. Creates a hazardous area that damages actors who come too close.
Responsibility: Represents the action of planting a seed. When executed, calls the seed's plant method to create the appropriate plant at the location.


Engine Core
Item
tick(Location, Actor)
Actor
int hitPoints
Action
execute(Actor actor, GameMap map): String
Items System
Item
WeaponItem
int damage
String verb
int hitRate
attack(Actor attacker, Actor target, GameMap map): String
Talisman
Talisman()
allowableActions(Actor owner, GameMap map): ActionList
canAttack(Actor actor): boolean
applyStaminaCost(Actor actor)
Action
creates
TalismanCureAction
Location targetLocation
TalismanCureAction(Location targetLocation)
execute(Actor actor, GameMap map): String
Responsibility: A special weapon that can both attack enemies and cure rot from creatures and the environment. Uses stamina when attacking.
cures
cures
Creature System
Rottable
resetRotTimer()
isCurable(): boolean
Environment System
CurableGround
cure(): Ground
isCurable(): boolean
Responsibility: Represents the action of using the Talisman to cure rot. Can target both ground tiles and actors that implement the curable interfaces.