**State Pattern:**

Khi mỗi trạng thái (chức vụ) **có hành vi phức tạp riêng.**Khi cần **dễ dàng thay đổi trạng thái trong runtime.  
Khi một nhân viên chỉ có một chức vụ duy nhất.**

Dùng cho các trạng thái ngang bằng với nhau (vd: table < free🡪ordered🡪fixing🡪free> )

➡ **Ví dụ:** Một ứng dụng có nhiều loại nhân viên với hành vi riêng biệt (VD: Nhân viên thường, Quản lý, Giám đốc, mỗi loại có các đặc quyền khác nhau).

**Decorator Pattern:**

Khi một nhân viên có thể có nhiều loại phụ cấp cùng lúc.  
Khi muốn tăng tính linh hoạt trong việc thêm/bớt phụ cấp mà không sửa NhanVien.  
Khi cần tách biệt từng loại phụ cấp để dễ quản lý.

Bài toán nhiệm vụ cơ bản + nhiệm vụ theo chức vụ add thêm có thể thay đổi linh động

Bài toán phụ phí

➡ **Ví dụ:** Một nhân viên có thể vừa là Tổ trưởng, vừa có bằng Tiến sĩ, vừa là Trưởng phòng, và cần tính lương dựa trên tất cả các yếu tố này. Trưởng phòng thì vẫn là một nhân viên

### ****Strategy Pattern:****

🔹 Khi cần **một cách tính lương duy nhất tại một thời điểm**.  
🔹 Khi muốn **thay đổi cách tính lương dễ dàng mà không cần sửa NhanVien**.  
🔹 Khi không cần kết hợp nhiều phụ cấp cùng lúc.

➡ **Ví dụ:** Một nhân viên chỉ có một chức vụ chính (Nhân viên thường, Tổ trưởng, hoặc Trưởng phòng) và không có nhiều phụ cấp chồng chéo.