# Plant Nursery Simulator

Creational Patterns

Factory Method – Plant Creation

**Intent:** Create plants without specifying the exact class

**Relations:** Works with **State** (plants start with a default state) and **Strategy** assigns care strategies

A diagram of a plant

AI-generated content may be incorrect.

**Note:**

New plants immediately enter State

Plants also get a Strategy

AbstractFactory – Related object families (plant types with their care products)

**Intent:** Create families of related objects

**Relations:** Complements **Factory Method** for more complex initialization

Works with **Factory Method** and **Strategy** to ensure proper care setup

Singleton - Garden area management

**Intent:** Ensure a class only has one instance and provide a global point of access to it.

**Relations:** Works with **Composite** (sections inside garden), accessed by **Facade**.

Provides plant data to **Iterator**, **Observer**, and **Template Method Reports**

A blue rectangular box with black text

AI-generated content may be incorrect.

Structural Patterns

Composite - Garden section hierarchy

**Composite is used because:** It lets the program **treat every part (section, bed, or plant)** the same way:  
 display(), add(), or remove() can be called on any node

Think of your nursery (or simulation garden) as a **real garden layout** — divided into **sections** and **beds**, each with specific plants.

**1. GardenSection**

* A **section** is like a **larger grouping or region** in your garden.
* Example: “North Section”, “Herb Section”, “Flower Section”.
* It can contain:
  + Other **subsections**, or
  + Several **PlantBeds**.

**2. PlantBed**

* A **plant bed** is a **smaller area** inside a section where **specific plants are placed**.
* It’s a **leaf node** in the Composite pattern — meaning it cannot contain other components

**How They Work Together**

1. **GardenArea (Singleton)** holds the **root composite** of the garden.
2. Each **GardenSection** can contain **subsections** or **PlantBeds**.
3. Each **PlantBed** can hold a **Plant** object.
4. Any system component (like SimulationFacade, Inventory, or Report) can access this unified structure through GardenArea.getInstance()

A diagram of a garden component

AI-generated content may be incorrect.

Placed inside **Singleton** GardenArea.

Traversed via **Iterator** and reported on with **Template Method Reports**.

Decorator - Plant enhancements (potted, labeled, gift-wrapped)

**Intent:** Attach additional responsibilities dynamically (wrap objects)  
**Connections:**

* Wraps around Plant objects created by **Factory Method**.
* Used before **Command SellPlant** or **Customer purchase**.

A diagram of a plant

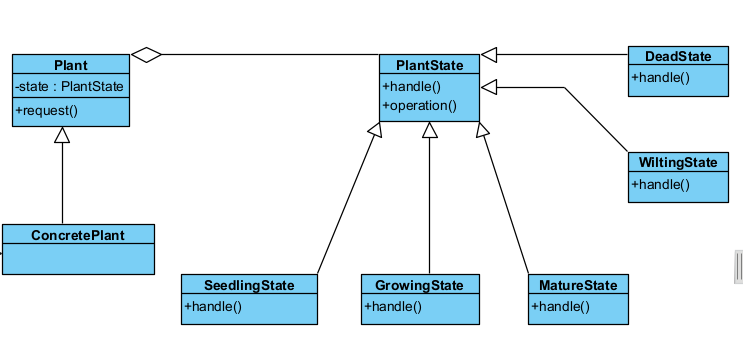
AI-generated content may be incorrect.

Behavioural Patterns

State - Plant health states

**Intent:** Allow object to change its behaviour when its internal state changes

**Relations:** Observed by Observer and influenced by Strategy



Strategy – Plant care strategies

**Intent:** Define family of algorithms, encapsulate, and swap them at runtime