**Mẫu State**

1. **Thông tin mẫu:**

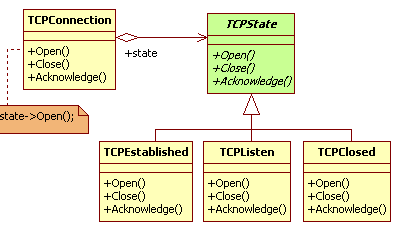
* Tên chính thức: State Pattern.
* Phân loại: Mẫu hành vi.

1. **Mục đích, ý định**

* Cho phép một đối tượng thay đổi hành vi khi trạng thái của chính nó thay đổi. Đối tượng sẽ xuất hiện để thay đổi lớp của nó.

1. **Động lực:**

* Khi bạn gặp phải một ứng dụng lớn và mã nguồn không kiểm soát nổi, điều đó thường giúp cho bạn bắt đầu suy nghĩ tới khái niệm về các trạng thái khác nhau. Đây là công cụ giúp bạn chia nhỏ mã nguồn ra thành từng đơn vị nhỏ hơn, lý tưởng nhất là từng trạng thái này phải độc lập với nhau, và tự động chia nhỏ mã nguồn bạn ra thành từng phần rời rạc.
* Một ví dụ về trường hợp ta có thê áp dụng mẫu State. Giả sử ta có một lớp TCPConnection thể hiện một kết nối mạng. Một đối tượng TCPConnection có thể có các trạng thái khác nhau: Đã kết nối, Đang lắng nghe, Đã đóng. Nó phản hồi lại các yêu cầu từ các đối tượng khác tùy theo trạng thái hiện tại của nó.



1. **Khi nào ứng dụng:**

* Một đối tượng có hành vi phụ thuộc vào trạng thái của nó, và nó phải thay đổi hành vi của mình tại thời điểm runtime tùy thuộc vào trạng thái đó.
* Các hoạt động có quá nhiều các câu lệnh điều kiện phụ thuộc vào trạng thái các đối tượng mà hoạt động đó xử lý.

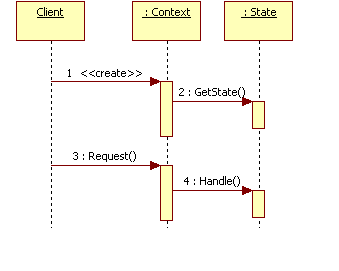
1. **Cấu trúc:**



1. **Các thành viên và mối quan hệ**

* **Context:**
  + Định nghĩa giao diện nơi cần quan tâm đến các trạng thái
  + Duy trì một thể hiện của một lớp con ConcreteState định nghĩa cho trạng thái hiện tại của **Context**.
* **State:**
  + Định nghĩa một giao diện cho việc đóng gói một hành vi gắn liền với một trạng thái cụ thể của **Context.**
* **ConcreteState**:
  + Mỗi lớp con A và B hiện thực một hành vi cụ thể cần thực hiện ứng với một trạng thái cụ thể của **Context**.

1. **Sự cộng tác:**

****

1. **Các hệ quả mang lại:**

* Bản địa hóa các hành vi khi có trạng thái cụ thể và phân vùng hành vi cho các trạng thái khác nhau:
  + Đặt tất cả các hành vi tương ứng vào một trạng thái cụ thể vào một đối tượng có kiểu State.
  + Bởi vì tất cả các đoạn code phụ thuộc vào trạng thái sống trong các lớp State con, các trạng thái mới và sự chuyển đổi trạng thái có thể dễ dàng thêm vào bằng cách định nghĩa một lớp State con mới.
* Làm tường minh sự chuyển đổi trạng thái:
  + Khi một đối tượng định nghĩa trạng thái hiện tại chỉ bằng giá trị dữ liệu nội bộ, việc chuyển đổi trạng thái của nó là không tường minh.
  + Bằng cách sử dụng các đối tượng State khác nhau, ta đã làm cho quá trình chuyển đổi của một đối tượng sử dụng đối tượng State rõ ràng hơn.
* Các đối tượng State có thể được chia sẻ:
  + Nếu các đối tượng trạng thái không có các biến thể hiện, nghĩa là trạng thái mà chúng thể hiện chính là kiểu của chúng – khi đó chúng có thể được sử dụng chung.
  + Khi một đối tượng State được chia sẻ theo cách này, về cơ bản chúng là flyweight không có các trạng thái nội tại, chỉ có các hành vi.

1. **Lưu ý cài đặt**
2. **Hệ thống liên quan:**

* Các game sử dụng State để quản lý đối tượng game và các trạng thái của chúng.

1. **Mẫu liên quan**

* Mẫu Flyweight giải thích khi nào và bằng cách nào một đối tượng State có thể được chia sẻ.
* Các đối tượng State thường là các Singleton.

1. **Demo**