TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**PHẠM ĐỨC MINH HIẾU – 52100796**

**PHAN HOÀNG PHÚ – 52100086**

**LÂM TRƯỜNG PHÚ – 52100570**

**XÂY DỰNG ỨNG DỤNG QUẢN LÝ**

**TRUNG TÂM TIN HỌC**

**BÁO CÁO CUỐI KỲ**

**MẪU THIẾT KẾ**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**PHẠM ĐỨC MINH HIẾU – 52100796**

**PHAN HOÀNG PHÚ – 52100086**

**LÂM TRƯỜNG PHÚ – 52100570**

**XÂY DỰNG ỨNG DỤNG QUẢN LÝ**

**TRUNG TÂM TIN HỌC**

**BÁO CÁO CUỐI KỲ**

**MẪU THIẾT KẾ**

Người hướng dẫn

**ThS. Vũ Đình Hồng**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024**

**LỜI CẢM ƠN**

Lời đầu tiên chúng em xin được phép cảm ơn quý thầy cô khoa Công nghệ Thông Tin trường Đại học Tôn Đức Thắng đã tạo mọi điều kiện cho chúng em tạo mọi điều kiện tốt nhất để được học tập môn Mẫu thiết kế.

Chúng em cũng xin được gửi lời cảm ơn đến ThS Vũ Đình Hồng, thầy đã nhiệt tình giảng dạy, trang bị đầy đủ kiến thức để em có thể hoàn thành dự án cuối kỳ này.

Cuối cùng, do hạn chế về mặt kiến thức, kính mong thầy cô có thể bỏ qua những sai sót nhỏ và chỉ ra được những lỗi sai của chúng em trong bài báo cáo này để những bài báo cáo sau của chúng em được hoàn thiện hơn.

Một lần nữa chúng em xin chân thành cảm ơn thầy và toàn thể quý thầy cô khoa Công Nghệ Thông Tin trường Đại học Tôn Đức Thắng.

*TP. Hồ Chí Minh, ngày 29 tháng 04 năm 2024*

*Tác giả*

*Phạm Đức Minh Hiếu*

*Phan Hoàng Phú*

*Lâm Trường Phú*

**CÔNG TRÌNH ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Chúng em xin cam đoan đây là công trình nghiên cứu của riêng chúng em và được sự hướng dẫn khoa học của ThS.Vũ Đình Hồng. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong Dự án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào chúng em xin hoàn toàn chịu trách nhiệm về nội dung Dự án của mình**. Trường Đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do chúng em gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày 29 tháng 04 năm 2024*

*Tác giả*

*Phạm Đức Minh Hiếu*

*Phan Hoàng Phú*

*Lâm Trường Phú*

**TÓM TẮT**

Ứng dụng này tập trung vào việc phát triển nghiệp vụ Quản lý trung tâm tin học. Sử dụng Winform kết hợp với .NET Framework phiên bản 4.7.2 và cơ sở dữ liệu SQL Server để lưu trữ. Ứng dụng cho phép quản lý Học Viên, Giảng Viên, Khóa học, Lớp học, Phòng học, xem được thống kê về doanh thu và tạo hóa đơn khi mua khóa học. Đặc biệt, dự án này còn áp dụng các mẫu thiết kế giúp cho việc tối ưu hóa cấu trúc mã nguồn để dễ dàng bảo trì và quản lý. Việc áp dụng các mẫu thiết kế cách hiệu quả sẽ giúp giảm sự phức tạp của hệ thống. Dễ dàng mở rộng chương trình và bảo trì ứng dụng trong tương lai.

**MỤC LỤC**

[DANH MỤC HÌNH VẼ vii](#_Toc165387808)

[DANH MỤC BẢNG BIỂU ix](#_Toc165387809)

[DANH MỤC CÁC CHỮ VIẾT TẮT x](#_Toc165387810)

[CHƯƠNG 1. GIỚI THIỆU 1](#_Toc165387811)

[1.1 Tổng quan về ứng dụng 1](#_Toc165387812)

[1.2 Các chức năng chính 1](#_Toc165387813)

[CHƯƠNG 2. ÁP DỤNG CÁC MẪU THIẾT KẾ 4](#_Toc165387814)

[2.1 Singleton Pattern 4](#_Toc165387815)

[2.1.1 Vấn đề 4](#_Toc165387816)

[2.1.2 Lí do áp dụng 4](#_Toc165387817)

[2.1.3 Sơ đồ lớp 4](#_Toc165387818)

[2.1.4 Code áp dụng 5](#_Toc165387819)

[2.2 Strategy Pattern 6](#_Toc165387820)

[2.2.1 Vấn đề 6](#_Toc165387821)

[2.2.2 Lí do áp dụng 6](#_Toc165387822)

[2.2.3 Sơ đồ lớp 7](#_Toc165387823)

[2.2.4 Code áp dụng 7](#_Toc165387824)

[2.3 Template Method Pattern 10](#_Toc165387825)

[2.3.1 Vấn đề 10](#_Toc165387826)

[2.3.2 Lí do áp dụng 11](#_Toc165387827)

[2.3.3 Sơ đồ lớp 11](#_Toc165387828)

[2.3.4 Code áp dụng 12](#_Toc165387829)

[2.4 Simple Factory Pattern 14](#_Toc165387830)

[2.4.1 Vấn đề 14](#_Toc165387831)

[2.4.2 Lí do áp dụng 15](#_Toc165387832)

[2.4.3 Sơ đồ lớp 15](#_Toc165387833)

[2.4.4 Code áp dụng 15](#_Toc165387834)

[2.5 Command Pattern 16](#_Toc165387835)

[2.5.1 Vấn đề 16](#_Toc165387836)

[2.5.2 Lí do áp dụng 16](#_Toc165387837)

[2.5.3 Sơ đồ lớp 17](#_Toc165387838)

[2.5.4 Code áp dụng 17](#_Toc165387839)

[2.6 State Pattern 19](#_Toc165387840)

[2.6.1 Vấn đề 19](#_Toc165387841)

[2.6.2 Lí do áp dụng 20](#_Toc165387842)

[2.6.3 Sơ đồ lớp 20](#_Toc165387843)

[2.6.4 Code áp dụng 20](#_Toc165387844)

[2.7 Façade Pattern 22](#_Toc165387845)

[2.7.1 Vấn đề 22](#_Toc165387846)

[2.7.2 Lí do áp dụng 22](#_Toc165387847)

[2.7.3 Sơ đồ lớp 22](#_Toc165387848)

[2.7.4 Code áp dụng 23](#_Toc165387849)

[2.8 Adapter Pattern 24](#_Toc165387850)

[2.8.1 Vấn đề 24](#_Toc165387851)

[2.8.2 Lí do áp dụng 25](#_Toc165387852)

[2.8.3 Sơ đồ lớp 25](#_Toc165387853)

[2.8.4 Code áp dụng 25](#_Toc165387854)

[2.9 Decorator Pattern 28](#_Toc165387855)

[2.9.1 Vấn đề 28](#_Toc165387856)

[2.9.2 Lí do áp dụng 28](#_Toc165387857)

[2.9.3 Sơ đồ lớp 28](#_Toc165387858)

[2.9.4 Code áp dụng 29](#_Toc165387859)

[TÀI LIỆU THAM KHẢO 33](#_Toc165387860)

# DANH MỤC HÌNH VẼ

[Hình 2.1 Sơ đồ lớp của Singleton Pattern 5](#_Toc165387770)

[Hình 2.2 Lớp HocVienFacade 5](#_Toc165387771)

[Hình 2.3 Lớp fThemHocVien 6](#_Toc165387772)

[Hình 2.4 Sơ đồ lớp của Strategy Pattern 7](#_Toc165387773)

[Hình 2.5 Interface IValidateStrategy 7](#_Toc165387774)

[Hình 2.6 Lớp ValidatedInput 8](#_Toc165387775)

[Hình 2.7 Lớp BirthdayValidate 8](#_Toc165387776)

[Hình 2.8 Lớp CourseFeeValidate 9](#_Toc165387777)

[Hình 2.9 Lớp PhoneNumberValidate 9](#_Toc165387778)

[Hình 2.10 Lớp PasswordValidate 10](#_Toc165387779)

[Hình 2.11 Lớp EmailValidate 10](#_Toc165387780)

[Hình 2.12 Sơ đồ lớp Template Method Pattern 12](#_Toc165387781)

[Hình 2.13 Lớp CRUD\_Template 13](#_Toc165387782)

[Hình 2.14 Lớp DAL\_HocVien 14](#_Toc165387783)

[Hình 2.15 Sơ đồ lớp của Simple Factory Pattern 15](#_Toc165387784)

[Hình 2.16 Lớp FileTypeImportFactory 16](#_Toc165387785)

[Hình 2.17 Sơ đồ lớp của Command Pattern 17](#_Toc165387786)

[Hình 2.18 Interface InavigationCommand và lớp NavigateToHocVienCommand 18](#_Toc165387787)

[Hình 2.19 Lớp CommandHistory 19](#_Toc165387788)

[Hình 2.20 Sơ đồ lớp của State Pattern 20](#_Toc165387789)

[Hình 2.21 Interface LopHocState và lớp NotStarted 21](#_Toc165387790)

[Hình 2.22 Lớp LopHocContext 21](#_Toc165387791)

[Hình 2.23 Sơ đồ lớp của Façade Pattern 23](#_Toc165387792)

[Hình 2.24 Lớp HocVienFacade 24](#_Toc165387793)

[Hình 2.25 Sơ đồ lớp Adapter Pattern 25](#_Toc165387794)

[Hình 2.26 Interface ImportExcel và lớp ImportExcelImpl 26](#_Toc165387795)

[Hình 2.27 Lớp CSVAdapter 27](#_Toc165387796)

[Hình 2.28 Giao diện ImportCSV và lớp ImportCSVImpl 27](#_Toc165387797)

[Hình 2.29 Sơ đồ lớp của Decorator Pattern 29](#_Toc165387798)

[Hình 2.30 Lớp DefaultSorting, NameHVSorting và NameGVSorting 30](#_Toc165387799)

[Hình 2.31 Lớp DecoratorSorting, EmailHVSorting và DOBHVSorting 31](#_Toc165387800)

[Hình 2.32 Lớp EmailGVSorting, DOBGVSorting 32](#_Toc165387801)

# DANH MỤC BẢNG BIỂU

# DANH MỤC CÁC CHỮ VIẾT TẮT

|  |  |
| --- | --- |
| BUS | Business Layer |
|  |  |

# GIỚI THIỆU

## Tổng quan về ứng dụng

Trong thời đại số hóa và phát triển công nghệ 4.0, việc sử dụng máy tính và kiến thức về công nghệ thông tin trở thành một phần quan trọng không thể thiếu trong cuộc sống hàng ngày và trong môi trường công việc. Đáp ứng nhu cầu này, nhiều trung tâm tin học đã ra đời, cung cấp các khóa học để cung cấp kiến thức và kỹ năng cần thiết cho mọi người.

Nhận thấy tầm quan trọng của việc áp dụng công nghệ vào quản lý, các trung tâm tin học ngày nay đã sử dụng các phương tiện kỹ thuật số và công nghệ thông tin để quản lý hoạt động của mình một cách hiệu quả hơn. Điều này giúp tăng cường hiệu suất làm việc, cải thiện quá trình quản lý và thúc đẩy sự phát triển của trung tâm.

Hệ thống quản lý trung tâm tin học được phát triển nhằm mục đích tự động hóa và tối ưu hóa các quy trình quản lý. Bằng cách số hóa dữ liệu và xử lý thông tin, hệ thống này giúp quản lý các hoạt động hàng ngày của trung tâm một cách nhanh chóng và hiệu quả hơn. Đồng thời, nó cung cấp các công cụ để giúp quản lý và vận hành trung tâm một cách linh hoạt và hiệu quả.

## Các chức năng chính

* Đăng nhập và quản lý tài khoản:
* Admin có thể đăng nhập vào hệ thống với tài khoản và mật khẩu duy nhất.
* Admin có thể quản lý tài khoản của học viên, giảng viên của trung tâm.
* Quản lý thông tin học viên:
* Admin có thể thêm mới, xóa, sửa đổi thông tin học viên.
* Admin có thể tìm kiếm thông tin về học viên.
* Admin cần đảm bảo tính chính xác và bảo mật của thông tin học viên.
* Admin có thể sắp xếp danh sách học viên theo nhiều tiêu chí (tính năng mới).
* Quản lý thông tin khóa học:
* Admin có thể thêm mới, xóa, sửa đổi thông tin về khóa học.
* Admin có thể tìm kiếm thông tin về các khóa học đang cung cấp.
* Admin cần đảm bảo rằng các thông tin về khóa học được cập nhật thường xuyên để phù hợp với nhu cầu của học viên.
* Quản lý thông tin giảng viên:
* Admin có thể thêm mới, xóa, sửa đổi thông tin giảng viên.
* Admin có thể tìm kiếm thông tin về giảng viên.
* Admin cần đảm bảo rằng thông tin giảng viên được cập nhật thường xuyên.
* Admin có thể thêm một danh sách giảng viên thông qua file excel hoặc csv (tính năng mới).
* Admin có thể sắp xếp danh sách giảng viên theo nhiều tiêu chí (tính năng mới).
* Quản lý lớp học:
* Admin có thể tạo mới, xóa, sửa đổi thông tin về lớp học.
* Admin có thể tìm kiếm thông tin về các lớp học.
* Admin có thể phân công giảng viên cho các lớp học.
* Quản lý hóa đơn:
* Admin có thể quản lý thông tin về học phí và các khoản thu khác.
* Admin có thể tạo hóa đơn cho học viên.
* Admin có thể tìm kiếm thông tin về các hóa đơn.
* Thống kê:
* Admin có thể xem thông tin thống kê về doanh thu trong một khoảng thời gian xác định.
* Giảng viên:
* Giảng viên có thể chỉnh sửa thông tin cá nhân.
* Giảng viên có thể thay đổi mật khẩu.
* Giảng viên có thể xem danh sách lớp học và in danh sách.
* Giảng viên có thể xem lịch dạy.
* Học viên:
* Học viên có thể chỉnh sửa thông tin cá nhân.
* Học viên có thể thay đổi mật khẩu.
* Học viên có thể xem lịch học.
* Học viên có thể xem khóa học.

# ÁP DỤNG CÁC MẪU THIẾT KẾ

## Singleton Pattern

### Vấn đề

Trong ứng dụng Quản lý Trung tâm tin học, lớp Façade được tạo ra làm cầu nối giữa giao diện Form và hệ thống BUS. Vì vậy mỗi lần gọi một chức năng để thực hiện một yêu cầu cụ thể thì một đối tượng mới của Façade sẽ được tạo ra, điều này gây tốn bộ nhớ và giảm hiệu suất của ứng dụng. Để giải quyết vấn đề này, chúng ta sẽ sử dụng Singleton Pattern để xác định rằng tại một thời điểm sẽ chỉ có một đối tượng Façade duy nhất tồn tại từ đó tăng hiệu suất và giảm việc sử dụng bộ nhớ của hệ thống gây tốn kém tài nguyên.

### Lí do áp dụng

**Đảm bảo tồn tại duy nhất:** Singleton đảm bảo rằng chỉ có một đối tượng Façade duy nhất được tạo ra trong suốt quá trình chạy của ứng dụng. Điều này giúp đảm bảo rằng mọi yêu cầu thực hiện chức năng từ giao diện Form đều sử dụng cùng một đối tượng Façade. Bằng cách này, việc tạo ra nhiều đối tượng Façade không cần thiết và gây tốn tài nguyên bộ nhớ sẽ được tránh.

**Tăng hiệu suất:** Bằng cách sử dụng Singleton, ta tránh việc tạo ra đối tượng Façade mới mỗi khi thực hiện một yêu cầu cụ thể. Thay vào đó, ta sử dụng đối tượng Façade hiện có để xử lý yêu cầu. Điều này giúp tiết kiệm thời gian và tăng hiệu suất của ứng dụng, vì không cần phải tạo ra và hủy bỏ đối tượng Façade trong quá trình thực hiện các yêu cầu.

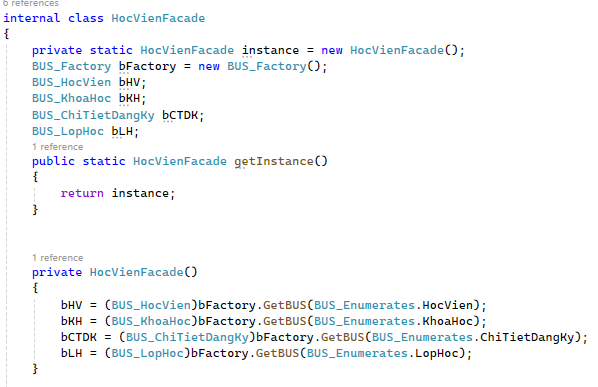
### Sơ đồ lớp

A diagram of a computer

Description automatically generated

Hình 2.1 Sơ đồ lớp của Singleton Pattern

### Code áp dụng



Hình 2.2 Lớp HocVienFacade

A screenshot of a computer code

Description automatically generated

Hình 2.3 Lớp fThemHocVien

## Strategy Pattern

### Vấn đề

Trong ứng dụng Quản lý Trung tâm tin học, việc kiểm tra tính hợp lệ của dữ liệu được nhập từ các TextBox và DataGridView cho việc thêm học viên, giảng viên là một vấn đề cần thiết để tránh việc người dùng nhập vào những thông tin không hợp lệ nhưng vẫn thêm thông tin này vào cơ sở dữ liệu dẫn đến việc quản lý dữ liệu không tốt. Hơn nữa, chúng ta cũng cần linh hoạt để thay đổi các tiêu chí kiểm tra tùy thuộc vào dữ liệu nhập của người dùng đang là gì. Ví dụ như dữ liệu nhập vào có phải email hợp lệ hay không, số điện thoại nhập vào có đúng định dạng hay không. Lúc này, giải pháp thực hiện là chúng ta sẽ đóng gói các quy tắc kiểm tra vào lớp riêng biệt và linh hoạt lựa chọn chúng khi sử dụng ứng dụng. Như vậy chúng ta cần sử dụng Strategy Pattern.

### Lí do áp dụng

**Tính linh hoạt:** Mẫu thiết kế Strategy cho phép ta đóng gói các quy tắc kiểm tra vào các lớp riêng biệt, gọi là các strategies. Mỗi strategy sẽ thực hiện một quy tắc kiểm tra cụ thể, chẳng hạn kiểm tra tính hợp lệ của email, số điện thoại, v.v. Điều này cho phép ta linh hoạt lựa chọn và thay đổi các quy tắc kiểm tra tùy thuộc vào dữ liệu nhập của người dùng.

**Tái sử dụng và duy trì:** Mẫu Strategy cho phép ta tái sử dụng các quy tắc kiểm tra và duy trì chúng trong các lớp riêng biệt. Điều này giúp tránh việc lặp lại mã kiểm tra trong nhiều nơi khác nhau trong ứng dụng.

**Mở rộng và bảo trì:** Mẫu Strategy cho phép ta mở rộng và bảo trì các quy tắc kiểm tra một cách dễ dàng. Nếu ta muốn thêm một quy tắc kiểm tra mới, ta chỉ cần tạo một chiến lược mới và triển khai quy tắc kiểm tra đó.

### Sơ đồ lớp

A diagram of a diagram

Description automatically generated

Hình 2.4 Sơ đồ lớp của Strategy Pattern

### Code áp dụng

A close-up of a computer code

Description automatically generated

Hình 2.5 Interface IValidateStrategy

A screen shot of a computer code

Description automatically generated

Hình 2.6 Lớp ValidatedInput

A computer screen shot of a code

Description automatically generated

Hình 2.7 Lớp BirthdayValidate

A screenshot of a computer program

Description automatically generated

Hình 2.8 Lớp CourseFeeValidate

A screenshot of a computer code

Description automatically generated

Hình 2.9 Lớp PhoneNumberValidate

A computer screen shot of a code

Description automatically generated

Hình 2.10 Lớp PasswordValidate

A computer screen shot of a program code

Description automatically generated

Hình 2.11 Lớp EmailValidate

## Template Method Pattern

### Vấn đề

Trong ứng dụng Quản lý Trung tâm tin học, các quy trình cho việc thêm, xóa, sửa dữ liệu khá phức tạp và cấu trúc của chúng khá giống nhau cho các bảng, quy trình thực hiện bao gồm: mở kết nối cơ sở dữ liệu, tạo và cấu hình câu lệnh truy vấn, thực thi câu lệnh truy vấn dựa vào loại thao tác yêu cầu, đóng kết nối và trả về kết quả. Như vậy chúng ta thấy rằng quy trình thực hiện của các lớp là khá giống nhau. Và nếu chúng ta không có lớp cha đại diện mà chỉ sao chép và dán code thì sẽ gây khó khăn cho việc bảo trì hệ thống và làm cho mã nguồn thêm dài dòng. Lúc này, chúng ta cần xây dựng một lớp đại diện chứa phương thức template cho quy trình thực hiện của việc thêm, xóa và sửa dữ liệu của các bảng. Các lớp con sẽ mở rộng lớp này để triển khai các phương thức cụ thể cho từng bảng của cơ sở dữ liệu.

### Lí do áp dụng

**Tái sử dụng mã nguồn:** Mẫu thiết kế Template Method cho phép tái sử dụng mã nguồn chung cho các quy trình thêm, xóa và sửa dữ liệu giống nhau. Bằng cách xây dựng một lớp cha đại diện (base class) chứa phương thức template, ta có thể định nghĩa các bước chung trong quy trình thực hiện và để các lớp con triển khai các bước cụ thể cho từng bảng dữ liệu

**Tách biệt trách nhiệm:** Mẫu Template Method giúp tách biệt trách nhiệm giữa các bước chung và bước cụ thể trong quy trình thực hiện. Các bước chung được định nghĩa trong lớp cha đại diện, trong khi các bước cụ thể được triển khai trong các lớp con.

**Bảo trì và mở rộng:** Mẫu Template Method giúp dễ dàng bảo trì và mở rộng quy trình thực hiện. Nếu có thay đổi trong quy trình chung, ta chỉ cần sửa đổi trong lớp cha đại diện mà không cần can thiệp vào các lớp con

### Sơ đồ lớp

A diagram of a computer

Description automatically generated

Hình 2.12 Sơ đồ lớp Template Method Pattern

### Code áp dụng

A screenshot of a computer program

Description automatically generated

Hình 2.13 Lớp CRUD\_Template

A screenshot of a computer program

Description automatically generated

Hình 2.14 Lớp DAL\_HocVien

Tương tự cho các lớp DAL\_GiangVien, DAL\_HoaDon, DAL\_ChiTietHoaDon, DAL\_ChiTietDangKy, DAL\_KhoaHoc.

## Simple Factory Pattern

### Vấn đề

Trong ứng dụng Quản lý Trung tâm tin học, khi thực hiện việc thêm danh sách giảng viên bằng tệp tin thì ứng dụng sẽ xử lý điều kiện xem tệp tin được thêm vào có đuôi là loại gì (Excel, CSV). Lúc này ứng dụng cần phải có khả năng tạo và biết được loại tệp tin chứa danh sách giảng viên mà người dùng thêm vào là gì để chọn lựa được phương thức phù hợp, giúp dễ dàng mở rộng và tăng tính linh hoạt. Lúc này chúng ta sẽ tách các đối tượng Import loại tệp tin riêng vào một factory method nằm trong lớp Factory để dựa vào Factory này tạo ra đối tượng Import danh sách giảng viên theo loại tệp tin theo lựa chọn của người dùng.

### Lí do áp dụng

**Tính linh hoạt và mở rộng:** Mẫu thiết kế Factory Pattern cho phép dễ dàng mở rộng và thêm các loại đối tượng Import danh sách giảng viên mới trong tương lai (vd: JSON, XML) mà không cần sửa đổi các phần khác trong ứng dụng.

**Giảm sự phụ thuộc:** Factory Pattern giúp giảm sự phụ thuộc giữa các lớp trong ứng dụng. Thay vì các lớp khác nhau phải biết và tạo trực tiếp các đối tượng Import danh sách giảng viên, chúng chỉ cần giao tiếp với Factory để tạo đối tượng tương ứng.

### Sơ đồ lớp

A diagram of a data flow

Description automatically generated

Hình 2.15 Sơ đồ lớp của Simple Factory Pattern

### Code áp dụng

A screenshot of a computer program

Description automatically generated

Hình 2.16 Lớp FileTypeImportFactory

## Command Pattern

### Vấn đề

Trong ứng dụng Quản lý trung tâm tin học, có những lúc người dùng làm việc trên các Tab mà đôi khi lại bấm nhầm vào Tab khác, thi thoảng xảy ra việc người dùng quên mất rằng trước đó mình đang làm việc trên Tab nào hoặc người dùng muốn điều hướng tới Tab mà vừa rời đi trước đó. Lúc này, chúng ta cần một bộ công cụ điều hướng để có thể dễ dàng di chuyển qua lại giữa các Tab mà không cần phải ghi nhớ những Tab mà mình đã truy cập trước đó. Việc sử dụng Command Pattern nhằm để tách biệt logic xử lý các hành động di chuyển Tab này thành các đối tượng Command riêng biệt. Điều này sẽ giúp tăng tính linh hoạt cho ứng dụng và dễ dàng bảo trì hệ thống.

### Lí do áp dụng

**Tách biệt logic:** Command Pattern giúp tách biệt logic xử lý các hành động di chuyển Tab và logic của các Tab khỏi nhau. Mỗi hành động di chuyển Tab sẽ được triển khai thành một đối tượng Command riêng biệt. Điều này giúp giữ cho mã nguồn dễ đọc, dễ bảo trì và tăng tính tái sử dụng.

**Điều hướng dễ dàng:** Command Pattern cho phép người dùng dễ dàng di chuyển qua lại giữa các Tab mà không cần ghi nhớ trạng thái trước đó của các Tab. Mỗi đối tượng Command sẽ chứa thông tin về Tab mà nó điều hướng đến hoặc rời đi.

**Mở rộng và bảo trì dễ dàng:** Command Pattern cho phép dễ dàng mở rộng và thêm các hành động di chuyển Tab mới hoặc các chức năng khác liên quan đến điều hướng và xử lý các nút trong giao diện người dùng.

### Sơ đồ lớp

A diagram of a command

Description automatically generated

Hình 2.17 Sơ đồ lớp của Command Pattern

### Code áp dụng

A screenshot of a computer program

Description automatically generated

Hình 2.18 Interface InavigationCommand và lớp NavigateToHocVienCommand

Thực hiện tương tự cho các lớp NavigateToHoaDonCommand, NavigateToGiangVienCommand, NavigateToBaoCaoCommand, NavigateToLopHocCommand, NavigateToKhoaHocCommand. Invoker của ứng dụng này là CommandHistory.

A screenshot of a computer program

Description automatically generated

Hình 2.19 Lớp CommandHistory

## State Pattern

### Vấn đề

Đối với đối tượng lớp học, ứng dụng Quản lý trung tâm tin học đã cung cấp cho đối tượng này nhiều trạng thái khác nhau (chưa bắt đầu, đang diễn ra, đã kết thúc, đã hủy), việc kiểm tra và xử lý các trạng thái khác nhau của Lớp học trong lúc run-time sẽ trở nên phức tạp hơn. Chúng ta sẽ phải sử dụng các câu lệnh điều kiện (if-else hoặc switch-case) để xác định trạng thái hiện tại và thực hiện cập nhật lại dữ liệu trong database. Điều này dẫn đến gây phức tạp mã nguồn, khó hiểu và khó bảo trì. Vì vậy, áp dụng State Pattern giúp tách biệt các trạng thái của đối tượng khỏi các hành vi của nó. Trong trường hợp này, các trạng thái của Lớp học được triển khai trong các lớp riêng biệt, và chúng có thể được thay đổi mà không ảnh hưởng đến cách thức thực hiện các hành vi liên quan. Đồng thời chúng ta có thể dễ dàng mở rộng hệ thống bằng cách thêm các trạng thái mới mà không ảnh hưởng đến các lớp hiện có.

### Lí do áp dụng

**Tách biệt trạng thái và hành vi:** State Pattern giúp tách biệt các trạng thái của đối tượng Lớp học khỏi các hành vi của nó. Thay vì sử dụng các câu lệnh điều kiện phức tạp để kiểm tra và xử lý các trạng thái khác nhau, chúng ta triển khai mỗi trạng thái thành một lớp riêng biệt. Mỗi lớp trạng thái chịu trách nhiệm xác định hành vi cụ thể là cập nhật lại mục trạng thái trong cơ sở dữ liệu lúc run-time khi đối tượng Lớp học ở trong trạng thái đó.

**Xử lý chuyển đổi trạng thái**: State Pattern cung cấp một cách tiện lợi để xử lý chuyển đổi trạng thái của đối tượng. Thay vì phải xử lý chuyển đổi trạng thái tại nơi sử dụng, chúng ta có thể xử lý chuyển đổi trạng thái trong lớp trạng thái tương ứng.

### Sơ đồ lớp

A diagram of a computer

Description automatically generated

Hình 2.20 Sơ đồ lớp của State Pattern

### Code áp dụng

A computer screen shot of a code

Description automatically generated

Hình 2.21 Interface LopHocState và lớp NotStarted

Thực hiện tương tự với các lớp OnGoing, Finished, Cancelled.

A screenshot of a computer code

Description automatically generated

Hình 2.22 Lớp LopHocContext

## Façade Pattern

### Vấn đề

Trong ứng dụng quản lý Trung tâm tin học, việc mà mã nguồn của giao diện Form thêm Học Viên chứa quá nhiều các BUS và điều này sẽ gây khó khăn trong việc quản lý mã nguồn, khiến cho giao diện thêm Học Viên chứa nhiều BUS gây rối mắt và sẽ khó quản lý nếu có thành phần BUS nào đó bảo trì và chỉnh sửa. Để giảm bớt sự phức tạp và cải thiện cấu trúc mã nguồn của Form thêm Học Viên trong ứng dụng quản lý Trung tâm tin học, chúng ta sẽ sử dụng mẫu thiết kế Façade Pattern. Mẫu thiết kế này cho phép chúng ta tạo ra một lớp trung gian (façade) để che giấu sự phức tạp của các BUS và công việc phức tạp khác và cung cấp một giao diện đơn giản để tương tác với giao diện người dùng.

### Lí do áp dụng

**Cung cấp giao diện đơn giản:** Façade Pattern cho phép tạo ra một giao diện đơn giản và dễ sử dụng để tương tác với giao diện người dùng. Thay vì trực tiếp gọi các BUS phức tạp từ giao diện Form, người dùng chỉ cần tương tác với lớp façade duy nhất.

**Giảm sự phức tạp của mã nguồn:** Giao diện Form thêm Học Viên có thể chứa nhiều BUS (Business Service) đảm nhiệm các nhiệm vụ phức tạp như xử lý dữ liệu, tương tác với database, kiểm tra điều kiện và thực hiện các logic phức tạp. Việc sử dụng Façade Pattern cho phép tạo ra một lớp trung gian (façade) nhằm che giấu sự phức tạp này.

**Tách biệt giao diện người dùng và logic xử lý:** Sử dụng Façade Pattern giúp tách biệt hoàn toàn giao diện người dùng và logic xử lý phức tạp bên trong.

### Sơ đồ lớp

A diagram of a computer program

Description automatically generated

Hình 2.23 Sơ đồ lớp của Façade Pattern

### Code áp dụng

A screenshot of a computer program

Description automatically generated

Hình 2.24 Lớp HocVienFacade

## Adapter Pattern

### Vấn đề

Đối với ứng dụng quản lý Trung tâm tin học, chúng ta có chức năng import danh sách Giảng Viên bằng tệp tin Excel. Tuy nhiên, nếu như tệp tin Excel có kích thước quá lớn thì sẽ không hiệu quả bằng việc import danh sách bằng tệp tin CSV. Vì vậy chúng ta bổ sung thêm chức năng import danh sách Giảng Viên bằng tệp tin CSV cho ứng dụng. Nhưng chúng ta vẫn muốn giữ lại chức năng import bằng tệp tin Excel vì nó thân thiện với người dùng. Lúc này chúng ta sẽ sử dụng Adapter Pattern, nó cho phép chúng ta tạo ra một lớp trung gian để chuyển đổi giao diện của một lớp thành một giao diện khác mà client mong muốn. Trong trường hợp này, chúng ta sẽ tạo ra một Adapter để chuyển đổi từ giao diện của tệp tin Excel sang giao diện của tệp tin CSV.

### Lí do áp dụng

**Giữ lại chức năng import dữ liệu từ tệp tin Excel:** Tệp tin Excel là một định dạng phổ biến và thân thiện với người dùng. Việc sử dụng Adapter Pattern cho phép giữ lại chức năng import dữ liệu từ tệp tin Excel mà không cần thay đổi các thành phần hiện tại của hệ thống. Adapter sẽ đóng vai trò là một lớp trung gian để chuyển đổi giao diện của tệp tin Excel thành giao diện mà hệ thống có thể xử lý.

### Sơ đồ lớp

A diagram of a data flow

Description automatically generated

Hình 2.25 Sơ đồ lớp Adapter Pattern

### Code áp dụng

A screenshot of a computer program

Description automatically generated

Hình 2.26 Interface ImportExcel và lớp ImportExcelImpl

A screenshot of a computer program

Description automatically generated

Hình 2.27 Lớp CSVAdapter

A screen shot of a computer code

Description automatically generated

Hình 2.28 Giao diện ImportCSV và lớp ImportCSVImpl

## Decorator Pattern

### Vấn đề

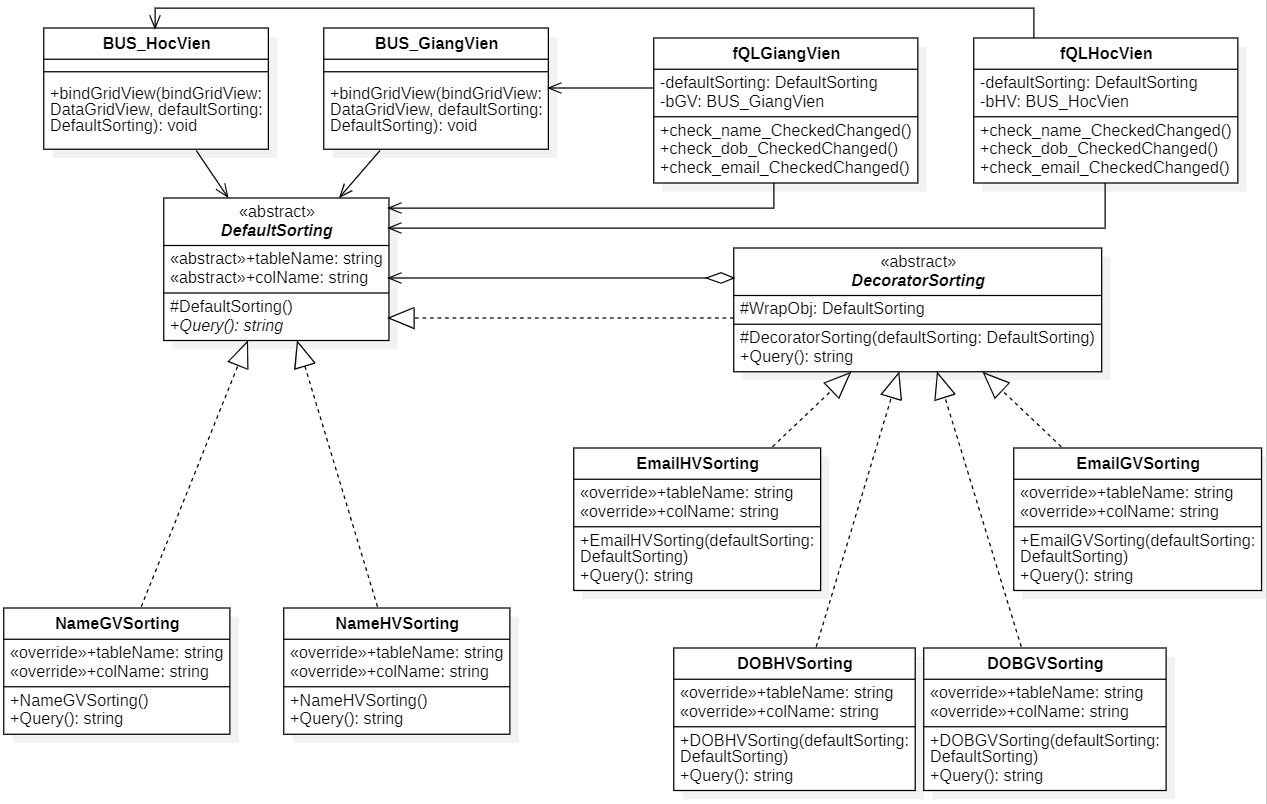
Đối với ứng dụng Quản lý trung tâm tin học, các danh sách Giảng Viên và Học Viên đã được DataGridView hỗ trợ cho việc sắp xếp theo các cột. Tuy nhiên, DataGridView chỉ hỗ trợ cho việc sắp xếp theo một cột mà thôi. Đối với người dùng, sẽ có người muốn sắp xếp danh sách theo nhiều tiêu chí. Ban đầu, chúng ta dự định tạo nhiều lớp và trả về câu truy vấn sắp xếp theo tiêu chí được đặt ra. Nhưng nếu như vậy thì nếu có quá nhiều tiêu chí sẽ tạo ra số lượng lớn tổ hợp sắp xếp và lúc này chúng ta sẽ phải tạo rất nhiều tổ hợp, điều này gây ra khó khăn trong việc bảo trì và quản lý ứng dụng.

Để giải quyết vấn đề này, chúng ta sẽ sử dụng Decorator Pattern. Trong trường hợp này, chúng ta sẽ tạo ra một lớp trừu tượng để đại diện cho các tiêu chí sắp xếp, sau đó tạo ra các lớp cụ thể để thực hiện sắp xếp theo từng tiêu chí cụ thể. Bằng cách này, chúng ta có thể kết hợp các tiêu chí sắp xếp khác nhau một cách linh hoạt mà không cần tạo ra một lớp mới cho mỗi tổ hợp sắp xếp.

### Lí do áp dụng

**Linh hoạt trong việc kết hợp và mở rộng tiêu chí sắp xếp:** Bằng cách sử dụng Decorator Pattern, chúng ta có thể tạo ra các lớp decorator cụ thể cho từng tiêu chí sắp xếp. Mỗi lớp decorator sẽ kết hợp một tiêu chí sắp xếp cụ thể vào danh sách hiện có.

### Sơ đồ lớp



Hình 2.29 Sơ đồ lớp của Decorator Pattern

### Code áp dụng

A screenshot of a computer program

Description automatically generated

Hình 2.30 Lớp DefaultSorting, NameHVSorting và NameGVSorting

A screenshot of a computer program

Description automatically generated

Hình 2.31 Lớp DecoratorSorting, EmailHVSorting và DOBHVSorting

A screenshot of a computer program

Description automatically generated

Hình 2.32 Lớp EmailGVSorting, DOBGVSorting

# TÀI LIỆU THAM KHẢO

Tiếng Việt

* 1. Hoàng Đinh, “Design Patterns là gì? Tại sao nó lại là trợ thủ đắc lực của Developers - Tổng hợp 23 mẫu Design Pattern”, Viblo, <https://viblo.asia/p/design-patterns-la-gi-tai-sao-no-lai-la-tro-thu-dac-luc-cua-developers-tong-hop-23-mau-design-pattern-GrLZDBQV5k0>.
  2. Giang Phan, “Tổng hợp các bài hướng dẫn về Design Pattern - 23 mẫu cơ bản của GoF”, Viblo, <https://viblo.asia/p/tong-hop-cac-bai-huong-dan-ve-design-pattern-23-mau-co-ban-cua-gof-3P0lPQPG5ox>.

Tiếng Anh

* 1. Refactoring Guru, “Design Pattern”, Refactoring Guru, <https://refactoring.guru/design-patterns/>.
  2. GeeksForGeeks, “Software Design Patterns Tutorial”, GeeksForGeeks, <https://www.geeksforgeeks.org/software-design-patterns/>.
  3. Source Making, “Design Patterns”, Source Making, <https://sourcemaking.com/design_patterns>.