UTM
UNIVERSITI TEKNOLOGI MALAYSIA

Malaysia-Japan International Institute of Technology (MJIIT)

Object Oriented Programming
SEM 4-2023/2024

MINI PROJECT PROPOSAL
RESTAURANT ORDERING

NAME	**:**	**HARITH BIN BENNET APRIANTO**
STUDENT ID	**:**	**A22MJ5022**
NAME	**:**	**JANOT EMMANUEL JIA HENG CHRISTOPHE**
STUDENT ID	**:**	**A22MJ5037**
NAME	**:**	**SAFUAN HAKIM BIN SHANI**
STUDENT ID	**:**	**A22MJ8014**
NAME	**:**	**FATHIYA AZELYA PUTRI**
STUDENT ID	**:**	**A22MJ9160**
SECTION	**:**	**15**
LECTURER	**:**	**Dr Siti Nur Khadijah Aishah binti Ibrahim**
DATE	**:**	**15/6/2024**

TABLE OF CONTENTS

1 Introduction

Recently, it has been observed that more and more places in the food and beverage industry have adopted an online menu where a customer can look through and order directly through digital means instead of having a server take the customer's order.

This mini project aims to develop a menu ordering system using object-oriented programming principles for a restaurant. This system will be developed using Java. Digitalizing the ordering system is expected to enhance the dining experience for customers while streamlining the workflow for the restaurant staff.

2 Problem Statement

This mini project aims to solve the issues faced with traditional methods of placing orders in restaurants, which usually involves a waiter taking the customer's order manually and then going through the process of communicating it with the other staff, which can cause many issues such as:

Order inaccuracy

With traditional methods, it is not uncommon for human errors to occur as a waiter is taking a customer's order. This miscommunication can cause major setbacks for the kitchen staff and even cause lower customer satisfaction whenever they receive an incorrect order. All this may also cause food wastage.

Inefficiencies

The manual process is inherently slow as the customer must wait for a server, and the waiter has to take the order and relay it to the kitchen staff. It has been seen that there is a potential for the customer to wait a very long time before receiving their order or even to wait for their order to be taken as the restaurant can get very busy and the servers overwhelmed. With some of the problems with traditional methods listed, this system aims to provide the following functionalities to overcome said problems

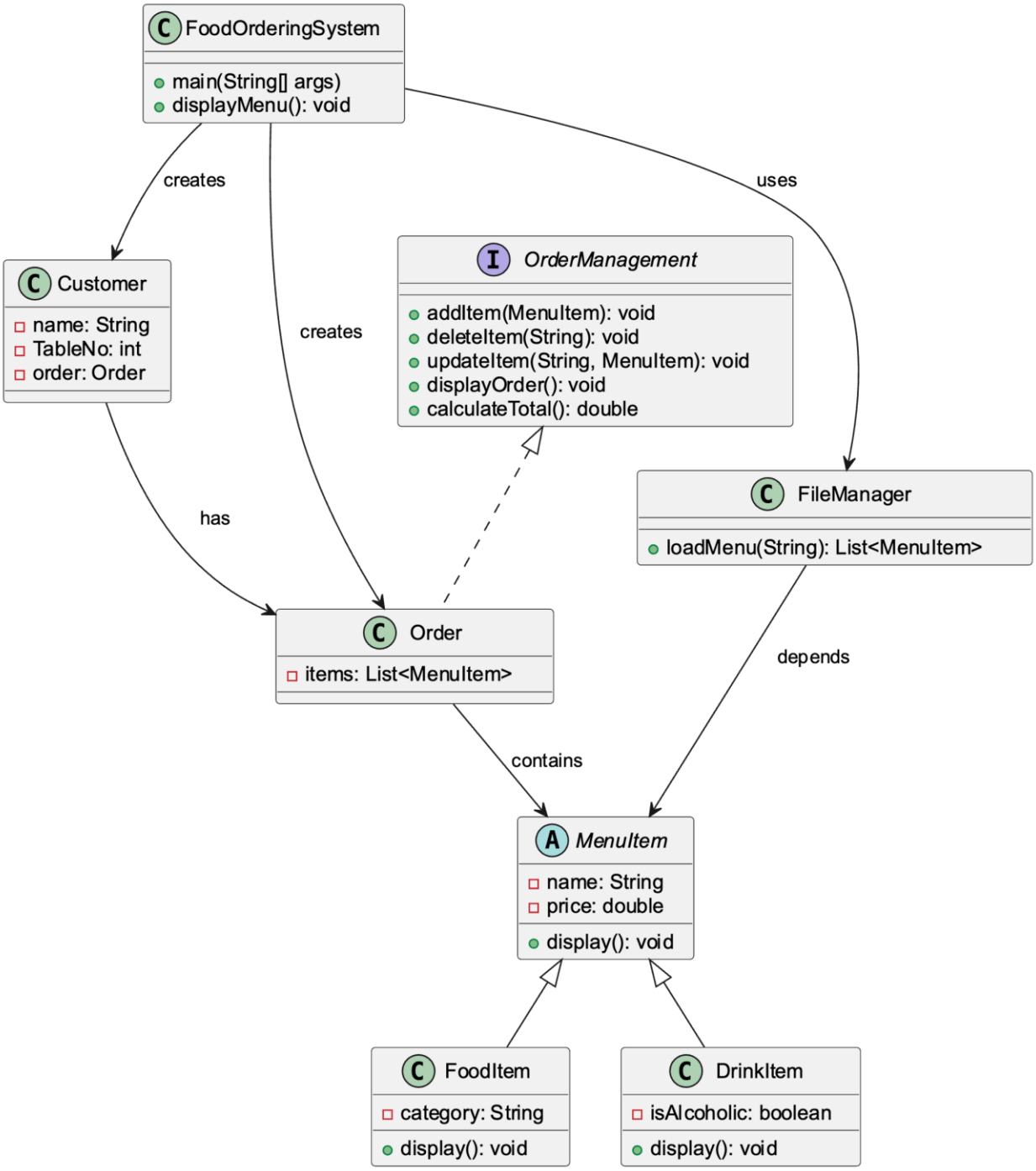
Digital Menu Interface

To ease the customer's decision to place an order, a digital menu interface showcasing all the food and drink items will be shown so the customer can pick what they want to order. Each item will have its name and description to help the customer make the right choice.

Order Confirmation

A confirmation menu will pop up after the customer presses the order button to ensure the customer is sure of what they want to order. This menu will display all items and allow customers to edit their order if needed. The price of their order will also be displayed on the same menu.

3 Class Diagram



3.1 Classes and Their Descriptions:

1. **FoodOrderingSystem**
    - **Description**: Represents the main class and entry point of the application. It initializes the system, manages user interactions, and displays the menu.
    - **Use**: Orchestrates the overall flow of the food ordering process, interacting with other classes to handle user inputs and display outputs.
2. **MenuItem**
    - **Description**: Abstract class defining common attributes (name, price) and methods (display ()) for menu items.
    - **Use**: Serves as a blueprint for concrete menu items (FoodItem and DrinkItem), promoting code reuse and maintaining a common interface for all menu items.
3. **FoodItem**
    - **Description**: Concrete subclass of MenuItem representing food items on the menu.
    - **Use**: Contains additional attributes specific to food (category). Implements display() to show details specific to food items.
4. **DrinkItem**
    - **Description**: Concrete subclass of MenuItem representing drink items on the menu.
    - **Use**: Contains additional attributes specific to drinks (isAlcoholic). Implements display() to show details specific to drink items.
5. **OrderManagement**
    - **Description**: Interface defining methods (addItem, deleteItem, updateItem, displayOrder, calculateTotal) for managing orders.
    - **Use**: Provides a contract for managing orders, which Order class implements. Enables consistent order operations across different types of orders.
6. **Order**
    - **Description**: Represents an order placed by a customer, containing a list of MenuItem instances.
    - **Use**: Implements OrderManagement, providing concrete implementations for order-related operations such as adding, deleting, and updating items, calculating total, and displaying the order details.

7. **Customer**
    - **Description**: Represents a customer placing an order, with attributes (name, TableNo) identifying the customer.
    - **Use**: Holds an instance of Order to manage the customer's current order. Facilitates operations related to customer-specific information and interactions with the order.
8. **FileManager**
    - **Description**: Manages file operations, specifically loading menu items from a file (menu.txt).
    - **Use**: Provides functionality to load menu items into a list of MenuItem instances. Supports data persistence and initialization of menu items from external storage.

3.2 Relationships:

- **FoodOrderingSystem**:
  - ⇒ Uses FileManager to load initial menu items and manages Customer interactions.
- **Customer**:
  - ⇒ Creates and manages an Order.
  - ⇒ Interacts with FoodOrderingSystem to place and manage orders.
- **Order**:
  - ⇒ Contains multiple MenuItem instances (FoodItem or DrinkItem).
  - ⇒ Implements OrderManagement to perform operations on items in the order.
- **MenuItem**:
  - ⇒ Serves as a superclass for FoodItem and DrinkItem, defining common attributes and behaviors for all menu items.
- **FoodItem** and **DrinkItem**:
  - ⇒ Inherit from MenuItem, providing specific attributes and behaviors for food and drink items.
- **FileManager**:
  - ⇒ Depends on MenuItem to manage and load menu items from menu.txt.

3.3 PIC for each class:

CLASS	PIC
FoodOrderingSystem	Fathiya
MenuItem	Fathiya
FoodItem	Harith
DrinkItem	Harith
OrderManagement	Janot
Order	Janot
Customer	Safuan
FileManager	Safuan

4 Techniques and tools

4.1 Java Techniques:

1. **File Handling**:
   ⇒ **Description**: Java provides classes and methods to handle file operations such as reading from and writing to files. In the food ordering system, FileManager class utilizes file handling techniques to load menu items from menu.txt.
2. **Polymorphism**:
   ⇒ **Description**: Polymorphism allows objects of different types to be treated as instances of a common superclass (Item in this case). It simplifies code by allowing methods to be written to manipulate objects of a superclass, and these methods work with instances of any subclass of the superclass.
3. **Interface Class**:
   ⇒ **Description**: Interfaces in Java define a contract for classes to implement. OrderManagement interface specifies methods (addItem, deleteItem, updateItem, displayOrder, calculateTotal) that classes implementing it must provide. This promotes code reusability and enables classes to be loosely coupled.
4. **Association**:
   ⇒ **Description**: Associations in object-oriented design represent relationships between classes. In the food ordering system, Customer has an association with Order, indicating that a customer can have one or more orders (Order is part of Customer).
5. **Inheritance**:
   ⇒ **Description**: Inheritance allows a class (FoodItem and DrinkItem) to inherit properties and methods from another class (Item). It promotes code reuse, establishes a hierarchical relationship between classes, and supports polymorphism.

6. **Abstract Class**:
    ⇒ **Description**: Abstract classes in Java cannot be instantiated directly but can contain abstract methods (methods without a body) and concrete methods. Item serves as a blueprint for FoodItem and DrinkItem, providing common attributes (name, price) and methods (display()).
7. **Exception Handling**:
    ⇒ **Description**: Exception handling in Java allows code to gracefully manage errors and unexpected situations. It involves try, catch, and finally blocks to handle exceptions (FileNotFoundException, IOException) that may occur during file operations, ensuring robustness and reliability in the application.
8. **Static Method**:
    ⇒ **Description**: Static methods in Java belong to the class rather than instances of the class. They can be called without creating an instance of the class (FoodOrderingSystem's main() method is a static method). They are used for operations that do not require an object state.
9. **ArrayList**:
    ⇒ **Description**: ArrayList in Java is a resizable array implementation of the List interface. Unlike traditional arrays, it dynamically resizes itself when elements are added or removed. In the food ordering system, the Order class uses an ArrayList<Item> (items) to manage menu items in an order, providing flexibility and efficiency.

4.2 Tools Used:

1. **GitHub**:
    - **Description**: GitHub is a web-based platform for version control using Git. It facilitates collaboration among team members, tracks changes to the codebase, manages issues, and hosts repositories, ensuring code integrity and project management.
2. **Discord**:
    - **Description**: Discord is a communication platform for text, voice, and video chat, primarily used for real-time communication among team members. It supports channels for different topics, direct messaging, and integration with other tools for enhanced collaboration.
3. **WhatsApp**:
    - **Description**: WhatsApp is a messaging application for smartphones and desktops. It is used for quick communication, sharing updates, and coordinating tasks among team members, offering features like group chats and multimedia sharing.
4. **Visual Studio Code (VS Code)**:
    - **Description**: Visual Studio Code is a lightweight but powerful source code editor developed by Microsoft. It supports syntax highlighting, intelligent code completion, debugging, version control integration, and extension support for various programming languages, including Java. It enhances productivity and facilitates code editing and debugging tasks.