Restaurant Menu Management System

An efficient solution through Java

KENT INSTITUTE AUSTRALIA

Developed By:

SUJAN DARJI

Student ID: K231673

Assignment:

Object Oriented Programming

Java™

Table of Contents

Project Overview

The Restaurant Menu Management System (RMMS) is an IT system used by a restaurant to manage menus and orders efficiently. It supports various functionalities such as:

- Managing different types of menus (e.g., Dine-In, Take-Away).
- Categorizing menu items (e.g., Breakfast, Lunch, Dinner).
- Handling customer orders with dynamic item selection and quantity specification.

- Applying customer-specific discounts and voucher-based discounts.
- Generating detailed invoices that include order summary and payment details.

How It Works

1. User Interaction:

- The user/restaurant customer is prompted to enter the customer's name and status.
- The system validates the customer status input (Active, VIP, or New).
- The user selects the menu type (Dine-In or Take-Away), with validation to ensure correct input.
- The user selects the menu category (Breakfast, Lunch, or Dinner), and the system displays the corresponding menu items.
- The user adds items to the order by entering item numbers and quantities. They can continue adding items until they type 'done'.
- The user can also add drinks (alcoholic or non-alcoholic) to the order, with age validation for alcoholic drinks.

2. Order Processing:

- The system calculates the total cost of the order, including customer-specific discounts.
- The user can apply a voucher code for additional discounts, with validation to ensure the code is valid.
- The user selects the payment method, with an additional surcharge for card payments.

3. Invoice Generation:

- The system generates a detailed invoice that includes customer details, a summary of the ordered items, applicable discounts, surcharges, and the total amount due.
- For Dine-In orders, a random table number is assigned and displayed on the invoice.

Components of the Project

1. MenuItem Class and Subclasses

1.1. MenuItem (abstract class):

- **Attributes:** *itemNumber*, *itemName*, *description*, *itemPrice*, *category*
- **Methods:** `calculatePrice()` (abstract), *getItemNumber()*, *getItemName()*, *getItemPrice()*, *getCategory()*, *toString()*

1.2. Subclasses:

- **`StandardMenuItem`**: Implements `calculatePrice()`.
- **`PremiumMenuItem`:** Implements `calculatePrice()` with a surcharge.
- **`DiscountMenuItem`:** Implements `calculatePrice()` with a discount.
- **`DrinkMenuItem`:** Implements `calculatePrice()` without any modification.

2. `Menu` Class

- **Attributes:** *id, name, purpose, venue, sessionTime, items* (ArrayList of `MenuItem`)
- **Methods:** *addMenuItem(), getMenuItem(), getItems(), toString()*

3. `Customer` Class

- **Attributes:** *name, status, discount*
- **Methods:** *Customer(), getName(), getStatus(), getDiscount(), setDiscount(), toString()*

4. `Order` Class

- **Attributes:** *customer, orderedItems* (ArrayList of `MenuItem`), *quantities* (ArrayList of Integer), *totalAmount, voucherDiscount, surcharge, isDineIn, tableNumber*
- **Methods:** *Order(), addMenuItem(), calculateTotal(), applyVoucherDiscount(),*

`applyPercentageVoucherDiscount()`, `applySurcharge()`, `generateInvoice()`, `getTotalAmount()`

5. `Voucher` Class

- **Attributes:** `code`, `discount`
- **Methods:** `Voucher()`, `getCode()`, `getDiscount()`

6. RestaurantDriver Class

- Contains the `main()` method that orchestrates user interaction, order processing, and invoice generation.

Key Methods and Concepts

1. Encapsulation:

- Each class encapsulates its own data and behavior, promoting modularity and reusability.
- Example: The `MenuItem` class encapsulates the properties and methods common to all menu items.

2. Inheritance:

- The `MenuItem` class serves as a base class for specific types of menu items, demonstrating inheritance.
- Example: `StandardMenuItem`, `PremiumMenuItem`, `DiscountMenuItem`, and `DrinkMenuItem` inherit from `MenuItem`.

3. Polymorphism:

- Polymorphism is achieved through the `calculatePrice()` method, which is overridden in each subclass of `MenuItem`.
- This allows different types of menu items to have their specific price calculation logic.

4. Abstraction:

- The `MenuItem` class is abstract, providing a template for its subclasses and ensuring that `MenuItem` cannot be instantiated directly.
- The `calculatePrice()` method is abstract, enforcing implementation in subclasses.

5. Validation:

- Input validation is implemented to ensure correct customer status, menu type, and voucher codes.
- Example: The system loops until valid inputs are provided for customer status and menu type.

Conclusion

The Restaurant Menu Management System (RMMS) showcases the effective application of OOP principles to create a maintainable and scalable solution for managing restaurant operations. The modular design ensures that each class has a clear responsibility, making the code easier to understand, maintain, and extend.