

Requirement 2



Growth logic is handled by a composite structure,

- the Tree component interface which is the common interface for both Branch and Leaf, promoting polymorphism and recursion.
- Branch class implements tree component and stores a list of children. On grow(). It randomly grows a branch or a leaf and recursively computes the attack bonus
- Leaf class implements tree component and a terminal node in tree structure. Adds 1 to the attack bonus and heals boss 5 HP

ADVANTAGES	DISADVANTAGES
Recursive and Scalable growth	Challenging to debug recursion
Clean separation of Growth types	Requires memory management
Easy to compute attack bonus	More complex than normal lists

Design 2: Action Design Encapsulation commands

- Attack action calculates the bonus damage adding the recursive attack bonus to base damage and calls hurt and unconscious on the player
- Grow action calls growSelf() on the boss and triggers recursive tree growth from the root

ADVANTAGES	DISADVANTAGES
Separates combat and growth logic	More classes were created can cause messy coding
Utilises command pattern	

Design 3: Boss Interface Growable abstraction

This allows the boss to be interacted with in a decoupled way due to its behaviour and actions

- Growable Interface
 - Defines growSelf()
 - Implemented by BedOfChaos as a private variable
 - Enables growth without the tight coupling between the boss implementation itself

ADVANTAGES	DISADVANTAGES
Supports Dependency inversion	Extra interface
Decouples behaviour/action from boss logic	More complex

ALTERNATE DESIGN STRATEGIES

DESIGN OPTION	ADVANTAGES	DISADVANTAGES
Composite tree using tree component	Recursive, scalable, clean	More complex
Direct list of components	Simpler traversal	Harder to manage and violates SRP
Growth in bed of chaos class	More centralised control	Not reusable, harder to manage and Violates SRP