**Design Pattern**

* **Sơ lược về Design Pattern:**

- Design Pattern là những cái mẫu thiết kế đã được định nghĩa sẵn và thường được áp dụng trong các tình huống phổ biến, chúng sẽ giúp code chúng ta sẽ dễ bảo trì, mở rộng và tái sử dụng hơn. Design Pattern không phải là đoạn code cụ thể, thường được sử dụng trong các ngôn ngữ lập trình hướng đối tượng.

1. **Strategy Design Pattern:**

- Định nghĩa một họ các thuật toán, đóng gói từng thuật toán riêng biệt và cho chúng thay thế lẫn nhau, tách phần lựa chọn thuật toán khỏi đối tượng sử dụng chúng, làm việc thay đổi linh hoạt mà không cần phải sửa mã gốc.

*Ví dụ:* Chúng ta có đặt hàng online thì chúng ta ứng dụng phương thức giao hàng vào, chúng ta có 1 class đơn hàng trong đó có các biến tên khách, giá trị đơn hàng, giao tiết kiệm, giao nhanh...) nếu như mà ta muốn thay đổi giao tiết kiệm hoặc giao nhanh thì chúng ta phải vào lại để thay đổi và nếu như chi phí giao hàng từ giao tiết kiệm sang giao nhanh thì chúng ta cũng không thể sửa trực tiếp được vì có thể là 30.000d, 50.000d hoặc tính theo km thì chúng ta không thể sửa 1 lần. Vì thế chúng ta sẽ để cái chiến dịch giảm giá này sang bên ngoài thành 1 cái interface, khi đó thì cái cái class đơn hàng này sẽ không quan tâm đến cái việc tính phí giao hàng như thế nào nữa, lúc này nó chỉ quan tâm là nó sẽ có chiến lược giao hàng cụ thể, việc tính phí giao theo khoảng cách hay trọng lượng hay như thế nào thì phụ thuộc khách hàng.

**2. Observer Design Pattern:**  
- Định nghĩa một mẫu thiết kế thuộc nhóm hành vi để thiết lập mối quan hệ 1-n giữa các đối tượng, khi một đối tượng thay đổi trạng thái, tất cả các đối tượng phụ thuộc đều được thông báo và cập nhật tự động.

*Ví dụ:*

Subject (WeatherStation):

Lớp chứa trạng thái (nhiệt độ, độ ẩm, áp suất).

Kế thừa từ lớp Subject cơ sở, quản lý danh sách observers.

Tự động thông báo khi dữ liệu thời tiết thay đổi thông qua các setter.

Observer (lớp trừu tượng):

Định nghĩa phương thức update() mà các concrete observer phải triển khai.

Concrete Observers:

PhoneDisplay, LCDDisplay, WebDisplay, SmartHomeSystem.

Mỗi loại xử lý và hiển thị dữ liệu theo cách riêng (màn hình, hệ thống tự động, điện thoại,…).

Tự đăng ký với WeatherStation khi được tạo.

**3. Singleton Design Pattern:**  
- Định nghĩa: Đảm bảo một lớp chỉ có một thể hiện (instance) duy nhất và cung cấp một điểm truy cập toàn cục đến thể hiện đó.

*Ví dụ:* Tạo bộ sinh số ngẫu nhiên duy nhất.

* Private int index: Biến instance lưu trữ giá trị ngẫu nhiên từ 1 đến 3.
* private static final Object lockObject: là khi có một thread đang chạy hàm này thì các thread khác phải chờ hàm này chạy xong thì mới cho hàm mới, không tạo ra nhiều instance.

- Private static volatile RandomIndexGenerator uniqueInstance: Biến static volatile để lưu trữ instance duy nhất, giúp các thread nhìn thấy giá trị mới nhất.

**4. Builder Design Pattern:**

- Định nghĩa: Mẫu thiết kế đối tượng được xây dựng một đối tượng phức tạp khỏi biểu diễn của nó và tiếp cận từng bước, việc xây dựng với các đối tượng độc lập với các đối tượng khác.

*Ví dụ:*

Tạo ra 2 cái sử dụng chương trình chạy trực tiếp bằng constructor và builder để xem cách nó hoạt động.

Chạy bằng constructor thì tạo ra lớp user có các thuộc tính như name, age, email, sau đó tạo constructor đầy đủ tham số rồi tạo các getter, setter như bình thường , xong rồi thêm 1 phương thức hiển thị thông tin.  
Chạy bằng builder ta sẽ tách constructor ra thành các bước nhỏ, mỗi bước gán một thuộc tính. Cách này để tạo ra đối tượng dễ nhìn, dễ hiểu cho người đọc trong lúc sử dụng dữ liệu lớn và rất hữu ích khi đối tượng có nhiều thuộc tính.

**5. Factory Method:**  
- Định nghĩa: Cung cấp một interface để tạo ra một đối tượng trong class, nhưng để cho class con kế thừa của nó có thể ghi đè để chỉ rỏ đối tượng nào sẽ được tạo, Factory Method giao việc khởi tạo một đối tượng cụ thể cho lớp con.

*Ví dụ:* Ta sẽ tạo ra 3 lớp con Oto, XeMay, XeDap kế thừa từ lớp PhuongTien, mỗi lớp ghi đè phương thức DiChuyen(). Sau đó ta tạo interface với phương thức trả về một đối tượng cụ thể(Oto, XeMay, XeDap), tạo 2 implement cho interface mới trên 1 phương thức là random tạo phương tiện ngẫu nhiên, còn 1 cái là là tạo phương tiện theo thứ tự (Oto -> XeMay -> XeDap).

Link code:  [https://github.com/NguyenThiNhuQuynh-Hub/NguyenThiNhuQuynh-](https://github.com/nvanchien-1/design-pattern)