OOP1 Project Report

Restaurant Order Management System

Student Name: Yash Bhausaheb Dhole
Student ID: A00335862
Module: OOP1 – Object-Oriented Programming
Github - https://github.com/YashBhausahebDhole/RestaurantOrderSystem

1. Introduction

A straightforward Java 21 console program, the Restaurant Order Management System is made to process customer orders, manage menu items, apply discounts, and provide formatted receipts.

The practical application of both basic and sophisticated Java features covered in the OOP1 module is demonstrated by this project.

The application computes totals with discounts, lets users choose menu items interactively, and prints a polished receipt that shows both the original and discounted costs together with the justifications for each discount.

2. Features and User Stories Completed

- The system displays a menu with different food and drink items.
- The user can select one or more items by entering numbers.
- The application calculates the total cost and applies eligible discounts automatically.
- A formatted receipt is generated showing item details, discount amounts, and total.
- Exceptions handle invalid input or missing menu items gracefully.

3. Language Features Used

Feature Category	Java Concept Used in Project	Class/Code Reference
Classes	Restaurant, MenuItem, Order, etc.	Entire model package
this() / this.	Used inside constructors	FoodItem, DrinkItem
Method Overloading	addMenuItem() methods	Restaurant.java
Varargs	addItems(MenuItem... items)	Order.java
LVTI (var)	Local variable declarations	Main.java
Encapsulation	Private fields with getters/setters	All model classes
Interfaces	Discountable with multiple method types	Discountable.java
Inheritance	FoodItem, DrinkItem extend MenuItem	Model package
Overriding / Polymorphism	applyDiscount()	FoodItem, DrinkItem
super()	Parent constructor calls	Subclasses
Exceptions	IllegalArgumentException, NoSuchElementException	Throughout
Enums	Category	Enum file
Arrays / Lists	Varargs arrays + ArrayList	Order.java
Core API	LocalDateTime, StringBuilder, Collections	Receipt.java
Call-by-value / Defensive Copying	Return copies of lists	Order.getItemsCopy()
Interface Methods (private/default/static)	All implemented	Discountable.java
Record Type	Customer	Record file
Immutable Custom Type	Receipt	Receipt.java
Lambda / Predicate	Filtering cheap items	Main.java
Final / Effectively Final	Local variables not reassigned	Order.java, Receipt.java
Method References	System.out::println	Restaurant.printMenu()
Switch Expressions	Category description	Restaurant.printMenu()
Sealed Classes	MenuItem permits FoodItem, DrinkItem	MenuItem.java

4. Evaluation

This project fully meets the assignment brief. All required fundamental and advanced OOP concepts are demonstrated. The code is modular, readable, and designed with extensibility in mind. Error handling and defensive copying ensure stability and correctness. During development, challenges included ensuring immutability and designing an interface that supports multiple discount types. These were overcome by using a sealed class structure and leveraging Java's default and static interface methods. Future improvements could include GUI integration (JavaFX) or persistence with a database, but for the current scope, the objectives are fully achieved.

- **Fundamentals:** method overloading, varargs, encapsulation, enums, inheritance, and exceptions.
- **Advanced concepts:** sealed classes, records, default/static interface methods, lambdas, method references, switch expressions, and an immutable Receipt.
- **Polymorphism:** different discount logic for food and drink via applyDiscount() overrides.
- **Encapsulation & defensive copying:** all data protected and immutable.
- **Interface flexibility:** Discountable interface contains default, static, and private methods.

5. Conclusion

The Restaurant Order Management System demonstrates a thorough understanding of Java OOP principles and modern Java 21 features in a clear, educational, and practical application. This report, alongside the code, UML diagram, and screencast, together form a complete submission per the OOP1 assignment requirements.