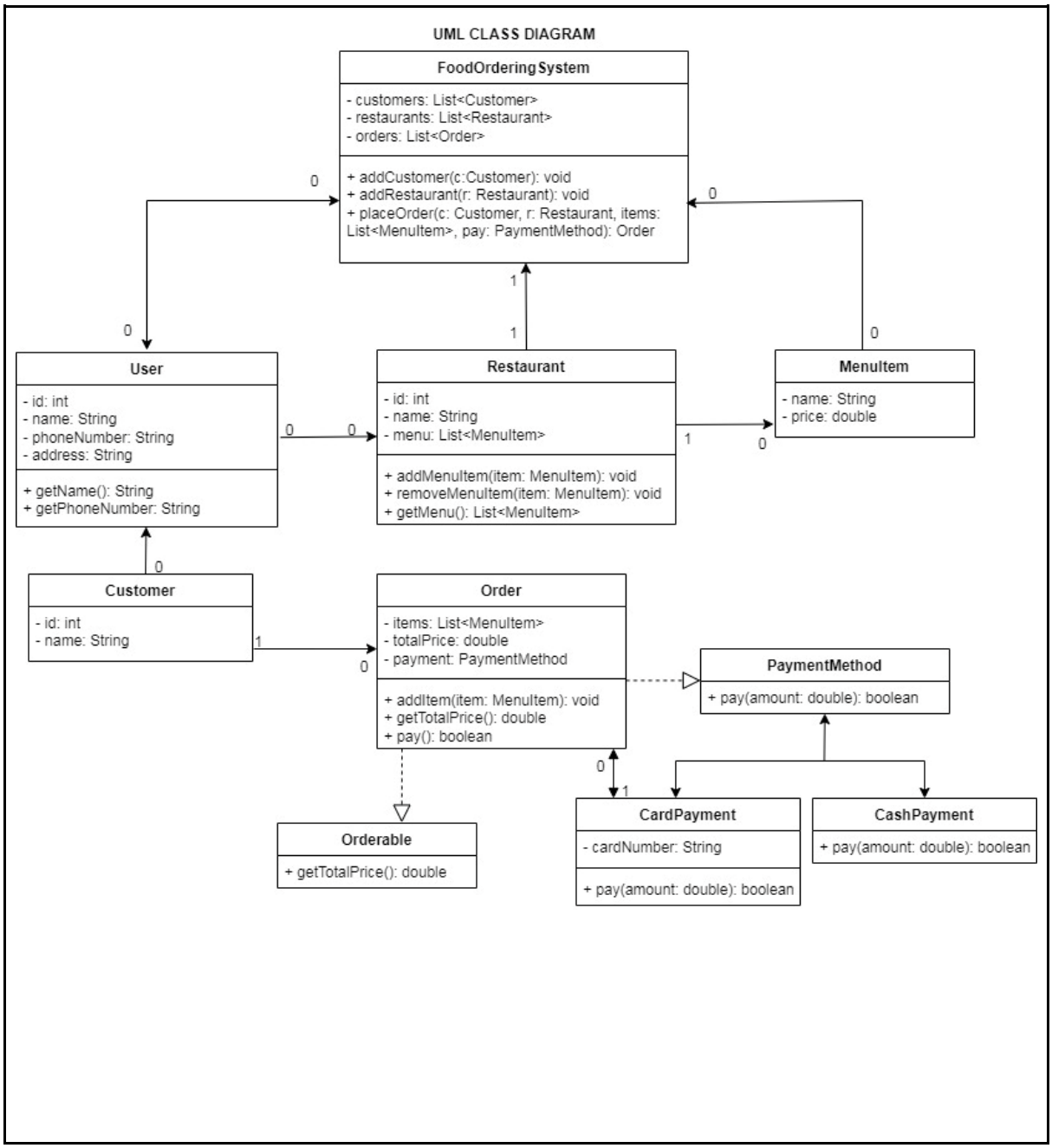
FINAL PROJECT REPORT
OBJECT ORIENTED PROGRAMMING - FALL 2025

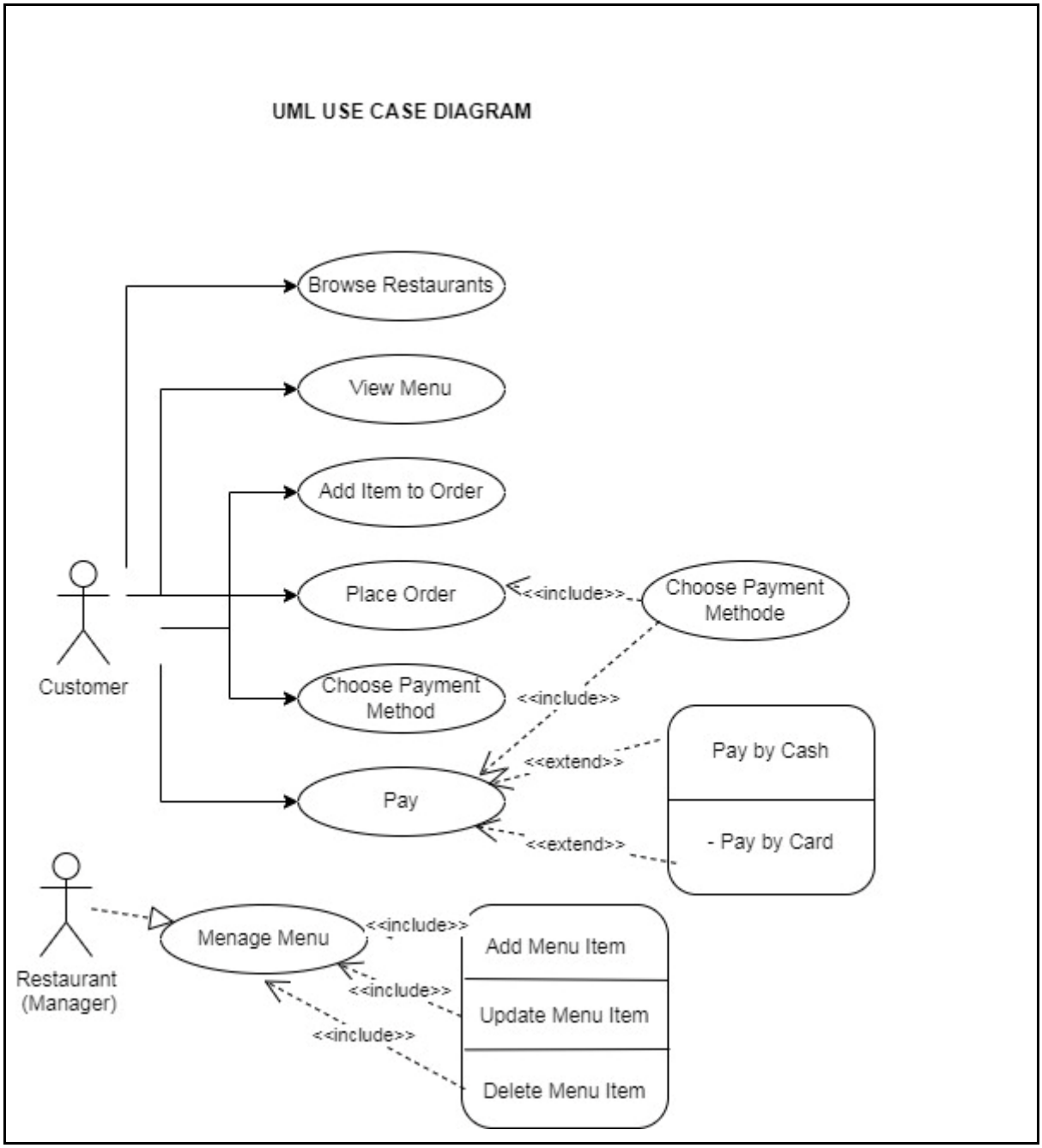
Name, Surname	Issa Konaté
Student ID	220303974
Name of the Project	Online Food Ordering System

PROJECT DESCRIPTION

The Online Food Ordering System is a console-based Java application designed to simulate the basic workflow of ordering food from restaurants. The system allows customers to browse restaurants, view menus, add menu items to an order, and complete payments using different payment methods. It aims to model real-world food ordering platforms in a simplified and object-oriented way.
The main users of the system are customers and restaurant managers. Customers can place orders and choose payment methods, while restaurant managers can manage menu items by adding, updating, or deleting them. The project focuses on applying Object Oriented Programming principles such as encapsulation, inheritance, polymorphism, and interfaces, while also using JUnit testing and Git/GitHub to simulate a real software development workflow.

UML DIAGRAMS




UML USE CASE DIAGRAM
Browse Restaurants
View Menu
Add Item to Order
Place Order
<<include>>
Choose Payment Methode
Customer
Choose Payment Method
<<include>>
<<extend>>
Pay by Cash
Pay
- Pay by Card
<<extend>>
Menage Menu
<<include>>
Add Menu Item
Restaurant (Manager)
<<include>>
Update Menu Item
<<include>>
Delete Menu Item

SYSTEM DESIGN

Number of classes	8
Number of interfaces	2
Number of abstract classes (if any)	0

Short explanation of your classes:

ClassName 1	FoodOrderingSystem Acts as the main controller of the application. It manages customers, restaurants, and orders, and coordinates the order placement process.
ClassName 2	User A base class that stores common user information such as name, phone number, and address.
ClassName 3	Customer Extends User. Represents a customer who can place food orders from restaurants.
ClassName 4	Restaurant Stores restaurant information and manages a list of menu items. Provides methods to add, remove, and retrieve menu items.
ClassName 5	MenuItem Represents a food item with a name and price.
ClassName 6	Order Represents a customer's order. Stores selected menu items, calculates the total price, and processes payment.

ClassName 7	PaymentMethod (Interface) Defines a common pay(amount) method for different payment strategies.
ClassName 8	CardPayment / CashPayment Concrete implementations of the PaymentMethod interface that handle different payment types.

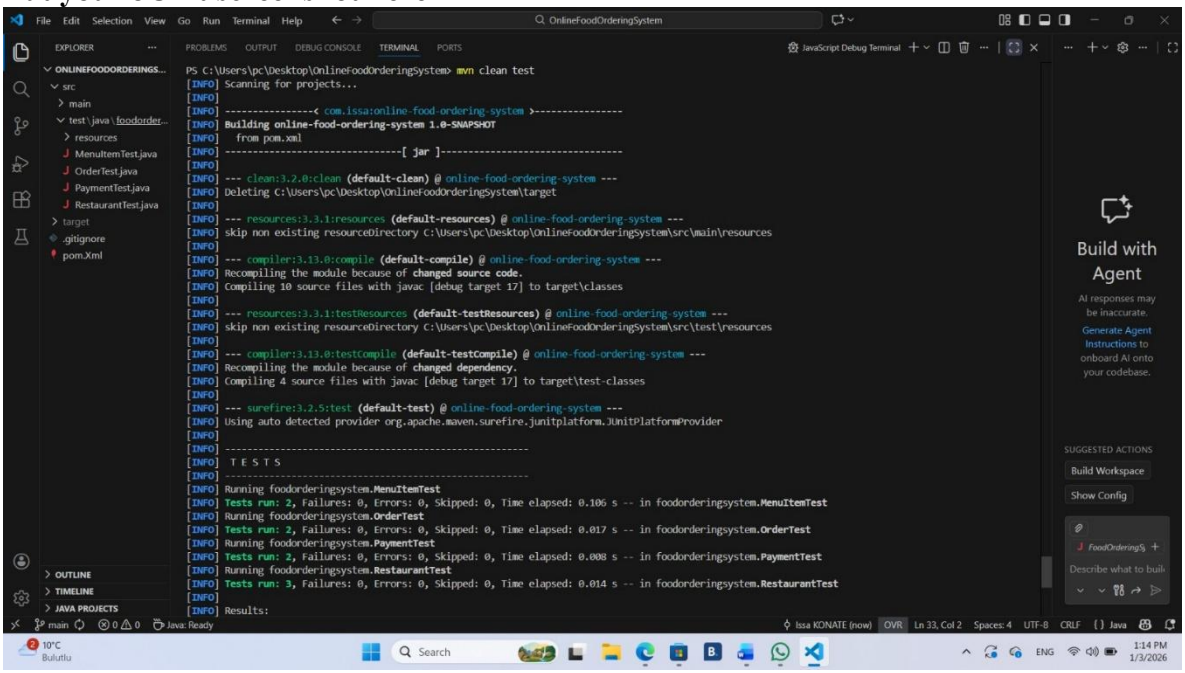
Explain how did you accomplish the following:

Encapsulation	Encapsulation is achieved by declaring class fields as private and accessing them through public getter and setter methods. This protects data from direct external access and ensures controlled modification.
Inheritance	Inheritance is implemented by having the Customer class extend the User class. This allows reuse of common attributes like name and phone number while adding customer-specific behavior.
Polymorphism	Polymorphism is used in the payment system. The PaymentMethod interface allows the program to process payments using different implementations (CashPayment or CardPayment) without changing the order logic.
Interfaces	Interfaces are used to define contracts. PaymentMethod enforces a common payment behavior, while Orderable defines pricing behavior, making the system flexible and extensible.

UNIT TESTING

How many classes you've tested?	4

Put your JUnit screenshot here



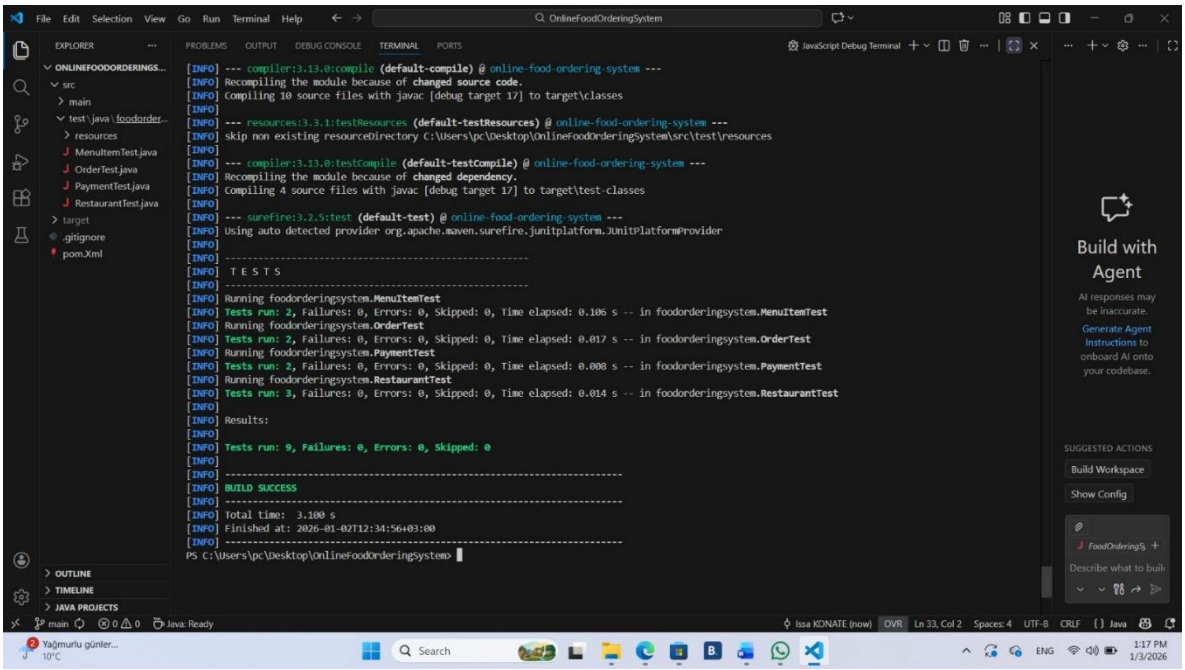


Figure X: Successful execution of JUnit tests using Maven (mvn clean test)

GIT & GITHUB

Github Repository Link:	https://github.com/tenakosai90/OnlineFoodOrderingSystem
Github Projects Link:	https://github.com/users/tenakosai90/projects/2/views/1

Add a screenshot of your **git log –oneline** here