MSSV: 2011423

Họ và tên: Ngô Hoài Phong

Lớp: CTK44PM

**LAB01: CREATIONAL PATTERNS**

**Abstract Factory**

**Định nghĩa:**

Abstract Factory cung cấp một interface cho việc tạo lập các đối tượng có liên hệ với nhau mà không cần qui định lớp hay xác định lớp cụ thể tạo mỗi đối tượng.

**Sơ đồ lớp UML:**



Sơ đồ trên sử dụng Abstract Factory để tạo ra các thành phần của giao diện (nút bấm, nút chọn,…) cho nhiều hệ điều hành khác nhau mà không ràng buộc client với một lớp cụ thể nào, giúp cho các thành phần được tạo ra phù hợp với mỗi hệ điều hành cụ thể.

Cùng một thành phần giao diện ở các hệ điều hành khác nhau đều mong muốn có chức năng hoạt động giống nhau nhưng khác nhau một vài điểm ở giao diện tùy thuộc vào hệ điều hành.

Abstract Factory interface định nghĩa một tập hợp các phương thức tạo lập mà client có thể dùng để tạo ra các thành phần giao diện khác nhau. Các concrete factory tương ứng với các hệ điều hành cụ thể và tạo ra thành phần phù hợp với hệ điều hành đó.

Cách thức hoạt động: Khi ứng dụng được chạy, nó sẽ kiểm tra loại hệ điều hành hiện tại. Ứng dụng sẽ sử dụng thông tin đó để tạo ra đối tượng factory từ một lớp phù hợp với hệ điều hành đó. Các đoạn mã còn lại của chương trình sẽ sử dụng factory này để tạo ra các thành phần giao diện tương ứng. Điều này sẽ giúp ngăn chặn việc tạo nhầm thành phần giao diện cho từng hệ điều hành.

Điều này sẽ giúp ích trong việc mở rộng ứng dụng. Khi cần thêm một thành phần giao diện mới, không cần phải chỉnh sửa code ở client mà chỉ cần tạo ra một lớp factory cung cấp thành phần đó và chỉnh sửa phương thức khởi tạo của ứng dụng để lựa chọn lớp cho phù hợp.

**Factory Method**

**Định nghĩa:**

Factory Method sẽ định nghĩa interface để tạo ra đối tượng nhưng để cho lớp con quyết định lớp nào được dùng để tạo ra đối tượng đó. Hay nói cách khác, Factory method cho phép một lớp có thể chuyển quá trình khởi tạo đối tượng cho lớp con.

**Sơ đồ lớp UML:**



Sơ đồ trên mô tả Factory Method có thể sử dụng để tạo ra cà thành phần của giao diện đa nền tảng mà không ràng buộc client với một lớp cụ thể.

Lớp Dialog sử dụng các thành phần giao diện khác nhau để render ra cửa sổ. Tùy vào mỗi hệ điều hành, các thành phần đó sẽ khác nhau về mặt giao diện nhưng chức năng hoạt động sẽ giống nhau.

Khi sử dụng Factory Method, ta sẽ không cần viết lại logic của lớp Dialog cho mỗi hệ điều hành. Nếu ta định nghĩa một Factory Method để cung cấp các nút bấm bên trong lớp Dialog, ta có thể tạo ra một lớp con trả về nút bấm có hình dạng của hệ điều hành Windows từ Factory Method đó. Lớp con sẽ kế thừa hầu hết từ lớp cha, nhưng nhờ vào Factory Method nên có thể tạo ra một nút bấm có giao diện của hệ điều hành Windows.

**Prototype**

**Định nghĩa:**

Prototype cho phép sao chép một đối tượng có sẵn mà không làm cho các đối tượng không phụ thuộc vào các lớp của nó.

**Sơ đồ lớp UML:**



Các lớp hình học sẽ thực thi cùng một interface có chứa phương thức để sao chép. Một lớp con có thể gọi đến phương thức sao chép của lớp cha trước khi sao chép các giá trị của chính nó vào đối tượng kết quả.

**Singleton**

**Định nghĩa:**

Singleton sẽ đảm bảo một lớp chỉ có một thể hiện và cung cấp khả năng truy cập toàn cục cho thể hiện đó.

**Ví dụ:**

Trong một ứng dụng, khi các client gửi yêu cầu đến server để lấy dữ liệu từ cơ sở dữ liệu thì trên server sẽ áp dụng Singleton để tạo ra một thể hiện duy nhất để truy cập cơ sở dữ liệu. Nếu có nhiều thể hiện cùng truy cập vào cơ sở dữ liệu một lúc thì có thể gây ra một số lỗi không mong muốn, do đó đây là một trưởng hợp điển hình cần phải ứng dụng Singleton Pattern.