# So sánh hai cách tổ chức Unit trong game Turn-Based

Mục lục

VỀ UNITMODEL

[So sánh hai cách tổ chức Unit trong game Turn-Based 1](#_Toc190212437)

[1. Cách 1: UnitBase → UnitModel (Composition - Thành phần dữ liệu) 2](#_Toc190212438)

[2. Cách 2: UnitModel → [UnitDragonModel, UnitDogModel] (Inheritance - Kế thừa) 3](#_Toc190212439)

[3. So sánh chi tiết 4](#_Toc190212440)

[4. Kết luận 5](#_Toc190212441)

## 1. Cách 1: UnitBase → UnitModel (Composition - Thành phần dữ liệu)

**Cấu trúc**

* UnitBase (**Resource**) chứa thông tin chung của loài (stats cơ bản, tên, v.v.).
* UnitModel (**Node**) sử dụng UnitBase, có thể có nhiều instance của cùng một loài.

**✅ Ưu điểm**

* **Tái sử dụng dữ liệu tốt**: Một UnitBase có thể dùng cho nhiều unit cùng loài.
* **Dễ quản lý và chỉnh sửa**: Thay đổi **stats, hình ảnh, kỹ năng** trong Godot Editor.
* **Giảm số lượng script**: Chỉ cần một UnitModel.gd, không cần class riêng.
* **Dễ mở rộng**: Thêm unit mới chỉ cần tạo **Resource .tres**.

**❌ Nhược điểm**

* **Hạn chế khi cần logic riêng cho từng loài**: Phải kiểm tra type (if unit\_base.name == "Dragon" then ...).
* **Có thể cần Dictionary để lưu skill**.

## 2. Cách 2: UnitModel → [UnitDragonModel, UnitDogModel] (Inheritance - Kế thừa)

**Cấu trúc**

* UnitModel (**Lớp cha**) chứa logic chung.
* UnitDragonModel, UnitDogModel kế thừa từ UnitModel, override phương thức hoặc thêm thuộc tính riêng.

**✅ Ưu điểm**

* **Tạo logic riêng dễ dàng**: Mỗi unit có **AI, kỹ năng, animation riêng**.
* **Dễ quản lý animation, hành vi**: Đặt trực tiếp trong UnitDragon.tscn.
* **Mô hình dễ đọc nếu có ít loại unit**.

**❌ Nhược điểm**

* **Tăng số lượng script**: 50 loại unit → 50 script.
* **Trùng lặp code**: Nếu logic giống nhau (chỉ khác skill), phải duplicate nhiều code.
* **Khó chỉnh sửa trong Godot Editor**.

## 3. So sánh chi tiết

|  |  |  |
| --- | --- | --- |
| Tiêu chí | UnitBase → UnitModel (Composition) | UnitModel → [UnitDragonModel, UnitDogModel] (Inheritance) |
| Quản lý dữ liệu | Tốt, ít trùng lặp, dễ chỉnh sửa | Kém, có thể trùng lặp dữ liệu |
| Tái sử dụng | Cao, chỉ cần thay unit\_base | Thấp, mỗi loại unit cần class riêng |
| Dễ chỉnh sửa trong Editor | Cao, chỉnh sửa `.tres` dễ dàng | Trung bình, mỗi loại có Scene riêng |
| Mở rộng số lượng Unit | Dễ dàng, chỉ cần thêm `.tres` | Khó, phải tạo nhiều class |
| Tạo hành vi riêng cho Unit | Khó, cần kiểm tra type | Dễ, override method |
| Số lượng script | Ít, chỉ cần UnitModel.gd | Nhiều, mỗi unit cần một script |

## 4. Kết luận

* 🔹 **Nếu game có nhiều Unit, muốn chỉnh sửa trong Editor** → **Chọn Cách 1** (UnitBase → UnitModel).
* 🔹 **Nếu mỗi Unit có hành vi, skill riêng** → **Chọn Cách 2** (UnitModel → [UnitDragonModel, UnitDogModel]).
* 🔹 **Kết hợp cả hai:**
  1. **Dùng Cách 1 cho unit thông thường**.
  2. **Dùng Cách 2 cho unit đặc biệt**.