**DESIGN PATTERN ADAPTER (CLASS)**

# Giới thiệu Adapter Class Design Pattern

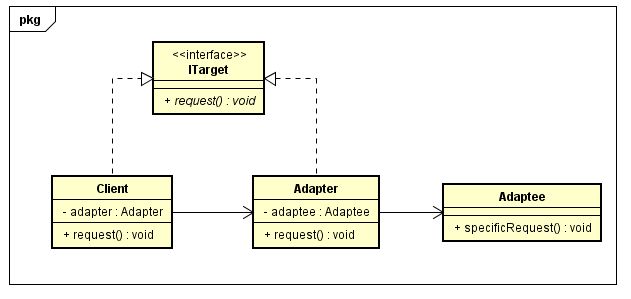
* Thuộc loại Structural Pattern
* Cho phép 2 interfaces không tương thích có thể kết nối với nhau.
* Được sử dụng khi 1 lớp implement interface mà mong đợi có thêm phương thức đặc biệt khác.

Ví dụ thực tế: Với dây cắm nồi cơm, thường là sẽ có 2 chân, nhưng để ý sẽ thấy dây cắm đó có 1 chân k được dùng. Trường hợp có 3 chân được dùng là để tương thích với ổ cắm.



# Phân tích Thiết kế Adapter

Hình ảnh dưới đây minh họa thiết kế Adapter Class trong UML.



* Client Class implement interface ITarget, vì thế mà Client có phương thức request().
* Theo yêu cầu, Client muốn có thêm phương thức specificRequest() của Adaptee Class
* Để làm được việc đó thì ta tạo thêm Class Adapter
* Class Adapter sẽ implement interface ITarget, vì thế mà Class Adapter cũng như Class Client sẽ có phương thức request(), nhưng ta sẽ thay phương thức request() bằng phương thức có chức năng mới. Ở đây là specificRequest().
* Như vậy, trong Class Client thì mình sẽ sử dụng class Adapter.
* Nếu như yêu cầu của Client là yêu cầu bình thường(request()) thì trong phương thức request() của class Client ta sẽ làm them yêu cầu bình thường.
* Còn nếu yêu cầu là specificRequest() thì đối tượng của class Adapter được sinh ra và thực hiện phương thức request của class Adapter (tương đương với specificRequest() của class Adaptee)

# Ưu điểm

* Thiết kế linh hoạt hơn
* Có thể sử dụng các giao diện khác nhau
* Hoàn toàn kết nối 2 giao diện không tương thích
* Thể hiện tính đa hình
* Tính tái sử dụng
* Nguyên tắc Open-Closed Principles

# Nhược điểm

* Phương thức của interface có thể không được áp dụng
* Nhiều khi có thể có 1 chuỗi Adapter Class để thực hiện chức năng yêu cầu

# So sánh với pattern khác

* Adapter chuyển đổi interface sang interface khác. Decorator không thay đổi giao diện nhưng bổ sung trách nhiệm. Façade làm cho interface đơn giản hơn
* Decorator hỗ trợ thành phần đệ quy, điều mà Adapter không thể thực hiện được.
* Adapter cho phép Client sử dụng thư viện và tập hợp con mà không cần thay đổi code. Decorator cho phép hành vi mới được thêm vào các lớp.
* Adapter làm cho mọi thứ hoạt động sau khi được thiết kế, Bridge làm cho chúng hoạt động trước khi thiết kế.
* Bridge được thiết kế phía trước để cho sự trừu tượng và việc thực hiện khác nhau một cách độc lập. Adapter được trang bị thêm để làm cho các lớp không liên quan làm việc cùng nhau
* Adapter cung cấp 1 giao diện khác cho Client. Proxy cung cấp cùng một giao diện. Decorator cũng cấp 1 giao diện nâng cao.
* Façade định nghĩa một giao diện mới, trong khi Adapter sử dụng lại giao diện cũ