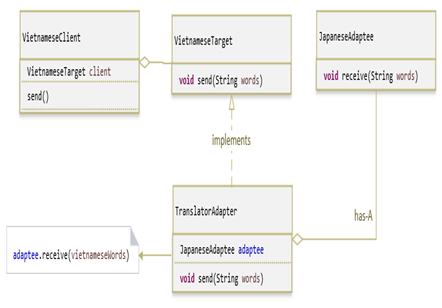
**Lab 2: Tìm hiểu và cài đặt nhóm mẫu Structural**

1. **Adapter**

Adapter pattern là loại design pattern thuộc nhóm cấu trúc (Structural). Adapter sử dụng một thành phần trung gian để chuyển đổi giao diện (interface) có sẵn thành một giao diện khác đảm bảo thích hợp với lớp đang viết. Đồng thời cho phép các đối tượng cộng tác với nhau bất kể interface không tương thích.

*Hình 0: Hình minh hoạ Adapter pattern*

**Ví dụ 1: Ứng dụng dịch ngôn ngữ Tiếng Việt sang Tiếng Nhật**

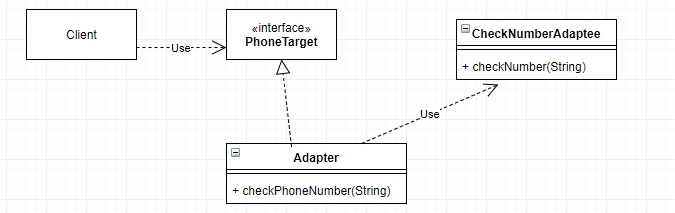


*Hình 1: Sơ đồ class ứng dụng chuyển đổi ngôn ngữ.*

Trong ví dụ này: Người việt sẽ là client và anh ta sẽ gửi một số tin nhắn cho người Nhật.

* VietnameseClient: là Client sẽ gửi một số message cho JapaneseAdaptee.
* VietnameseTarget: là nội dung message được VietnameseClient cung cấp cho thông dịch viên (TranslatorAdapter).
* TranslatorAdapter: nó sẽ là thông dịch viên và là Adapter, nó sẽ nhận message Tiếng Việt từ VietnameseClient và chuyển đổi sang tiếng Nhật để gửi cho JapaneseApdatee.
* JapaneseApdatee: đây là interface hoặc class được người nhật dùng để nhận message đã được phiên dịch từ TranslatorAdapter.

**Ví dụ 2: Ứng dụng chuyên thực hiện việc kiểm tra 1 chuỗi kí tự có phải là kiểu số “không”. Client bây giờ cần chức năng kiểm tra 1 chuỗi kí tự có phải là số điện không.**



*Hình 2: Sơ đồ class ứng dụng kiểm tra chuỗi kí tự*

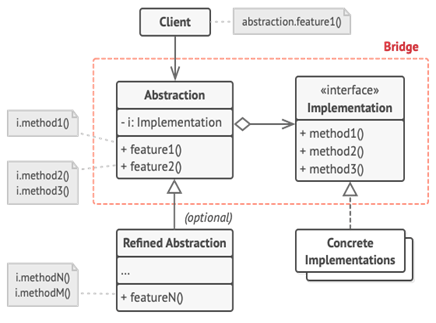
Ta có thể thấy rằng Client sẽ không thể dùng trực tiếp được. CheckNumberAdaptee mà phải thêm các class Adapter để thực hiện kiểm tra số điện thoại.

PhoneTarget: là kí tự người dùng nhập vào

Adapter sẽ lấy dữ liệu từ PhoneTarget và dùng checkNumber có class CheckNumberAdaptee để kiểm tra kí tự.

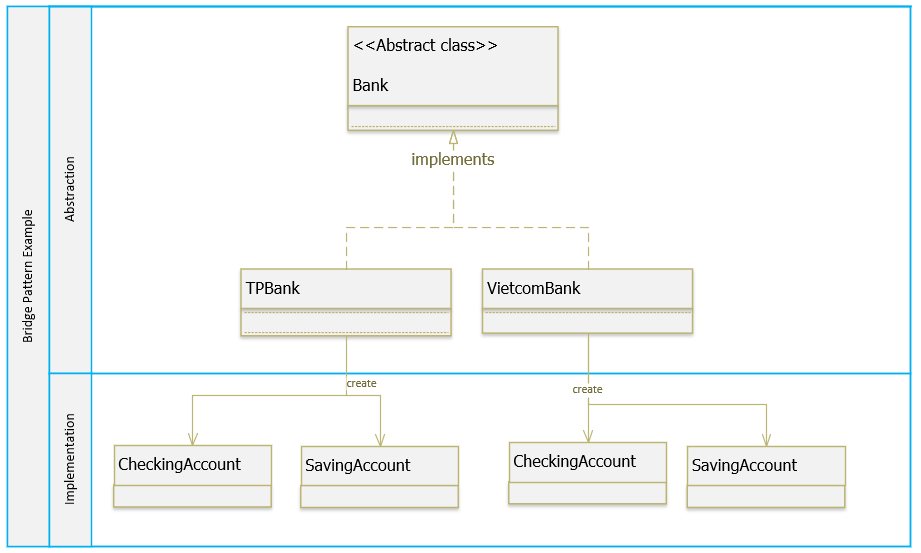
1. Bridge

Bridge Pattern là Pattern thuộc nhóm cấu trúc (Structural Pattern). Ý tưởng của nó là tách tính trừu tượng (abstraction) ra khỏi tính hiện thực (implementation) của nó. Từ đó, có thể dễ dàng chỉnh sửa hoặc thay đổi độc lập với nhau.



*Hình 3: Sơ đồ minh hoạ Bridge Pattern*

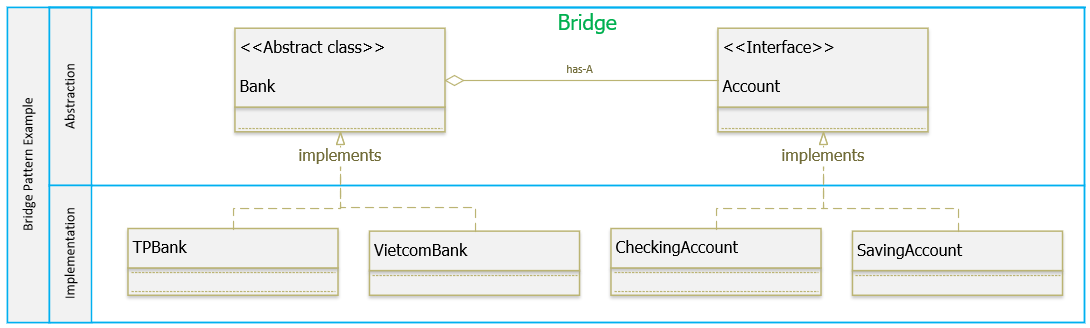
**Ví dụ 1: Một hệ thống ngân hàng cung cấp các loại tài khoản khác nhau cho khách hàng, chẳng hạn: Checking account và Saving account. Có sơ đồ sau đây:**



*Hình 4: Sơ đồ class ứng dụng hệ thống ngân hàng*

Với cách thiết kế nhưu vậy, khi hệ thống cần cung cấp thêm một loại tài khoản khác, chúng ta phải tạo class mới cho tất cả ngân hàng và số lượng class sẽ tăng lên rất nhiều.

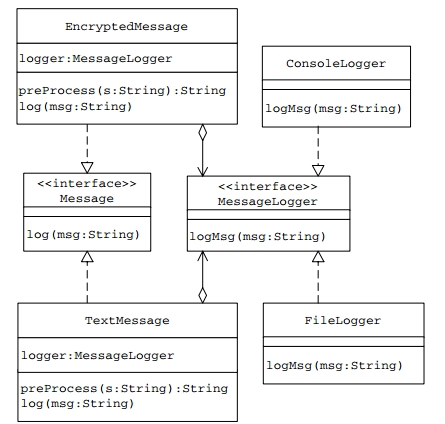
Do vậy sử dụng bridge pattern để tái cấu trúc hệ thống.



*Hình 5: Sơ đồ class ứng dụng hệ thống ngân hàng sau khi áp dụng bridge pattern*

Với cấu trúc như vậy, khi thêm tài khoản, chỉ cần implements Account và nó sẽ không ảnh hưởng tới các thành phần khác.

**Ví dụ 2: Ta có tin nhắn với 2 cách log là ghi ra file và show ra màn hình mỗi cách lại được thực hiện làm 2 kiểu là thực hiện với kết quả là text rõ hoặc text đã được mã hoá.**



*Hình 6: Sơ đồ ví dụ2 bridge pattern*

Message: là 1 lớp trừu tượng khai báo chức năng log

MessageLogger: là 1 interface thực thi chức năng log của Message

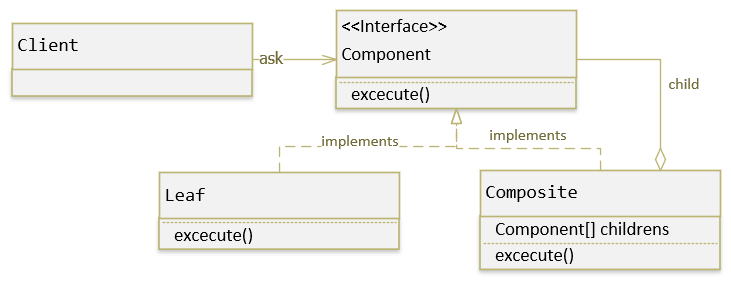
ConsoleLogger: thừa kế MessageLogger, có chức năng show message ra console

FileLogger: thừa kế MessageLogger, có chức năng ghi message vào file

TextMessage và EncryptedMessage: thừa kế Message, quyết định xem việc log tin nhắn là text rõ hoặc text đã được mã hóa.

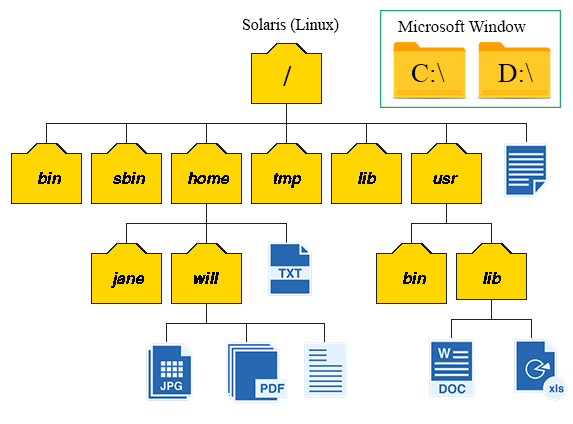
1. Composite

Composite pattern là một mẫu thiết kế thuộc nhóm cấu trúc (Structural Pattern). Dùng để tạo ra các đối tượng trong các cấu trúc cây để biểu diễn hệ thống phân lớp: bộ phận – toàn bộ. Cho phép client tác động đến từng đối tượng và các thành phần của đối tượng một cách thống nhất.



*Hình 7: Sơ đồ minh hoạ Composite*

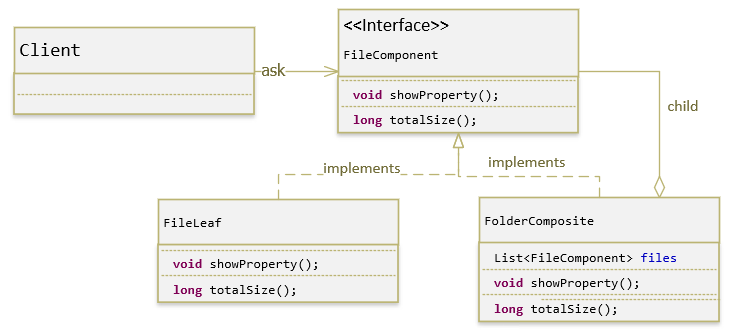
**Ví dụ 1: Chương trình quản lý một hệ thống tập tin.**



*Hình 8: Sơ đồ minh hoạ hệ thống tập tin*

Trong đó các thư mục là các Composite và các file sẽ là các Leaf. Một folder có thể chứa một hoặc nhiều file hoặc folder.

Áp dụng cài đặt Composite Pattern vào ví dụ:

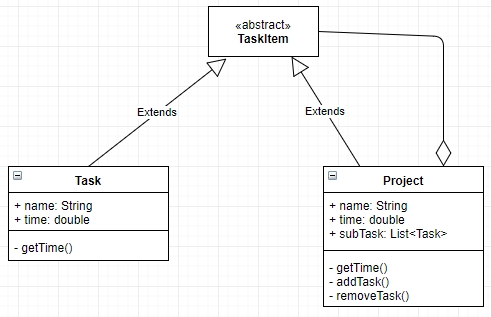


*Hình 9: Sơ đồ ứng dụng hệ thống tập tin áp dụng composite*

Trong đó:

* Client là nơi sử dụng FileComponent để làm việc với các đối tượng Composite.
* FileComponent là một interface hay abstract class quy định các thuộc tính cho FileLeaf và FolderComposite như là showProperty(hiển thị đối tượng con) và totalSize: dung lượng.
* FolderComposite: là nơi tập hợp các FileLeaf và sử dụng các phương thức từ FileComponent.
* FileLeaf: là các lớp implements từ FileComponents và nó là object không có đối tượng con.

**Ví dụ 2: 1 project là 1 tập hợp nhiều tác vụ (Task con) đồng thời bản thân project cũng là 1 task lớn. Tính tổng thời gian của project thông qua thời gian của các task con.**



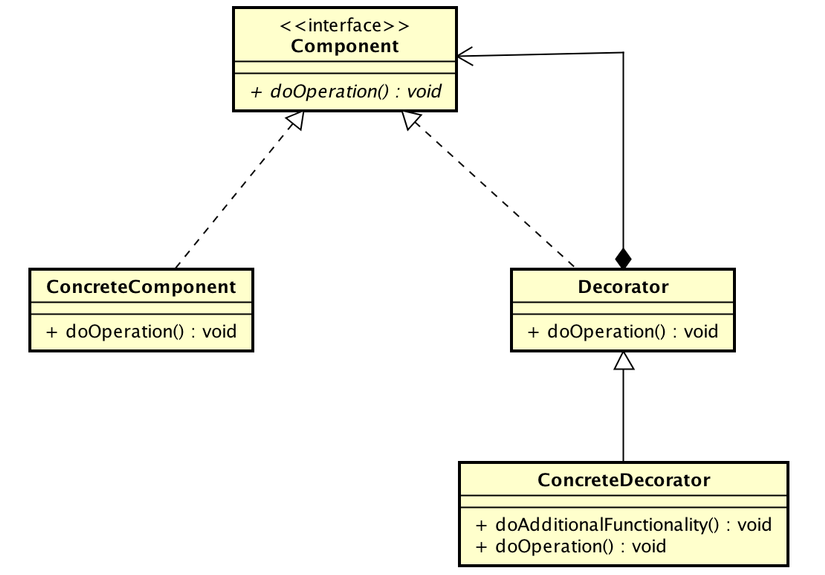
*Hình 10: sơ đồ class ví dụ 2*

Ta có:

* TaskItem là một abstract class, nó có method là getTime();
* Task sẽ implementation từ TaskItem và trả về thời gian của Task đó.
* Project nó là tập hợp nhiều Task (List<Task>) và trả về tổng thời gian của List<Tash>

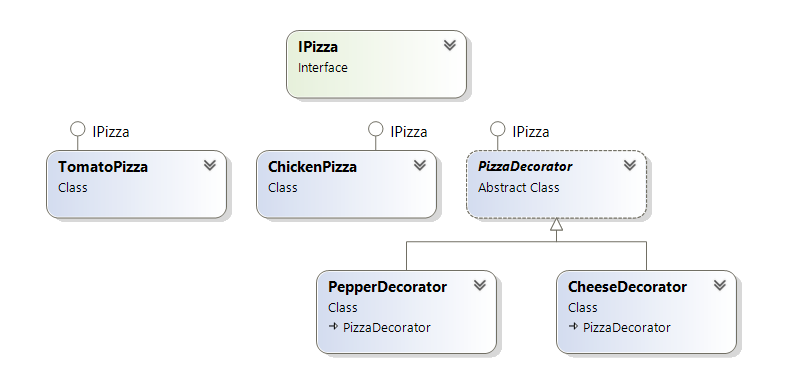
1. Decorator

Decorator pattern là một trong những Pattern thuộc nhóm cấu trúc (Structural Pattern). Cho phép người dùng thêm chức năng mới vào đối tượng hiện tại mà không muốn ảnh hưởng đến các đối tượng khác.

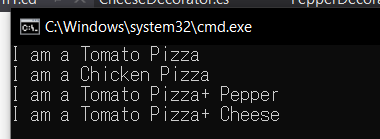


*Hình 11: Minh hoạ Decorator design pattern*

**Ví dụ 1: Cửa hàng bánh pizza, cửa hàng vừa làm pizza cà chua và pizza phô mai. Và có một số nguyên liệu nữa lên phần trên của bánh vì khách hàng có quyền lựa chọn thêm gà hoặc hồ tiêu vào chiếc bánh của họ.**



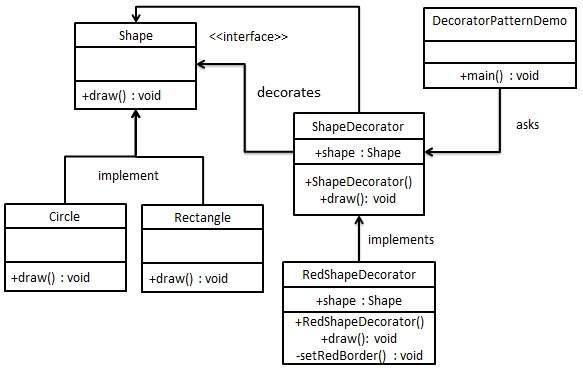
*Hình 12: Sơ đồ triển khai ứng dụng*



Trong đó:

* Giao diện IPizza là thành phần Component trong mẫu thiết kế Decorator, nó chứa phương thức doPizza, đây là phương thức dùng để tạo ra một pizza phù hợp.
* TomatoPizza và ChickenPizza là những cài đặt, triển khai của IPizza. Chúng cung cấp cụ thể các thể hiện của lớp mà chúng ta cẩn mở rộng trong quá trình chương trình đang chạy.
* PizzaDecorator là nhân vật chính của sơ đồ thiết kế trên. Nó giữ một thể hiện đã tồn tại của pizza như TomatoPizza hoặc ChickenPizza. Thuộc tính này sẽ được cài đặt thông qua phương thức cởi tạo, và nó được mở rộng trong khi chương trình chạy.
* PepperDecorator và CheeseDecorator cài đặt các phương thức mở rộng, trong trường hợp ví dụ này, PepperDecorator sẽ thêm phụ gia vào một pizza đã có. Tính năng mở rộng là được cài đặt trong phương thức addPepper().

**Ví dụ 2: Vẽ hình vuông, hình tròn và đặt màu border: màu đỏ với màu mặc định**



*Hình 13: Sơ đồ triển khai ứng dụng*

Shape là một component trong Decorator design pattenr nó là 1 interface chứa phương thức draw(), đây là phương thức vẽ hình.

Circle, Rectangle implementation Shape và overide phương thức draw() để vẽ hình theo hình dạng class đó (Circle, Rectangle)

ShapeDecorator, RedShapeDecorator là các decorator thực hiện gọi lệnh từ DecoratorPatternDemo để vẽ.

Riêng RedShapeDecorator nó có thể setRedBorder() để đặt border màu đỏ.

DecoratorPatternDemo: là Client yêu cầu ShapeDecorator thực hiện.