**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** 4](#_Toc212217901)

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

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

[**6. Kết quả thực thi** 5](#_Toc212217904)

[**7. Hướng phát triển** 6](#_Toc212217905)

[**8. Kết luận** 6](#_Toc212217906)

[**9. Tài liệu tham khảo** 6](#_Toc212217907)

# **1. Giới thiệu đề tài**

Đề tài **Coffee Shop Manager** được xây dựng nhằm mô phỏng hoạt động quản lý quán cà phê thông qua ứng dụng **Java console**.  
Chương trình cho phép người dùng thao tác trực tiếp trên màn hình dòng lệnh để quản lý sản phẩm, đơn hàng và doanh thu.

**Các chức năng chính:**

* Đăng nhập vào hệ thống (phân quyền *Admin* và *Nhân viên*).
* Hiển thị danh sách menu đồ uống và giá.
* Thêm, xóa, sửa giá món trong menu (chức năng dành cho *Admin*).
* Tạo đơn hàng mới, chọn topping (Milk, Caramel, Ice...).
* Áp dụng các chiến lược tính giá khác nhau (Regular, Member, HappyHour, Combo).
* Lưu đơn hàng vào file dữ liệu orders.txt.
* Xem thống kê doanh thu và xuất hóa đơn ra file .txt.

Hệ thống minh họa ba mẫu thiết kế phần mềm phổ biến 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)**, chia thành các lớp theo chức năng:

* **MenuService**:  
  Quản lý danh sách sản phẩm (thêm, xóa, sửa, hiển thị menu), lưu menu vào file menu.txt.
* **OrderService**:  
  Tạo và xử lý đơn hàng, tính giá theo chiến lược, thống kê doanh thu và xuất hóa đơn.
* **ProductDAO**:  
  Đảm nhiệm việc đọc và ghi dữ liệu menu từ file (cấu trúc DAO – Data Access Object).
* **DatabaseManager (Singleton)**:  
  Quản lý đọc/ghi dữ liệu đơn hàng vào file orders.txt, đảm bảo chỉ có một thể hiện duy nhất truy cập file cùng lúc.
* **PricingStrategy (Strategy)**:  
  Định nghĩa các cách tính giá linh hoạt: RegularPricing, MemberPricing, HappyHourPricing, và BulkOrderPricing.
* **Product / Decorator (Decorator Pattern)**:  
  Cho phép mở rộng đồ uống bằng cách thêm topping như sữa, caramel, hoặc đá mà không cần tạo nhiều lớp con.

# **3. Thiết kế hệ thống**

Hệ thống áp dụng 3 mẫu thiết kế chính:

1. **Singleton Pattern**  
   → Áp dụng cho lớp DatabaseManager để quản lý file đơn hàng một cách an toàn và tránh trùng ghi dữ liệu.
2. **Strategy Pattern**  
   → Giúp thay đổi thuật toán tính giá đơn hàng (giảm giá theo thời điểm, thành viên, combo, v.v.) mà không ảnh hưởng đến cấu trúc chương trình.
3. **Decorator Pattern**  
   → Giúp thêm topping (Milk, Caramel, Ice) vào sản phẩm gốc mà không cần chỉnh sửa lớp Product gốc.

**Các lớp chính:**

* Product, Coffee, Tea, Snack
* MilkDecorator, CaramelDecorator, IceDecorator
* MenuService, OrderService, ProductDAO, DatabaseManager
* RegularPricing, MemberPricing, HappyHourPricing, BulkOrderPricing

# **4. UML Class Diagram**

 **MenuService** ↔ **ProductDAO**: tương tác để tải và lưu menu.

 **OrderService** ↔ **DatabaseManager (Singleton)**: ghi đơn hàng vào orders.txt.

 **PricingStrategy** là **interface**, được kế thừa bởi các lớp chiến lược giá cụ thể.

 **Product** là lớp cha trừu tượng, được mở rộng bằng **Decorator** để thêm topping động.

# **5. Mô tả chi tiết các lớp**

* **Product (abstract)**: Cung cấp cấu trúc cho sản phẩm, gồm getName(), cost(), và mô tả sản phẩm.
* **Coffee / Tea / Snack**: Triển khai cụ thể từng loại sản phẩm cơ bản.
* **MilkDecorator / CaramelDecorator / IceDecorator**: Gói thêm topping vào sản phẩm hiện có.
* **PricingStrategy (interface)**: Định nghĩa phương thức tính giá linh hoạt.
* **OrderService**:
* Cho phép chọn món từ menu
* Thêm topping
* Áp dụng chiến lược tính giá
* Ghi hóa đơn vào orders.txt
* Xuất file hóa đơn receipt\_xxx.txt
* **MenuService**: Quản lý menu, cho phép thêm, sửa, xóa món, cập nhật vào menu.txt.
* **DatabaseManager (Singleton)**: Ghi và đọc file đơn hàng.

# **6. Kết quả thực thi**

Chương trình chạy trên môi trường **VS Code / CMD console**, giao diện text thân thiện.

**Quy trình chạy mẫu:**

* Người dùng đăng nhập (Admin hoặc Staff).
* Admin có thể quản lý menu: thêm, sửa giá, xóa món.
* Người dùng tạo đơn hàng: chọn món, topping, số lượng, và loại giảm giá.
* Hệ thống tính toán tổng tiền, giảm giá, và xuất hóa đơn chi tiết.
* Dữ liệu đơn hàng được lưu trong orders.txt.
* Thống kê doanh thu tổng hợp từ file hiển thị trên console.

# **7. Hướng phát triển**

* Tích hợp cơ sở dữ liệu **SQLite hoặc MySQL** để lưu trữ menu và đơn hàng thay vì file text.
* Xây dựng **giao diện đồ họa (JavaFX / Swing)** để người dùng thao tác trực quan hơn.
* Mở rộng đăng nhập, phân quyền người dùng, và thống kê theo ngày/tháng.
* Áp dụng thêm **Factory Pattern** để tạo sản phẩm động.
* Kết nối mạng nội bộ (LAN) cho phép nhiều nhân viên thao tác song song.

# **8. Kết luận**

Đề tài **Coffee Shop Manager** đã đạt được các mục tiêu đặt ra:

* Mô phỏng thành công quá trình quản lý quán cà phê cơ bản.
* Áp dụng hiệu quả 3 mẫu thiết kế phần mềm: **Singleton, Strategy, Decorator.**
* Mã nguồn có cấu trúc rõ ràng, dễ mở rộng, dễ bảo trì, và sẵn sàng phát triển lên giao diện GUI 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 Java chính thức – Oracle Docs: <https://docs.oracle.com/javase/>
* Giáo trình “**Mẫu Thiết Kế Phần Mềm**” – Khoa CNTT, Đại học Đà Lạt.