**TRƯỜNG ĐẠI HỌC ĐÀ LẠT**

**KHOA CÔNG NGHỆ THÔNG TIN**



**MẪU THIẾT KẾ**

Giáo viên hướng dẫn: Đoàn Minh Khuê

Sinh viên thực hiện: 2212431-Phạm Tống Thái Nhân

*Đà Lạt,24 tháng 10 năm 2025*

**MỤC LỤC**

[**1. Giới thiệu đề tài** 3](#_Toc212217899)

[**2. Phân tích hệ thống** 3](#_Toc212217900)

[**3. Thiết kế hệ thống** 3](#_Toc212217901)

[**4. UML Class Diagram** 4](#_Toc212217902)

[**5. Mô tả chi tiết các lớp** 4](#_Toc212217903)

[**6. Kết quả thực thi** 4](#_Toc212217904)

[**7. Hướng phát triển** 5](#_Toc212217905)

[**8. Kết luận** 5](#_Toc212217906)

[**9. Tài liệu tham khảo** 5](#_Toc212217907)

# **1. Giới thiệu đề tài**

Đề tài *Coffee Shop Manager* được xây dựng nhằm mô phỏng hoạt động quản lý quán cà phê dưới dạng ứng dụng console.  
Chương trình cho phép người dùng:

* Đăng nhập vào chương trình (admin/staff).
* Xem danh sách menu đồ uống.
* Tạo đơn hàng mới và thêm topping.
* Lưu đơn hàng vào file dữ liệu.
* Xem thống kê doanh thu.  
  Đề tài minh họa ba mẫu thiết kế phần mềm quan trọng trong Java: **Singleton**, **Strategy** và **Decorator**, giúp chương trình dễ mở rộng, bảo trì và quản lý dữ liệu hiệu quả hơn

# **2. Phân tích hệ thống**

Hệ thống được phát triển theo mô hình hướng đối tượng (OOP), bao gồm các thành phần chính:

* **MenuService:** Quản lý danh sách đồ uống, hiển thị menu và cập nhật sản phẩm.
* **OrderService:** Xử lý việc tạo đơn hàng, tính tổng tiền, thống kê doanh thu, xuất hóa đơn.
* **FileService (Singleton):** Quản lý đọc/ghi dữ liệu từ file orders.txt để lưu trữ lịch sử đơn hàng.
* **PaymentStrategy (Strategy):** Định nghĩa cách tính tổng tiền khác nhau (ví dụ: thanh toán thường, giảm giá theo khung giờ, hoặc ưu đãi thành viên).
* **DrinkDecorator (Decorator):** Cho phép thêm topping như sữa, kem, caramel vào đồ uống mà không cần chỉnh sửa lớp gốc.

# **3. Thiết kế hệ thống**

Hệ thống áp dụng ba mẫu thiết kế cốt lõi:

1. **Singleton Pattern:** Quản lý việc ghi dữ liệu đơn hàng vào file để tránh xung đột truy cập.
2. **Strategy Pattern:** Tách riêng thuật toán tính giá đơn hàng để có thể thay đổi linh hoạt.
3. **Decorator Pattern:** Giúp mở rộng thuộc tính của đồ uống bằng cách thêm topping mà không cần tạo nhiều lớp con.

Các lớp chính gồm: Drink, Coffee, MilkTea, Order, MenuService, OrderService, FileService, và các lớp topping mở rộng như MilkDecorator, CreamDecorator.

# **4. UML Class Diagram**

Sơ đồ UML thể hiện các mối quan hệ:

* MenuService và OrderService là hai lớp nghiệp vụ chính.
* FileService là **Singleton**, được gọi từ nhiều nơi để ghi và đọc file.
* PaymentStrategy là interface, các lớp như NormalPayment, OnlinePayment triển khai chiến lược thanh toán cụ thể.
* Drink là lớp trừu tượng, được kế thừa và mở rộng bằng DrinkDecorator và các lớp topping cụ thể.

# **5. Mô tả chi tiết các lớp**

 **Drink (abstract):** Cung cấp cấu trúc cho tất cả đồ uống, gồm getDescription() và getCost().

* **Coffee / MilkTea:** Cụ thể hóa đồ uống với giá cơ bản.
* **DrinkDecorator:** Cho phép thêm topping động như sữa, kem hoặc caramel.
* **FileService (Singleton):** Ghi/lưu đơn hàng vào file orders.txt.
* **PaymentStrategy:** Giao diện tính tổng tiền đơn hàng.
* **OrderService:** Quản lý đơn hàng, gọi FileService và chiến lược tính giá.

# **6. Kết quả thực thi**

Chương trình chạy trên **VS Code**, hiển thị menu và lựa chọn qua **console**.  
Khi người dùng tạo đơn hàng:

* Hệ thống hiển thị menu và cho phép chọn đồ uống, topping, hình thức thanh toán.
* Tổng tiền được tính toán theo chiến lược tương ứng.
* Dữ liệu được lưu vào file orders.txt.
* Doanh thu tổng hợp hiển thị chính xác trên màn hình.

# **7. Hướng phát triển**

 Tích hợp cơ sở dữ liệu (MySQL hoặc SQLite) để lưu đơn hàng và menu.

 Xây dựng giao diện đồ họa (JavaFX hoặc Swing).

 Thêm đăng nhập và phân quyền người dùng.

 Mở rộng hệ thống bằng các pattern khác như **Factory** hoặc **Observer** để quản lý nhiều loại sản phẩm hơn

# **8. Kết luận**

Đề tài đạt được mục tiêu mô phỏng quản lý quán cà phê đồng thời minh họa 3 mẫu thiết kế phổ biến trong Java.  
Việc áp dụng **Singleton**, **Strategy** và **Decorator** giúp mã nguồn dễ mở rộng, bảo trì và linh hoạt cho các cải tiến trong tương lai.

# **9. Tài liệu tham khảo**

 *Design Patterns: Elements of Reusable Object-Oriented Software* – Erich Gamma et al.

 *Head First Design Patterns* – Freeman & Freeman.

 Tài liệu hướng dẫn Java – Oracle: <https://docs.oracle.com/javase/>

 Giáo trình “Mẫu Thiết Kế Phần Mềm” – Khoa CNTT, Đại học Đà Lạt.