ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

**REPORT**

**Nhóm 7:**

18127014 – Huỳnh Nhật Nam

18127223 – Nguyễn Phúc Thịnh

18127208 – Ngô Thanh Phương Thái

**Lớp:** 18CLC

**Môn : Design Patterns**

Thành phố Hồ Chí Minh năm 2021

# SƠ ĐỒ LỚP

## Sơ đồ lớp:

### Tổng:

A picture containing timeline

Description automatically generated

## Giải thích:

### Các lớp validator:

#### Interface:

* **Validator:** là interface chung cho tất cả các validator. Sử dụng Chain Of Responsibility pattern. Tất cả các validator sẽ validate 1 object thuộc Type T (generic)

#### Các lớp cơ bản:

* **BaseValidator:** lớp validator cơ bản. Lớp này đã implement một số chức năng cơ bản của một validator như setNext(), hasNext(),… Tất cả các validator đều sử dụng lớp này
* **BuiltInValidator:** là lớp để xây dựng một lớp validator dễ dàng. Lớp này đã sử dụng template method để xử lý chung việc ghi nhận, trả về kết quả và truyền tiếp tới validator kế tiếp trong chuỗi. Các lớp Validator kế thừa lớp này chỉ cần implement một số hàm đơn giản để hoạt động (cụ thể là 2 hàm: tạo default failed message v

#### Các lớp xử lý một lớp phức tạp do người dùng tạo ra:

* **AbstractValidator:** là công cụ để người dùng xây dựng một validator cho một Object không thuộc các kiểu dữ liệu cơ bản của java như String, boolean, … Người dùng sẽ kế thừa class này để xây dựng một validator cho một Object. Class này có các hàm đã được cài sẵn để xây dựng một cách dễ dàng các luật cho các trường dữ liệu của Class cần được validated
* **Rule:** Về cơ bản, Rule là một validator của một lớp T để validate một trường (một biến) S của lớp đó. Do đó, nó sẽ có hàm để lấy giá trị của trường đó và một chuỗi validator cho trường đó. Khi Rule được yêu cầu validate một object của lớp T, nó sẽ lấy giá trị của trường S và truyền vào chuỗi validator.

#### Built-in validators: kế thừa BultInValidator

* **MinValidator:** So sánh giá trị đưa vào lớn hơn giá trị cho trước (bằng hoặc không bằng)
* **MaxValidator:** So sánh giá trị đưa vào nhỏ hơn giá trị cho trước (bằng hoặc không bằng)
* **EqualValidator:** So sánh giá trị đưa vào bằng giá trị cho trước
* **NotNullValidator:** Kiểm tra giá trị đưa vào khác null
* **NotEmptyValidator:** Kiểm tra giá trị string đưa vào không rỗng
* **RegexValidator:** Kiểm tra giá trị string đưa vào trùng với regex
* **AssertTrueValidator:** Kiểm tra giá trị boolean đưa vào true
* **AssertFalseValidator:** Kiểm tra giá trị boolean đưa vào false

#### Các lớp validator xử lý Annotation:

* **AnnotationValidator:** validate một class theo ràng buộc dữ liệu (annotation). Khi validate một object lần đầu sẽ tạo chain AnnotatedFieldValidator cho từng trường của class đó
* **AnnotatedFieldValidator:** validate một trường của class theo ràng buộc dữ liệu. sử dụng builder tùy theo field

### Các lớp Builder:

### Comparators:

### Các lớp Result

### Các lớp Helpers:

# CÁC MẪU THIẾT KẾ SỬ DỤNG

## Chain Of Responsibility:

### Sơ đồ lớp:

### Đoạn code

### Ý nghĩa:

* Sử dụng mẫu chain of responsibility để ta có thể liên kết bất kì số lượng validator có thứ tự cho một kiểu dữ liệu ở runtime.
* Mỗi validator không cần biết về chuỗi mà chỉ cần thực hiện validate, ghi kết quả và quyết định việc nên truyền cho validator tiếp theo hay nên terminate chuỗi khi việc validate thất bại.Điều này khiến cho việc thêm một validator dễ dàng hơn rất nhiều và sẽ không ảnh hưởng tới các validator khác đã được cài đặt.

## Builder:

### Sơ đồ lớp:

### Đoạn code

### Ý nghĩa:

* Sử dụng Builder để có thể tạo chuỗi các validator phức tạp một cách hợp lý và tuần tự từng bước
* Sử dụng builder, ta có thể thêm vào chuỗi từng validator một

## Factory Method:

### Sơ đồ lớp:

### Đoạn code

### Ý nghĩa:

## Strategy:

### Sơ đồ lớp:

### Đoạn code

### Ý nghĩa:

## Template Method:

### Sơ đồ lớp:

### Đoạn code

### Ý nghĩa: