Question:

1. Mỗi pattern chỉ vẽ 1 sơ đồ lớp ở một chỗ sài tượng trưng thôi hay chỗ nào sài cũng phải vẽ.

Mô hình 3 lớp

Lý do sài

Mô hình ba lớp giúp ta tách các phần code về giao diện, xử lý, truy vấn ra riêng nên chúng em áp dụng mô hình trong ứng dụng để giúp người code dễ quản lý code, đọc code, khi có lỗi xảy ra thì có thể khoanh vùng lỗi nằm ở đâu, dễ bảo trì và khi ta muốn thay đổi giao diện hoặc một câu lệnh truy vấn nào đó cũng không ảnh hưỡng đến phần còn lại.

Front Controller Pattern

Nơi sài

Điều phối chuyển giữa các màn hình (Form)

Folder frontController (Dispatacher, FrontController)

Lý do sài

Front Controller Pattern là mô hình giúp ta xử lý các tiền xử lý và hậu xử lý trước khi điều hướng đến một nơi nào đó nên chúng em áp dụng mô hình vào ứng dụng để nó là một nơi tập trung để quản lý các điều hướng đến các Form trong ứng dụng, đồng thời giúp chúng em có thể tự điều chỉnh code, thêm code trước và sau khi điều hướng đến Form khác như là gọi hủy Form trước, mở Form sau hoặc sau này ta muốn thêm một hành động nào đó trước khi chuyển Form thì ta chỉ cần chỉnh sữa một nơi là được không cần phải chỉnh sữa ở nhiều nơi.

Template method, strategy pattern

Nơi sài

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 sài

Strategy Pattern giúp ta thay đổi dạng thuật toán hoặc một hành vi nào đó trong quá trình chạy ứng dụng nên chúng em áp dụng mô hình vào ứng dụng của chúng em để có thể thay đổi các UserControl trong quá trình chạy ứng dụng và mẫu thiết kế trên cũng tạo ra một nơi tập hợp các UserControl để thực hiện một hành vi có ý nghĩa chung nhưng xử lý khác nhau trên mỗi UserControl, nên sau này em có chỉnh sữa gì trên UserControl nào thì cũng không ảnh hưởng đến những UserControl khác và khi muốn chĩnh sửa đến việc thay đổi UserControl thì chỉ cần thay đổi ở một nơi không cần thay đổi ở nhiều nơi khác.

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 nên chúng em áp dụng mô hình vào ứng dụng của chúng em để có thể khi thay đổi các UserControl thì nó sẽ thực hiện các hành vi trong một bộ khung đã được định nghĩa trước đó và điều đó tránh đi việc lặp đi lặp lại code ở nhiều nơi, đồng thời nếu có hành vi nào trong bộ khung mà các UserControl khác nhau thì ta vẫn có thể điều chỉnh lại ở lớp UserControl đó, đồng thời nếu sau này ta muốn thay đổi bộ khung đó thì ta cũng sẽ chỉ cần chĩnh sửa ở một nơi mà không phải đi kiếm những chỗ sài mà chĩnh sửa.

Factory Pattern

Nơi sài

Folder frontController (FormFactory)

Để tạo ra các màn hình để chuyển

Lý do sài

Factory Pattern là mô hình giúp ta tạo ra các đối tượng theo yêu cầu nên chúng em áp dụng mô hình này vào ứng dụng để có thể tạo ra cái đối tượng Form khác nhau phụ thuộc vào tên Form mà muốn khởi tạo, đồng thời mẫu thiết sẽ 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 sài

Folder helper

@DatabaseMySql

Folder Dao

FrontController

Lý do sài

Singleton Pattern là một trong những mô hình được sử dụng phổ biến nhất, nó giúp cho việc tạo đối tượng diễn ra đúng một lần và lần sau không cần khởi tạo nữa nên chúng em áp dụng mô hình này vào ứng dụng để có thể giúp giữ các instance của những lớp chỉ cần khởi tạo một lần như những lớp Dao, Helper,… điều đó giúp cho việc gọi đến những lớp đó dễ dàng và điều quan trọng hơn là dữ liệu trong những lớp đó sẽ không bị thay đổi khi gọi lại.

AbstractFactory:

Nơi sài

Folder helper

Folder Dao

@DatabaseFactory, @DatabaseMySql

Lý do sài

Abstract Factory Pattern là mẫu thiết kế giúp quản lý việc trả về các đối tượng theo yêu cầu nên chúng em áp dụng mẫu thiết kế này vào ứng dụng để có thể tạo ra một nơi thực hiện một việc duy nhất là quản lý và tạo, trả về các đối tượng kết nối database, command, param,… Hiện tại ứng dụng của chúng em chỉ kết nối với một loại cơ sở dữ liệu chính là SQL nhưng nếu sau này chúng em có tạo một loại cơ sở dữ liệu khác để kết nối thì chúng em vẫn 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 mà sẽ tạo ra tùy thuộc vào cơ sở dữ liệu đó mà không ảnh hưởng đến lớp cơ sở dữ liệu cũ và chương trình đã được thực hiện trước đó. Và cũng từ đó ta cũng giảm đi việc if else quá nhiều để tạo các đối tượng ở cơ sở dữ liệu mà ta muốn.