Template method

Nơi sử dụng

Template method trong thư mục ui/templatePattern

strategyPattern trong thư mục ui/strategyPattern

FormHome: đổi màn hình(đổi chiến lược) khi chạy và template method làm lại hoạt động (các child là các control)

Lý do sử dụng

Template Pattern giúp ta tạo nên một bộ khung của một chức năng nào đó và có thể cho lớp con định nghĩa lại một số hành vi trong bộ khung. Mô hình này áp dụng khi thay đổi các UserControl, nó sẽ thực hiện các hành vi trong một bộ khung đã được định nghĩa trước đó và điều đó tránh việc lặp đi lặp lại code ở nhiều nơi, đồng thời đối với việc các lớp con có hành vi khác nhau, nó giúp định nghĩa lại những hành vi riêng biệt đó, tránh ảnh hưởng đến các hành vi có cấu trúc giống nhau giữa lớp con, cũng như việc khởi tạo riêng cho các phương thức gây nên tình trạng dư thừa không đáng có. Về lâu về dài, điều này giúp hạn chế việc thay đổi code ở nhiều nơi mà chỉ cần chỉnh sửa ở một nơi nhất định.

Factory Pattern

Nơi sử dụng

Folder frontController (FormFactory)

Để tạo ra các màn hình để chuyển

Lý do sử dụng

Factory Pattern là mô hình giúp ta tạo ra các đối tượng theo yêu cầu, được áp dụng để tạo ra các đối tượng Form khác nhau phụ thuộc vào Form muốn khởi tạo, đồng thời mẫu thiết kế cũng giúp ta có một cấu trúc code dễ nhìn, dễ quản lý hơn rất nhiều khi muốn tạo ra một đối tượng.

Singleton Pattern:

Nơi sử dụng

Folder helper

@DatabaseMySql

Folder Dao

FrontController

CenterMeidatorImpl

Lý do sử dụng

Singleton Pattern là một trong những mô hình được sử dụng phổ biến nhất, nó giúp cho việc tạo đối tượng chỉ diễn ra đúng một lần từ khi khởi tạo ứng dụng. Mô hình này giúp giữ các instance của những lớp chỉ cần khởi tạo một lần như: lớp Dao, Helper,… điều đó giúp cho việc gọi đến những lớp này dễ dàng và quan trọng hơn là dữ liệu trong những lớp đó sẽ không bị thay đổi khi gọi lại, làm cho quá trình xây dựng ứng dụng được liền mạch và xuyên suốt.

AbstractFactory:

Nơi sử dụng

Folder helper

Folder Dao

@DatabaseFactory, @DatabaseMySql

Lý do sử dụng

Abstract Factory Pattern là mẫu thiết kế giúp quản lý việc trả về các đối tượng theo yêu cầu. Do đó mẫu thiết kế này tạo ra một nơi thực hiện một việc duy nhất là quản lý, tạo và trả về các đối tượng kết nối database, command, param,… Hiện tại ứng dụng chỉ kết nối với một loại cơ sở dữ liệu chính là SQL nhưng sau này nếu kết nối với một loại cơ sở dữ liệu khác thì mô hình vẫn hỗ trợ để có thể gọi các method trả về đối tượng trước đó đã sử dụng bằng cách thêm một lớp con thừa kế lớp cơ sở dữ liệu của lớp cha mà đã định nghĩa các hàm khởi tạo và điều chỉnh các đối tượng được tạo ra tùy thuộc vào cơ sở dữ liệu đó mà không ảnh hưởng đến lớp cơ sở dữ liệu cũ, cũng như chương trình đã được thực hiện trước đó. Mặt khác, ta cũng giảm thiểu đi việc sử dụng if else quá nhiều để tạo các đối tượng ở cơ sở dữ liệu mà ta muốn.

Command Pattern

Nơi sử dụng

Folder ui/Command

QuanLyKhachHangVaSuKienControl, QuanLyNhanVienControl, QuanLyNhapSanPhamControl, QuanLySanPhamControl

Lý do sử dụng

Command Pattern là mẫu thiết kế giúp đóng gói các thông tin cần thiết vào 1 đối tượng để thực hiện một hành động hay sự kiện nào đó. Nên chúng em áp dụng mẫu thiết kế này vào ứng dụng của chúng em để có thể thực hiện các hoạt động ẩn hoặc hiện các nút thêm, xóa, sửa, refresh, bảng dữ liệu ở mỗi màn hình quản lý. Điều đó giúp em có thể sau này muốn thay đổi code thực hiện các hoạt động ẩn hiện các nút cũng không ảnh hưởng đến lớp mà sử dụng nó, và đồng thời vì có nhiều màn hình sử dụng chức năng này nên khi muốn chĩnh sửa cũng không cần phải chĩnh sửa nhiều nơi.

Mediator Pattern

Nơi sử dụng

Thư mục MediatorControlScreen

Thư mục ui/control

Lý do sử dụng

Mediator Pattern là mẫu thiết kế giúp các class tương tác, giao tiếp với nhau. Chúng em áp dụng pattern này vào ứng dụng để giúp cho việc khi có một màn hình nào thay đổi data thì có thể nói cho những màn hình khác rằng data đã được thay đổi và khi bấm sang các màn hình được thông báo thì có thể biết là có nên load lại data không và việc này cũng giúp chúng em tránh đi việc lúc nào cũng load lại data khi chuyển giữa các màn hình.

Memento Pattern

Nơi sử dụng

BanHangControl

MementoBanHang

Lý do sử dụng

Memento Pattern là pattern giúp lưu lại trạng thái của một đối tượng để sau này có thể khôi phục lại trạng thái của đối tượng đó. Nên chúng em áp dụng pattern này vào ứng dụng để có thể lưu lại trạng thái giỏ hàng sau mỗi hoạt động để có thể thực hiện chức năng hoàn tác giỏ hàng trên ứng dụng và pattern này giúp em đơn giản code hơn trong việc sao lưu trạng thái của một đối tượng trong ứng dụng.