1. **Tên và phân loại:**

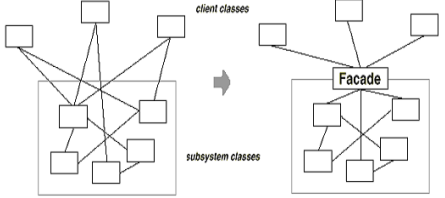
* Tên đầy đủ: Façade Pattern
* Phân loại: Mẫu kiến trúc.

1. **Mục đích, ý nghĩa:**

* Façade Pattern là pattern cung cấp một giao diện chung đơn giản thay cho một nhóm các giao diện có trong một hệ thống con (subsystem).
* Façade Pattern định nghĩa một giao diện ở một cấp độ cao hơn để giúp cho người dùng có thể dễ dàng sử dụng hệ thống con này vì chỉ cần giao tiếp với một giao diện chung duy nhất.

1. **Động lực sử dụng:**

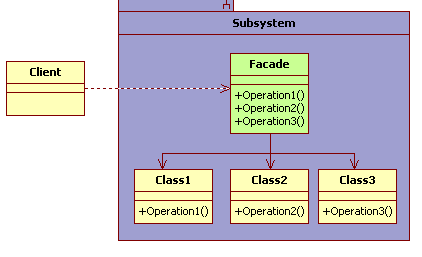
* Cơ cấu một hệ thống thành các hệ thống con nhằm giúp giảm độ phức tạp.
* Giao diện bên ngoài của các lớp trong một hệ thống con hoặc một tập các hệ thống con có thể trở nên khá phức tạp.
* Khi đó có một cách để giảm thiểu tính phức tạp này là hiện thực một đối tượng façade chịu trách nhiệm cung cấp một giao diện đơn giản hơn biểu diễn cho các chức năng chung của một hệ thống con.



1. **Khi nào sử dụng:**

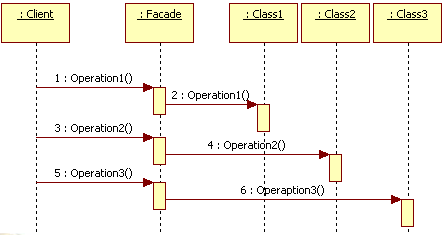
* Bạn muốn cung cấp một giao diện đơn giản hơn cho một hệ thống con phức tạp:
  + Các hệ thống con càng phát triển theo thời gian thì chúng càng trơ nên phức tạp
  + Nhiều mẫu khi được áp dụng thường để lại kết quả là có nhiều và rất nhiều những lớp nhỏ hơn.
  + Cung cấp một góc nhìn mặc định đơn giản hơn cho một hệ thống con và đủ dùng cho hầu hết các client. Chỉ có các client cần nhiều khả năng chỉnh sửa (customizability) hơn mới cần phải nhìn vượt ra khởi façade.
* Có quá nhiều phụ thuộc giữa các client và sự hiện thực các class của một trừu tượng nào đó.
  + Façade giúp decouple hệ thống con ra khỏi client và các hệ thống con khác, nâng cấp khả năng độc lập và tính di động (portability) của hệ thống con.

1. **Cấu trúc**



1. **Các thành viên - Sự cộng tác**

* **Façade** 
  + Biết các lớp của hệ thống con nào chịu trách nhiệm cho một request.
  + Ủy thác các request của client đến các đối tượng của hệ thống con thích hợp.
* **Subsystem classes**:
  + Thực thi chức năng của hệ thống con.
  + Xử lý công việc được gán bởi đối tượng Façade.
  + Không cần biết gì về façade và không giữ tham chiếu đến nó.



1. **Các hệ quả mang lại:**

* Bảo vệ client khỏi các thành phần của hệ thống con, bằng cách đó nó làm giảm số lượng đối tượng mà client cần giải quyết và làm cho hệ thống con trở nên dễ sử dụng hơn.
* Làm giảm coupling giữa hệ thống con và client của nó:
  + Chúng ta có thể sửa đổi các lớp trong hệ thống con mà ko làm ảnh hưởng đến các client sử dụng nó.
  + Giúp tách lớp một hệ thống và sự phụ thuộc giữa các đối tượng. Chúng có thể loại bỏ sự phức tạp hay phụ thuộc vòn tròn (circular dependencies). Đây là một hệ quả rất quan trọng bởi vì client và hệ thống con được hiện thực một cách độc lập.
* Không hề ngăn cản ứng dụng sử dụng các lớp của hệ thống con nếu chúng cần. Do đó ta có thể thoải mái lựa chọn giữa việc dùng façade (giao diện sử dụng trở nên đơn giản hơn) hoặc không dùng façade.

1. **Chú ý liên quan đến cài đặt:**
2. **Hệ thống thực tế**

* Các thư viện API cung cấp cho người dùng là một ví dụ.

1. **Mẫu liên quan**

* Abstract Factory:
  + Có thể được sử dụng cùng với Facade đẻ cung cấp một giao diện cho việc tạo ra các đối tượng hệ thống con một cách độc lập với nhau. Abstract Factory cũng có thể được sử dụng như một giải pháp thay thế cho Facage nhằm ẩn đi các lớp phụ thuộc nền tảng (platform-specific).
* Mediator:
  + Tương tự như Façade, tuy nhiên mục đích chính của Mediator là trừu tượng hóa các giao tiếp truyền thông tùy ý giữa các đối tượng colleague. Giao tiếp của các đối tượng colleague phải thông qua Mediator.
  + Tuy nhiên Façade chỉ đơn giản là trừu tượng hóa giao diện của một đối tượng hệ thống con và làm cho nó dễ xài hơn mà thôi. Façade cũng không định nghĩa các tính năng mới, và các lớp của hệ thống con cũng không biết gì về nó.
* Thông thường chỉ một đối tượng Façade là đủ. Do đó các đối tượng Façade thường là các Singleton.