**Báo cáo đồ án**

**MẪU THIẾT KẾ HƯỚNG ĐỐI TƯỢNG**

VALIDATION FRAMEWORK

**Nhóm sinh viên:**

1512569 - Nguyễn Đình Tiến

1512557- Phan Trọng Thuyên

1512559 – Nguyễn Hồ Toàn Thư

**Giáo viên thực hành:**

Mai Anh Tuấn

**Năm học: 2018 - 2019**

C:\Users\tdqua_000\Dropbox\SS-Slides\DeCuong-CDIO\Template CDIO v4.2\Templates\Hinh anh\LogoTruong.png

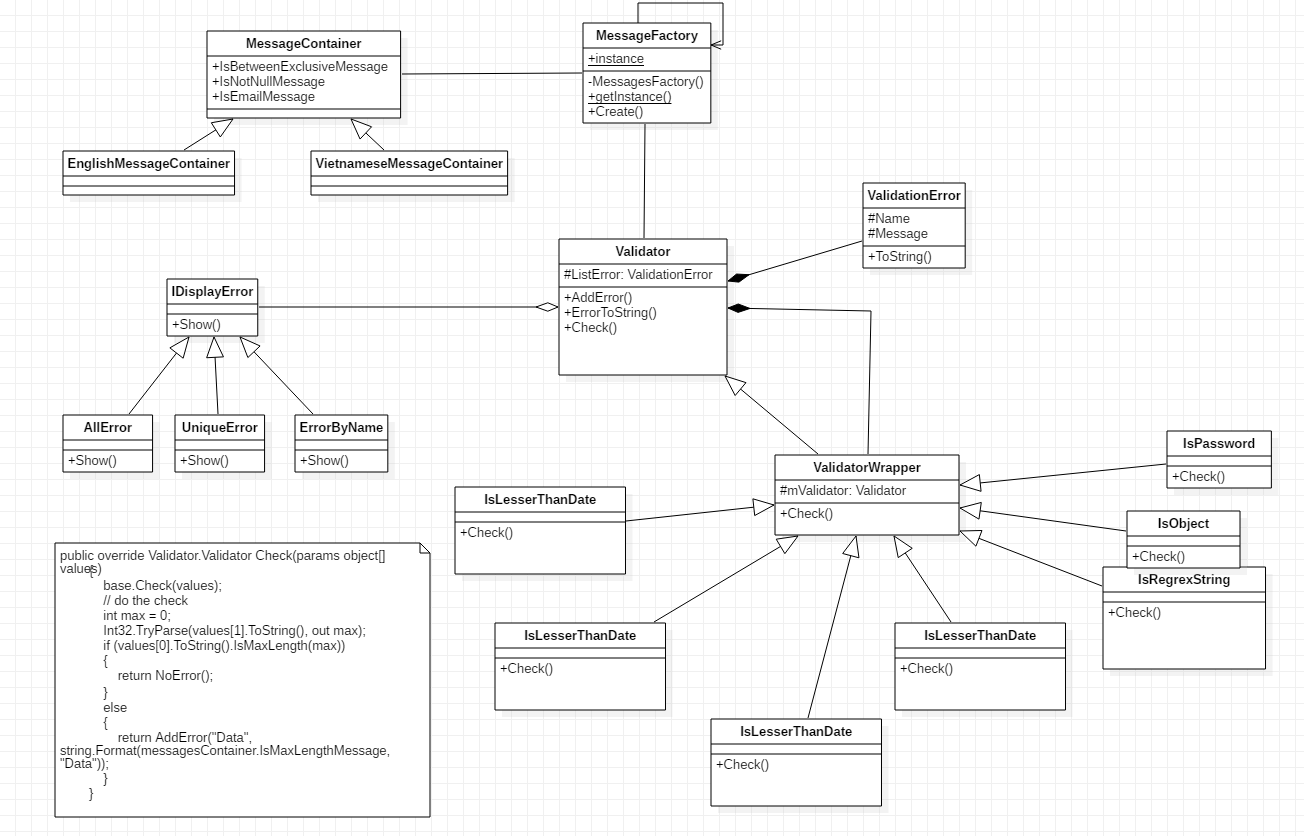
Bộ môn Công nghệ phần mềm

Khoa Công nghệ thông tin

Đại học Khoa học tự nhiên TP HCM

1. **Giới thiệu**

* MyValidate là một thư viện validation đơn giản, được viết bằng C#.
* MyValidate sử dụng một số mẫu thiết kế cơ bản như Factory, Decorator, Strategy, Template, Singleton.

1. **Sơ đồ lớp**
2. Sơ đồ lớp toàn bộ Framework
3. Ý nghĩa của từng lớp

**MessageContainer**

Đây là lớp abstract chứa các attribute là các biến có kiểu String là nội dung thông điệp để hiển thị cho người dùng.

Ý nghĩa của lớp là tạo một giao diện chung cho cách hiển thị thông điệp theo ngôn ngữ, tùy vào người dùng lựa chọn mà có thể linh động hiển thị theo ý của người dùng.

**EnglishMessageContainer**

Đây là lớp con của MessageContainer, mô tả nội dung thông điệp hiển thị bằng Tiếng Anh.

**VietnameseMessageContainer** tương tự EnglishMessageContainer tuy nhiên nó mô tả thông điệp bằng Tiếng Việt.

**MessageFactory** là lớp có nhiệm vụ nhận yêu cầu tạo lập từ người dùng, sẽ tạo ra đối tượng của lớp **MessageContainer** tương ứng với kiểu mà người sử dụng truyền vào. Người dùng không cần trực tiếp tạo ra đối tượng thuộc kiểu cách hiển thị nào mà chỉ cần gọi hàm Create(<en, vi>).

**IDisplayError** là một abstract class là giao diện chung cho cách hiển thị thông điệp.

**AllError** với phương thức show có nhiệm vụ hiển thị tất cả error của tất cả các trường đang kiểm tra.

**UniqueError** với phương thức show được override từ IDisplayError hiển thị error phân biệt từng trường, giả sử một trường không thỏa hai điều kiện thì nó chỉ hiển thị ra điều kiện cuối cùng.

**ErrorByName** phương thức show sẽ hiển thị theo tên trường được truyền vào, sẽ chỉ hiển thị lỗi của trường đó.

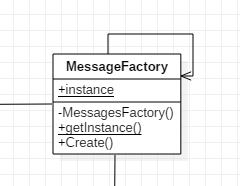
**ValidationError** là lớp đặc tả tên thông điệp và nội dung của thông điệp đó, đây như là một cấu trúc đầy đủ của một thông điệp.

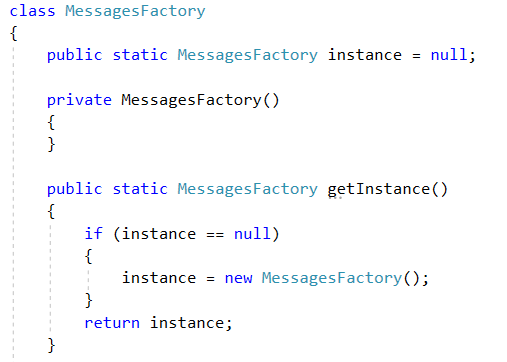
**Validator** lớp này chứa danh sách các thông báo lỗi, là nơi khởi tạo factory để tạo ngôn ngữ hiển thị cho thông báo. Đồng thời nó cũng làm nhiệm vụ thêm các thông báo lỗi khi data không thỏa mãn yêu cầu. Đây là một interface cho ValidatorWrapper, interface này sử dụng các thành phần đệ quycho phép một số lượng layer không giới hạn để add vào core validator.

**ValidatorWrapper** lớp này sinh ra để đóng gói đối tượng gốc là Validator vào trong một abstract wrapper interface.

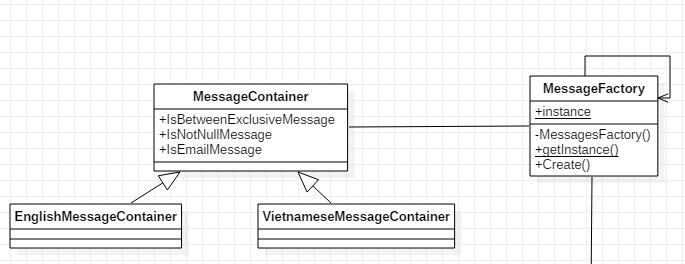
**DatetimeValidator,** **StringValidator, NumberValidator, ObjectValidator** có các phương thức dùng để kiểm tra điều kiện mà người dùng truyền vào

1. Các mẫu sử dụng trong đồ án
2. Singleton
   * + Sơ đồ lớp ứng với mẫu Singleton

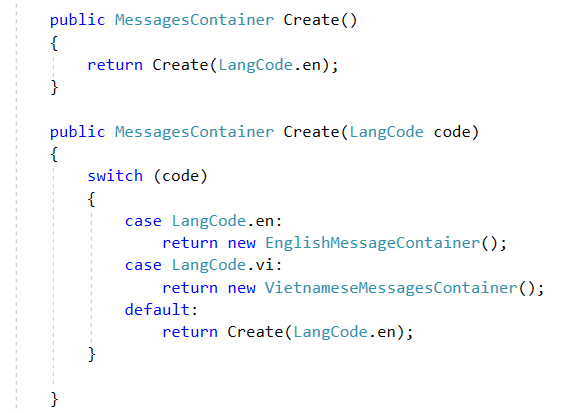


* + - Đoạn code sử dụng
    - Ý nghĩa: Trong giới hạn project thì ta chỉ cần một factory để tạo message, do đó để tránh tình trạng lãng phí tài nguyên không cần thiết thì áp dụng mẫu Singleton vào MessageFactory là cần thiết.

1. Factory
   * + Sơ đồ lớp



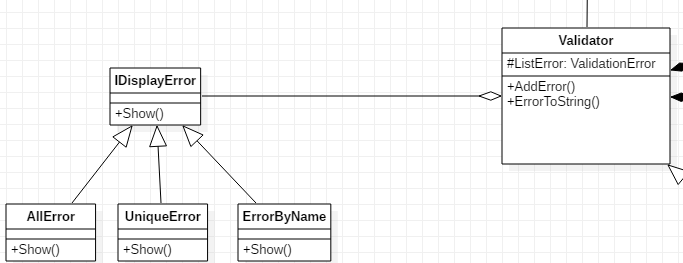
* + - Đoạn code sử dụng:



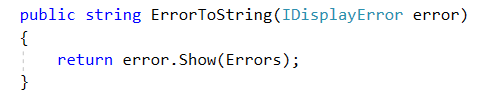


* + - Ý nghĩa: che giấu sự khởi tạo của đối tượng, người dùng sẽ không thấy cách mà đối tượng tạo ra.

1. Strategy

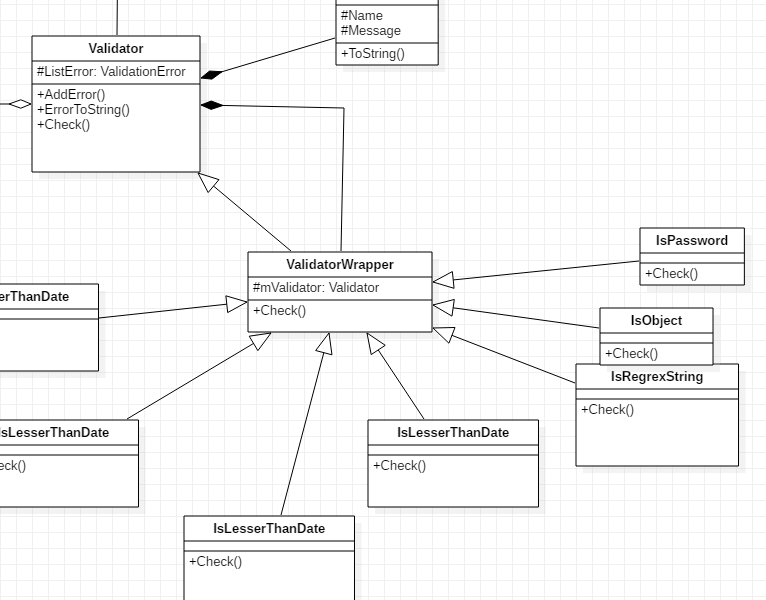


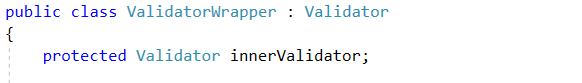
* + - Đoạn code sử dụng

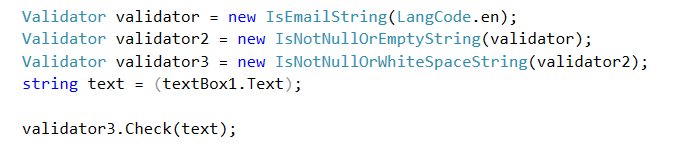


* + - Ý nghĩa: tách phần hiển thị thông điệp ra riêng với Validator, do đó ta có thể thay cách hiển thị trực tiếp khi đang chạy chương trình.

1. Decorator

Đoạn code mẫu:



Cách sử dụng:

Ý nghĩa: có thể sử dụng nhiều validator cho một kiểu dữ liệu.

1. **Danh sách các tính năng**

|  |  |
| --- | --- |
| Tên tính năng | Mức độ hoàn thành |
| Cơ chế thông báo khi data không valid   * Tổng hợp các thông báo * Có các cách khác nhau thể hiện thông báo trên giao diện | **50%** |
| Thao tác thiết lập valid bằng code   * Tự kiểm tra với code thêm vào | **100%** |
| Thao tác thiết lập valid tự động thông qua khai báo ràng buộc dữ liệu | **0%** |
| Có thể kết hợp các validation với nhau cho cùng kiểu dữ liệu | **100%** |
| Hỗ trợ kết hợp regular expression | **100%** |
| Cho phép tạo custom validation | **100%** |