TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP LỚN CUỐI KÌ MÔN MẪU THIẾT KẾ**

**ÁP MẪU DỤNG THIẾT KẾ VÀO**

**PHẦN MỀM QUẢN LÝ NHÀ HÀNG**

*Người hướng dẫn*: **THS VŨ ĐÌNH HỒNG**

*Người thực hiện*: **NGUYỄN TRIỆU VI– 52100143**

**NGUYỄN ĐĂNG HUY VŨ – 52100144**

**LƯƠNG MAI THANH KHÔI**

Lớp **: 21050201**

Khoá  **: 25**

**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**



**BÀI TẬP LỚN CUỐI KÌ MÔN MẪU THIẾT KẾ**

**ÁP DỤNG MẪU THIẾT KẾ VÀO**

**PHẦN MỀM QUẢN LÝ NHÀ HÀNG**

Người hướng dẫn: **THS VŨ ĐÌNH HỒNG**

Người thực hiện: **NGUYỄN TRIỆU VI**

**NGUYỄN ĐĂNG HUY VŨ**

**LƯƠNG MAI THANH KHÔI**

Lớp **: 21050201**

Khoá  **: 25**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024**

# LỜI CẢM ƠN

Chúng em xin gửi lời cảm ơn chân thành đến giảng viên bộ môn – Thầy Vũ Đình Hồng, đã dạy dỗ và tâm huyết truyền đạt những kiến thức quý giá cho chúng em trong suốt thời gian học tập vừa qua. Thầy là người đã mở rộng vốn kiến thức của chúng em từ lý thuyết để áp dụng vào những vấn đề thực tế trong chuyên ngành, bên cạnh đó là những lời định hướng hết sức thiết thực cho tương lai của chúng em. Môn học mẫu thiết kế là môn học thú vị, bổ ích và có tính thực tế cao. Đảm bảo cung cấp đầy đủ kiến thức, kỹ năng, giúp sinh viên có thể ứng dụng vào thực tế. Tuy nhiên, do khả năng tiếp thu thực tế còn nhiều hạn hẹp, kiến thức chưa sâu rộng. Mặc dù bản thân đã cố gắng hết sức nhưng chắc chắn bài báo cáo khó tránh khỏi những thiếu sót, kính mong quý thầy cô xem xét và góp ý để bài báo cáo của chúng em được hoàn thiện và tốt hơn.

Chúng em xin chân thành cảm ơn!

**ĐỒ ÁN ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm đồ án của riêng chúng tôi và được sự hướng dẫn của TS Nguyễn Văn A;. 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 đồ 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 tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày tháng năm*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Nguyễn Đăng Huy Vũ*

*Nguyễn Triệu Vi*

*Lương Mai Thanh Khôi*

**PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN**

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

# MỤC LỤC

[LỜI CẢM ƠN 1](#_Toc165413220)

[MỤC LỤC 1](#_Toc165413221)

[CHƯƠNG 1 – CÁC MẪU THIẾT KẾ ĐÃ SỬ DỤNG 4](#_Toc165413222)

[1.1 Factory Pattern 4](#_Toc165413223)

[*1.1.1 Sơ đồ lớp* 4](#_Toc165413224)

[*1.1.2 Lý do áp dụng* 4](#_Toc165413225)

[1.2 State Pattern 5](#_Toc165413226)

[*1.2.1 Sơ đồ lớp* 5](#_Toc165413227)

[*1.2.2 Lý do áp dụng* 5](#_Toc165413228)

[1.3 Singleton Pattern 6](#_Toc165413229)

[*1.3.1 Sơ đồ lớp* 6](#_Toc165413230)

[*1.3.2 Lý do áp dụng* 6](#_Toc165413231)

[1.4 Command Pattern 7](#_Toc165413232)

[*1.4.1 Sơ đồ lớp* 7](#_Toc165413233)

[*1.4.2 Lý do áp dụng* 7](#_Toc165413234)

[1.5 Proxy Pattern 8](#_Toc165413235)

[*1.5.1 Sơ đồ lớp* 8](#_Toc165413236)

[1.6 Strategy Pattern 9](#_Toc165413237)

[*1.6.1 Sơ đồ lớp* 9](#_Toc165413238)

[*1.6.2 Lý do áp dụng* 9](#_Toc165413239)

[1.7 Template Method Pattern 10](#_Toc165413240)

[*1.7.1 Sơ đồ lớp* 10](#_Toc165413241)

[*1.7.2 Lý do áp dụng* 10](#_Toc165413242)

[1.8 Observer Pattern 11](#_Toc165413243)

[*1.8.1 Sơ đồ lớp* 11](#_Toc165413244)

[*1.8.2 Lý do áp dụng* 11](#_Toc165413245)

[CHƯƠNG 2 – CODE CHI TIẾT TRONG ỨNG DỤNG 12](#_Toc165413246)

[2.1 Factory Pattern 12](#_Toc165413247)

[2.2 State Pattern 15](#_Toc165413248)

[2.3 Singleton Pattern 16](#_Toc165413249)

[2.4 Command Pattern 17](#_Toc165413250)

[2.5 Proxy Pattern 19](#_Toc165413251)

[2.6 Strategy Pattern 21](#_Toc165413252)

[2.7 Template Method Pattern 23](#_Toc165413253)

[TÀI LIỆU THAM KHẢO 26](#_Toc165413254)

**DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ**

**DANH MỤC HÌNH**

[Hình 1: Sơ đồ lớp Factory Pattern 4](#_Toc165413205)

[Hình 2: Sơ đồ lớp State Pattern 5](#_Toc165413206)

[Hình 3: Sơ đồ lớp Singleton Pattern 6](#_Toc165413207)

[Hình 4: Sơ đồ lớp Command Pattern 7](#_Toc165413208)

[Hình 5: Sơ đồ lớp Proxy Pattern 8](#_Toc165413209)

[Hình 6: Sơ đồ lớp Strategy Pattern 9](#_Toc165413210)

[Hình 7: Sơ đồ lớp Template method Pattern 10](#_Toc165413211)

[Hình 8: Sơ đồ lớp Observer Pattern 11](#_Toc165413212)

[Hình 9: Code Factory Pattern 15](#_Toc165413213)

[Hình 10: Code State Pattern 16](#_Toc165413214)

[Hình 11: Code Singleton Pattern 17](#_Toc165413215)

[Hình 12: Code Command Pattern 19](#_Toc165413216)

[Hình 13: Code Proxy Pattern 21](#_Toc165413217)

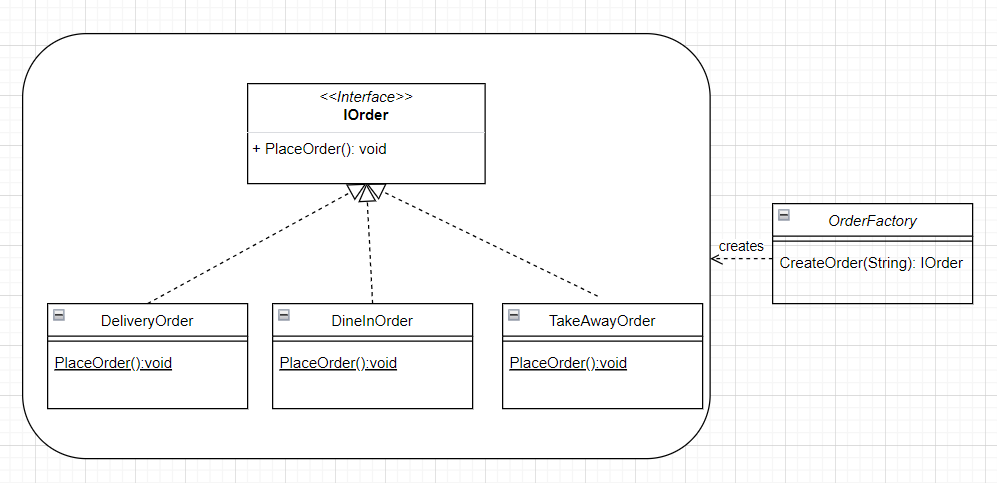
[Hình 14: Code Strategy Pattern 23](#_Toc165413218)

[Hình 15: Code Template Method pattern 25](#_Toc165413219)

# CHƯƠNG 1 – CÁC MẪU THIẾT KẾ ĐÃ SỬ DỤNG

## 1.1 Factory Pattern

### *1.1.1 Sơ đồ lớp*



Hình 1: Sơ đồ lớp Factory Pattern

### *1.1.2 Lý do áp dụng*

Factory pattern là một mẫu thiết kế hành vi cho phép bạn tạo ra các đối tượng mà không cần chỉ định loại đối tượng cụ thể sẽ được tạo ra. Thay vào đó, việc tạo đối tượng được ủy quyền cho một lớp nhà máy (Factory class).

Thay vì viết code xử lý đơn hàng riêng biệt cho từng loại đơn hàng (Delivery, TakeAway, DineIn), ta có thể sử dụng Factory Pattern để tạo ra một interface chung IOrder và các lớp con riêng biệt cho từng loại đơn hàng. Điều này giúp tránh trùng lặp code đáng kể.

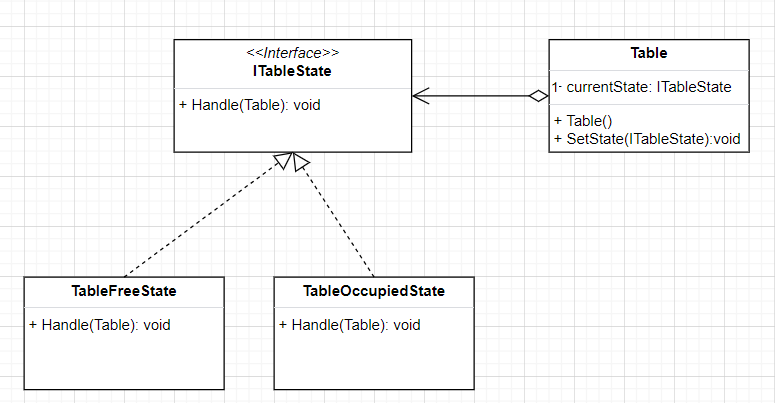
Việc thêm loại đơn hàng mới trở nên đơn giản hơn. Chỉ cần tạo thêm một lớp con mới implement interface IOrder mà không cần phải sửa đổi code chính của chương trình.

Tách biệt logic xử lý đơn hàng khỏi logic chính của chương trình giúp cho code dễ đọc và dễ maintain hơn. Việc sửa đổi code xử lý đơn hàng không ảnh hưởng đến logic chính của chương trình.  
 Logic xử lý đơn hàng được tách biệt khỏi logic chính của chương trình, giúp cho code dễ hiểu và dễ maintain hơn, đồng thời cũng giúp cho việc test code dễ dàng hơn.

Factory Pattern cho phép bạn tạo các đối tượng khác nhau dựa trên cùng một interface. Điều này giúp cho bạn có thể dễ dàng thay đổi cách tạo đối tượng mà không cần phải sửa đổi code chính của chương trình.

## 1.2 State Pattern

### *1.2.1 Sơ đồ lớp*



Hình 2: Sơ đồ lớp State Pattern

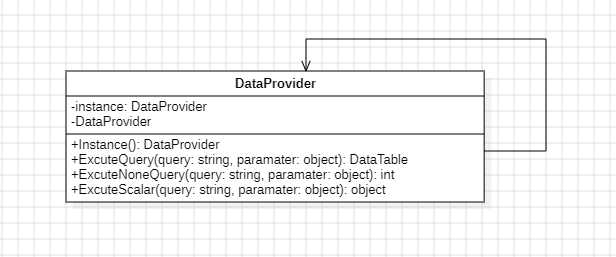
### *1.2.2 Lý do áp dụng*

State pattern là một design pattern được sử dụng để quản lý trạng thái của một đối tượng. Trong trường hợp quản lý bàn, state pattern mang lại nhiều lợi ích:

* Mỗi bàn có thể ở một trong nhiều trạng thái khác nhau, chẳng hạn như trống, đang phục vụ.
* State pattern giúp bạn dễ dàng quản lý trạng thái của mỗi bàn bằng cách sử dụng các lớp state riêng biệt cho mỗi trạng thái.
* Mỗi lớp state sẽ có các phương thức riêng để xử lý các hành động liên quan đến trạng thái đó (ví dụ: đặt bàn, phục vụ, thanh toán).
* Có thể dễ dàng thêm các trạng thái mới cho bàn mà không cần phải sửa đổi code của các lớp state hiện có.
* Chỉ cần tạo một lớp state mới cho trạng thái mới và implement các phương thức cần thiết.

## 1.3 Singleton Pattern

### *1.3.1 Sơ đồ lớp*

****

Hình 3: Sơ đồ lớp Singleton Pattern

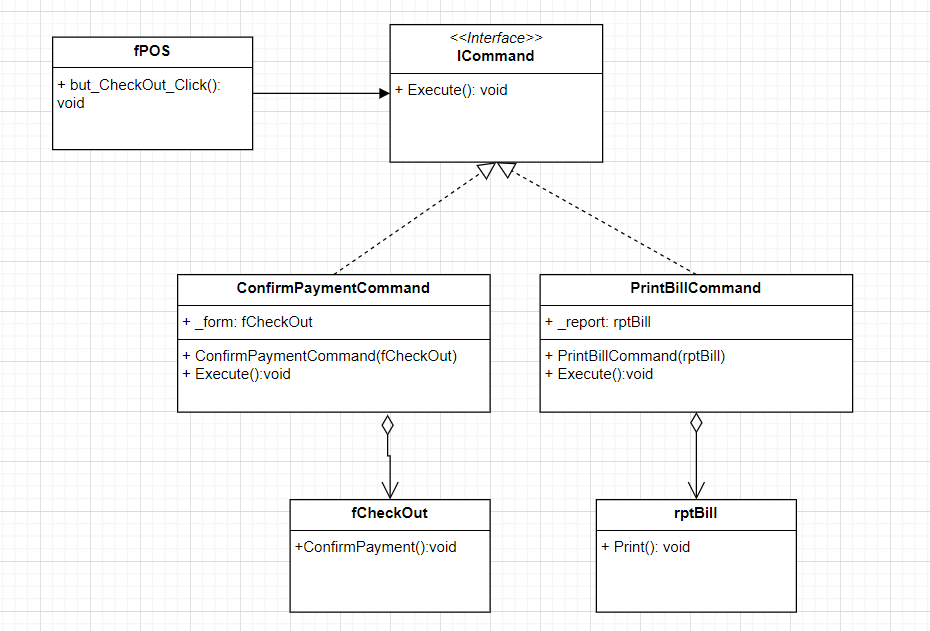
### *1.3.2 Lý do áp dụng*

Singleton là một mẫu thiết kế sáng tạo, đảm bảo rằng chỉ tồn tại một đối tượng cùng loại và cung cấp một điểm truy cập duy nhất vào đối tượng đó cho bất kỳ mã nào khác.

Áp dụng Singleton Pattern vào dự án để một lần lấy dữ liệu từ database chỉ duy nhất một connectionString được kết nối cơ sở dữ liệu.

## 1.4 Command Pattern

### *1.4.1 Sơ đồ lớp*



Hình 4: Sơ đồ lớp Command Pattern

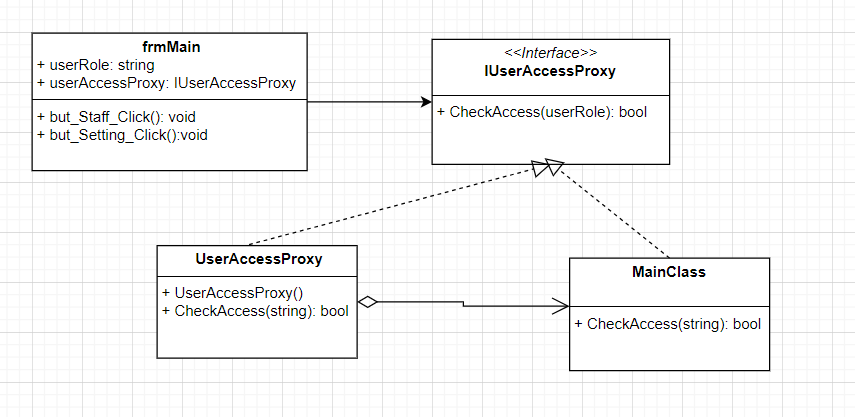
### *1.4.2 Lý do áp dụng*

Command Pattern giúp code dễ dàng mở rộng và bảo trì. Thay vì viết logic xử lý thanh toán và in hóa đơn trực tiếp trong form fCheckOut, có thể tạo các lớp riêng biệt implement interface ICommand để xử lý logic này. Điều này giúp bạn dễ dàng thay đổi logic xử lý thanh toán và in hóa đơn mà không cần phải sửa đổi code chính của form.

Command Pattern giúp code dễ dàng test. Có thể dễ dàng test logic xử lý thanh toán và in hóa đơn bằng cách mock các lớp implement ICommand.

## 1.5 Proxy Pattern

### *1.5.1 Sơ đồ lớp*



Hình 5: Sơ đồ lớp Proxy Pattern

***1.5.2 Lý do áp dụng***

Sử dụng Proxy Pattern để kiểm soát quyền truy cập của người dùng vào các chức năng khác nhau trong ứng dụng.

Biến userRole lưu trữ vai trò của người dùng hiện tại.

Phương thức CheckAccess của lớp UserAccessProxy được sử dụng để kiểm tra xem người dùng có quyền truy cập vào chức năng được yêu cầu hay không.

Nếu người dùng có quyền truy cập, code sẽ gọi đến phương thức AddControls để hiển thị chức năng tương ứng.

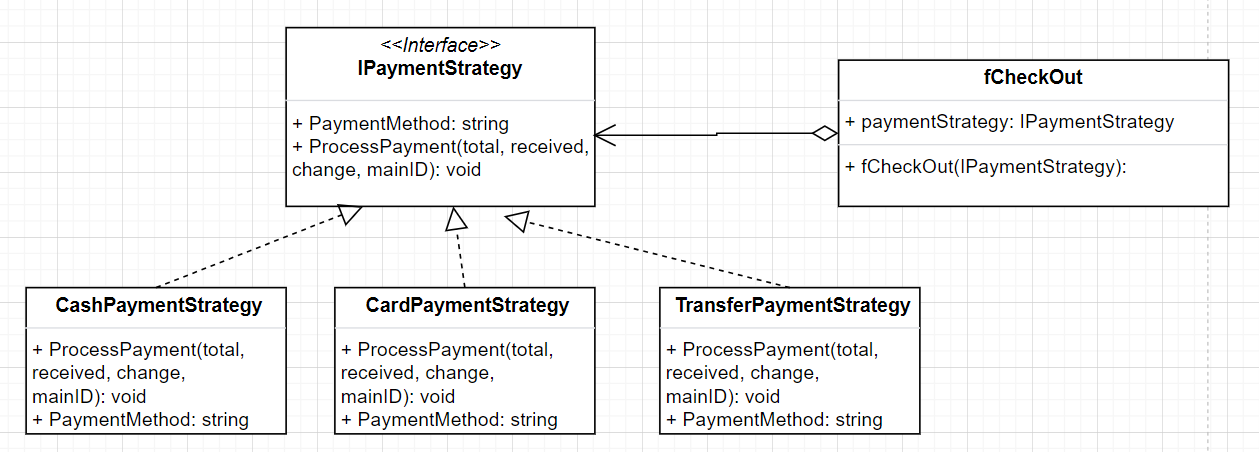
Nếu người dùng không có quyền truy cập, code sẽ hiển thị thông báo lỗi.

Proxy Pattern giúp tách biệt logic kiểm tra quyền truy cập khỏi code chính của ứng dụng. Điều này giúp code dễ maintain và mở rộng hơn. Có thể dễ dàng thay đổi logic kiểm tra quyền truy cập mà không cần phải sửa đổi code chính của ứng dụng.

**⇒** Proxy Pattern được sử dụng trong đoạn code này để kiểm soát quyền truy cập, tách biệt logic kiểm tra quyền truy cập

## 1.6 Strategy Pattern

### *1.6.1 Sơ đồ lớp*



Hình 6: Sơ đồ lớp Strategy Pattern

### *1.6.2 Lý do áp dụng*

Việc thêm phương thức thanh toán mới trở nên dễ dàng hơn. Bạn chỉ cần tạo một lớp mới implement interface IPaymentStrategy và không cần sửa đổi các lớp hiện có.

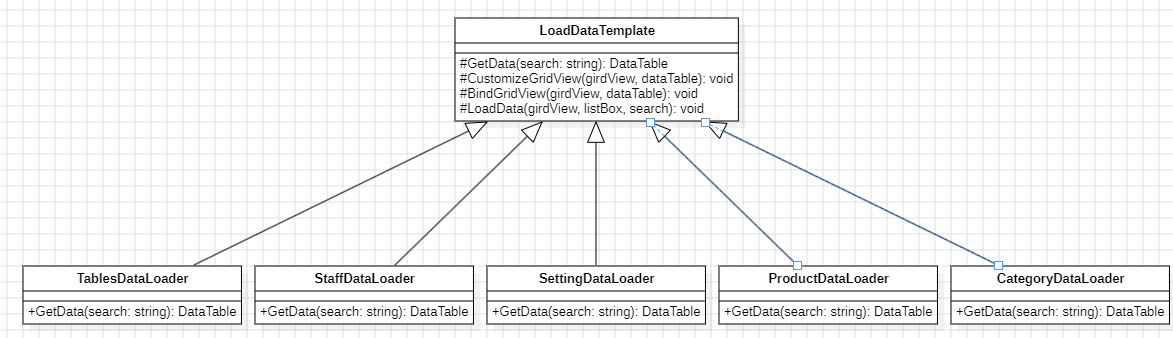
Việc sửa lỗi hoặc cập nhật logic của một phương thức thanh toán chỉ cần sửa đổi lớp tương ứng, mà không ảnh hưởng đến các lớp khác.

Mỗi lớp implement IPaymentStrategy có thể được test riêng biệt, đảm bảo tính chính xác của từng phương thức thanh toán.

Các lớp implement IPaymentStrategy có thể được tái sử dụng trong các ứng dụng khác cần xử lý thanh toán.

## 1.7 Template Method Pattern

### *1.7.1 Sơ đồ lớp*



Hình 7: Sơ đồ lớp Template method Pattern

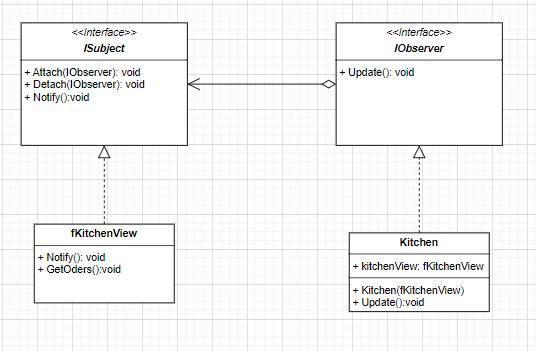
### *1.7.2 Lý do áp dụng*

Template Method Pattern xác định khung của một thuật toán trong một phương thức, trì hoãn một số bước cho các lớp con. Phương thức mẫu cho phép các lớp con xác định lại các bước nhất định của thuật toán mà không thay đổi cấu trúc của thuật toán.

Các thuật toán có cấu trúc giống nhau, lập lại nhiều lần trên dự án.

## 1.8 Observer Pattern

### *1.8.1 Sơ đồ lớp*



Hình 8: Sơ đồ lớp Observer Pattern

### *1.8.2 Lý do áp dụng*

fKitchenView là đối tượng chịu trách nhiệm quản lý các đơn hàng trong nhà hàng. Khi có một đơn hàng mới được thêm vào, fKitchenView cần thông báo cho các đối tượng bếp biết để họ có thể chuẩn bị các món trong đơn hàng đó.

Điều này giúp tăng tính linh hoạt và tái sử dụng trong mã, và dễ dàng mở rộng hệ thống trong tương lai nếu cần thiết.

# CHƯƠNG 2 – CODE CHI TIẾT TRONG ỨNG DỤNG

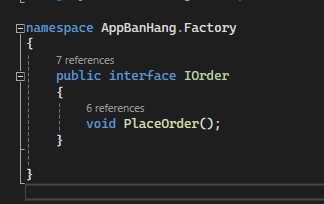
## 2.1 Factory Pattern

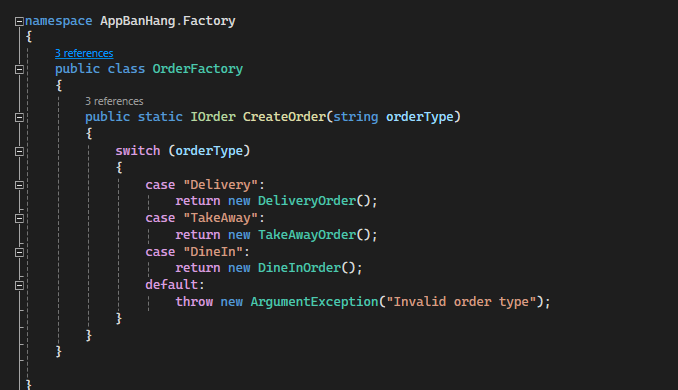
Trong đoạn code ta có thể thấy rõ ràng sự trùng lặp code trong các lớp DeliveryOrder, TakeAwayOrder và DineInOrder. Các lớp này đều có phương thức PlaceOrder với nội dung tương tự nhau. Sử dụng Factory Pattern giúp ta tách biệt logic xử lý đơn hàng khỏi logic chính của chương trình và tránh trùng lặp code.

Interface IOrder: Interface này định nghĩa các phương thức chung cho tất cả các loại đơn hàng.

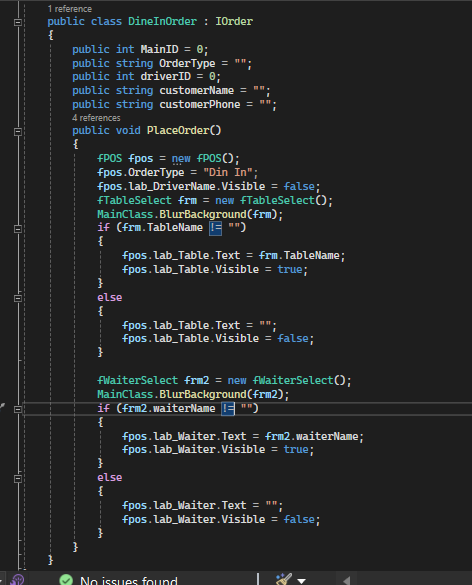
Các lớp con của IOrder: Mỗi lớp con đại diện cho một loại đơn hàng cụ thể, chẳng hạn như DeliveryOrder, TakeAwayOrder và DineInOrder.

Lớp OrderFactory: Lớp này có phương thức CreateOrder để tạo các đối tượng đơn hàng dựa trên loại đơn hàng được yêu cầu.









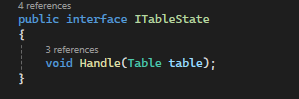


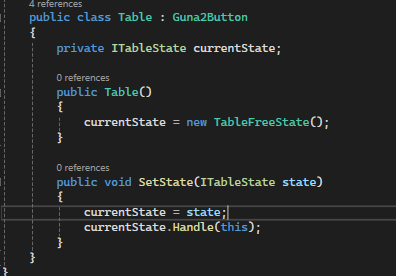
Hình 9: Code Factory Pattern

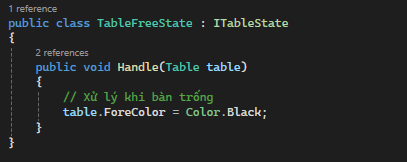
## 2.2 State Pattern

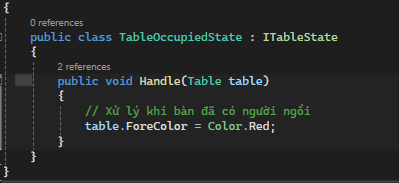
Lớp Table đại diện cho một bàn trong nhà hàng.   
 Giao diện ITableState định nghĩa các phương thức mà một đối tượng trạng thái phải triển khai.

Lớp TableFreeState biểu thị trạng thái của một bàn trống và có sẵn cho khách hàng.  
 Lớp TableOccupiedState biểu thị trạng thái của một bàn đã có khách hàng ngồi.  
 Handle(Table table): Phương thức đặt màu chữ của bàn thành màu đỏ, biểu thị rằng bàn đã có người ngồi.Phương thức màu nền của bàn thành màu đen, biểu thị rằng bàn trống.



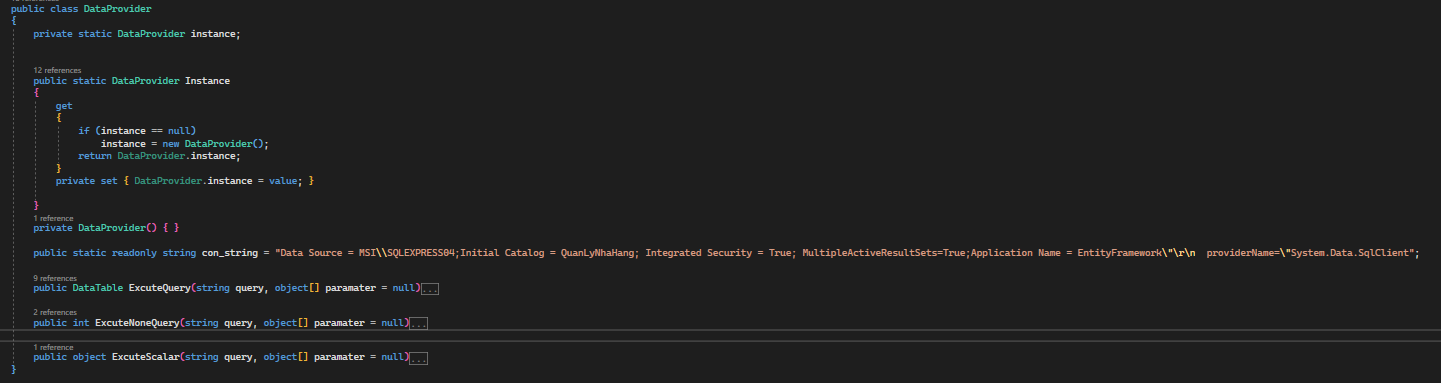


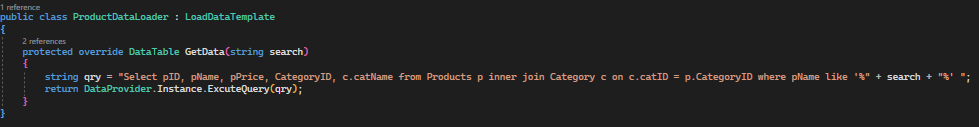




Hình 10: Code State Pattern

## 2.3 Singleton Pattern

****

****

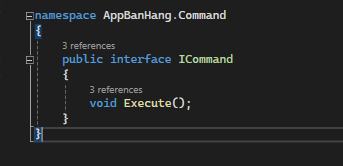
Hình 11: Code Singleton Pattern

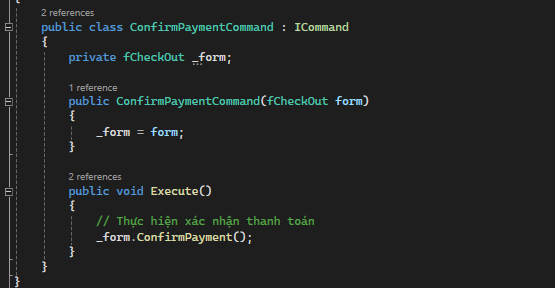
## 2.4 Command Pattern

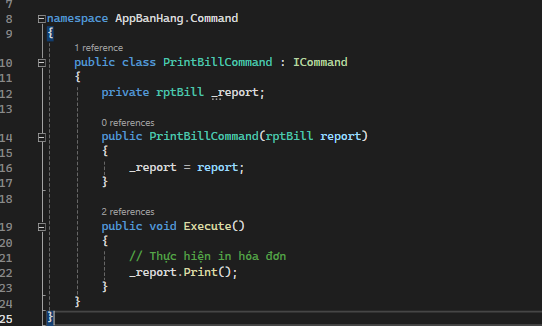
Trong bài toán thanh toán, bạn có thể sử dụng Command Pattern để tách biệt logic của command xác nhận thanh toán và logic của command in hóa đơn.

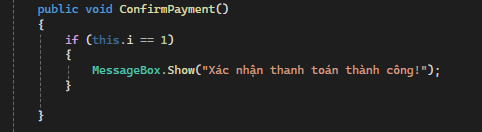
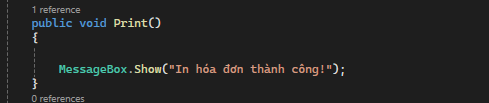
* Lớp ICommand: Interface định nghĩa phương thức Execute() mà các lớp implement ICommand phải implement.
* Lớp ConfirmPaymentCommand: Lớp implement ICommand và thực hiện logic xác nhận thanh toán.
* Lớp PrintBillCommand: Lớp implement ICommand và thực hiện logic in hóa đơn.
* Lớp fCheckOut: Lớp có phương thức ConfirmPayment() để thực hiện logic xác nhận thanh toán.
* Lớp rptBill: Lớp có phương thức Print() để thực hiện logic in hóa đơn.

Để sử dụng Command Pattern trong bài toán thanh toán, cần thực hiện: Khởi tạo một instance của lớp ICommand phù hợp với command bạn muốn thực hiện. Gọi phương thức Execute() của instance ICommand.









Hình 12: Code Command Pattern

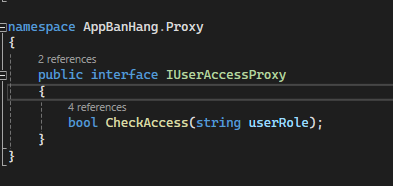
## 2.5 Proxy Pattern

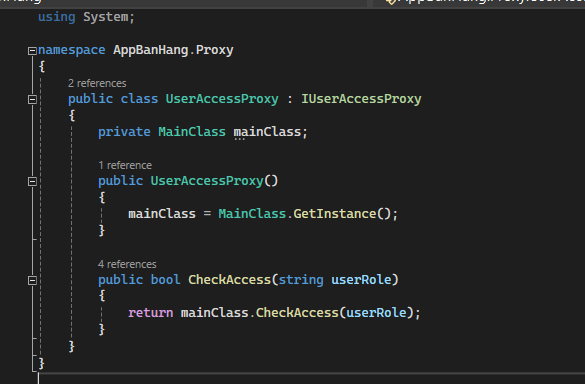
Proxy Pattern được sử dụng để kiểm soát quyền truy cập của người dùng vào các chức năng khác nhau trong ứng dụng.

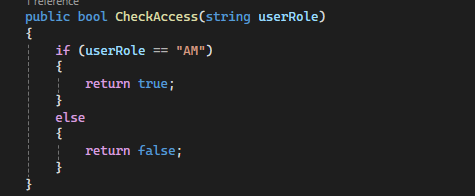
Lớp UserAccessProxy được sử dụng để tách biệt logic kiểm tra quyền truy cập khỏi code chính của ứng dụng.

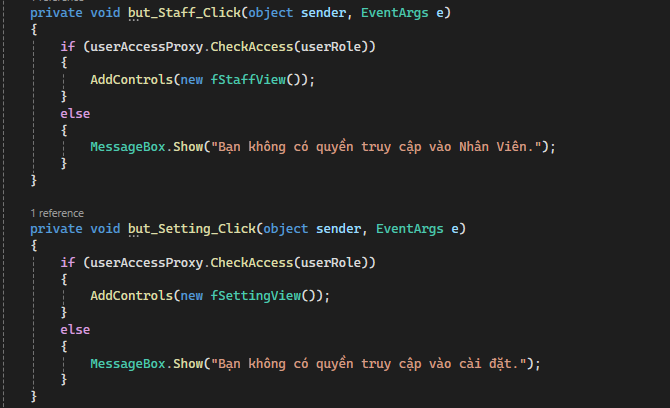
Lớp MainClass chứa logic kiểm tra quyền truy cập của người dùng.

IUserAccessProxy định nghĩa hàm CheckAccess để kiểm tra quyền truy cập của người dùng.









Hình 13: Code Proxy Pattern

## 2.6 Strategy Pattern

Có 3 lớp implement IPaymentStrategy: TransferPaymentStrategy, CashPaymentStrategy, và CardPaymentStrategy. Mỗi lớp có phương thức ProcessPayment() riêng để xử lý logic thanh toán theo phương thức tương ứng.

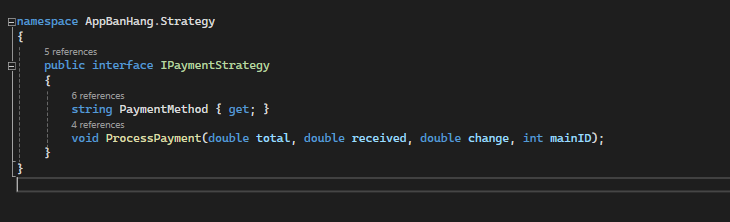
IPaymentStrategy được sử dụng để khai báo biến paymentStrategy. Biến này có thể được gán với bất kỳ implementation cụ thể nào của IPaymentStrategy, cho phép bạn dễ dàng thay đổi phương thức thanh toán được sử dụng.

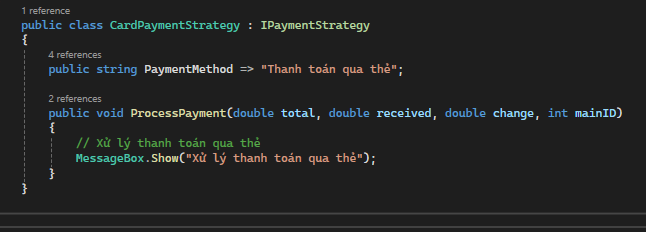
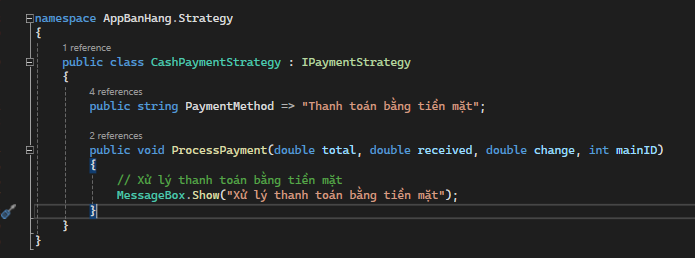
Lớp IPaymentStrategy là một giao diện định nghĩa các phương thức mà một chiến lược thanh toán phải triển khai.

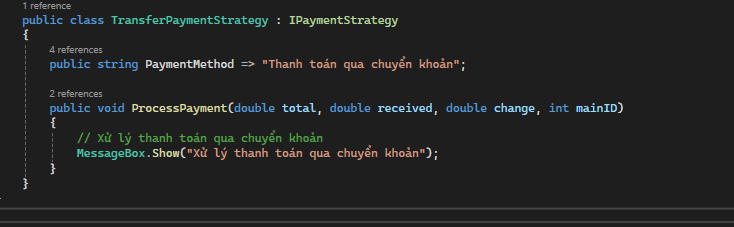
Lớp TransferPaymentStrategy là một lớp triển khai IPaymentStrategy cho phương thức thanh toán qua chuyển khoản.

Lớp CashPaymentStrategy là một lớp triển khai IPaymentStrategy cho phương thức thanh toán bằng tiền mặt.

Lớp CardPaymentStrategy là một lớp triển khai IPaymentStrategy cho phương thức thanh toán qua thẻ.



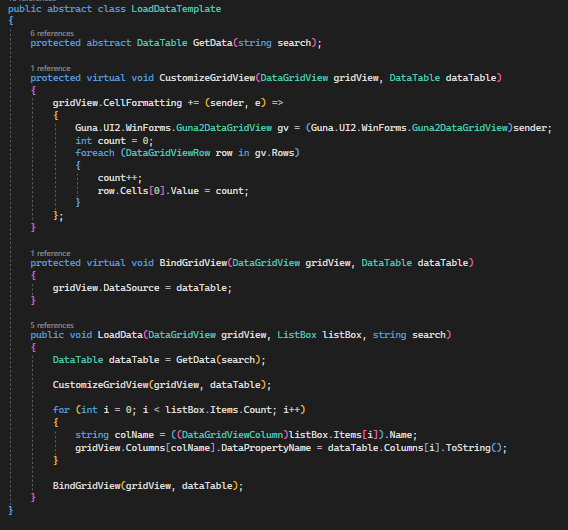


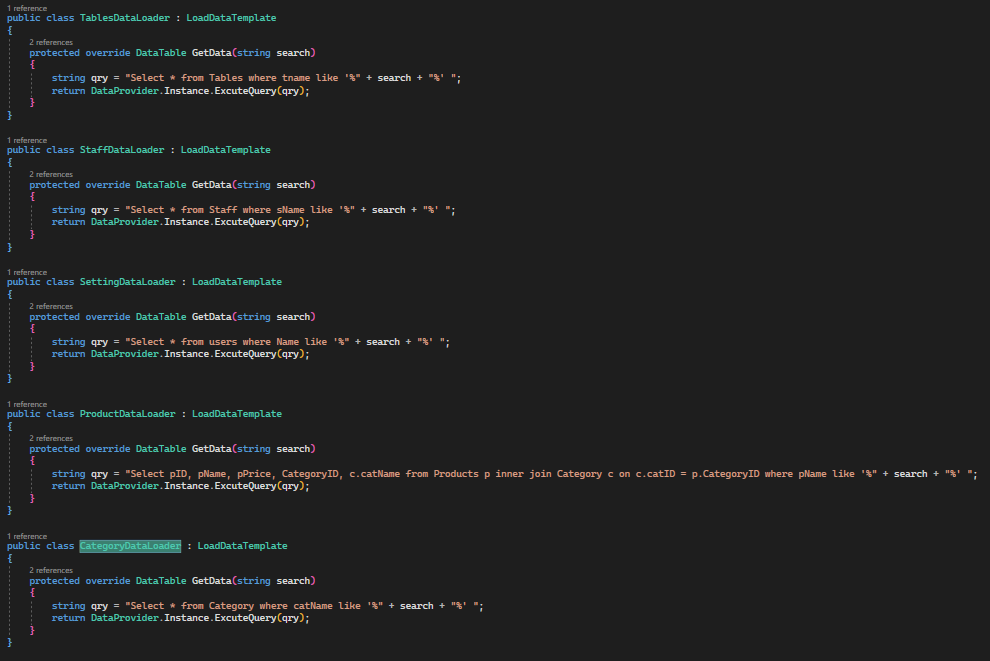


Hình 14: Code Strategy Pattern

## 2.7 Template Method Pattern

Template Method dùng để giải quyết các vấn đề lấy dữ liệu, và chỉ khác nhau vài bước.

****

****

Hình 15: Code Template Method pattern

# TÀI LIỆU THAM KHẢO