Coffee Shop Management System

The Coffee Shop Management System is a Java-based console application developed to simulate the management of a coffee shop. The system provides an interface for two primary users: the admin (shop owner) and the customers. The admin can manage stock, menu items, and generate sales reports, while customers can view the menu, place orders, and make payments. Basic file-handling techniques are employed to ensure data persistence, meaning menu and order details are saved between sessions.

The system's functionalities include managing the menu, placing customer orders, updating stock levels, generating sales reports, and handling customer payments. The goal is to automate routine coffee shop operations to provide efficiency and ease in management.

The system's class structure includes:

- **`Main Class:`** Manages navigation between admin and customer functionalities.
- Person Class: Super class for both Admin and Customer, holding common attributes like name.
- **`Admin Class:`** Inherits from Person and handles admin tasks, such as managing the menu, analyze customer orders and information and analyze sales report.
- **`Customer Class:`** Inherits from Person and placing orders.
- Menu Class: Create a file variable for storing menu details. Responsible for menu-related operations such as displaying, adding, deleting, and updating items price and stocks and save the updates in file.
- **`MenuItem Class:`** Stores details about individual items such as price and stock.
- **`TakeOrder Class:`** Handles customer order processing such as create order, including stock updates and payment calculations.
- **`ShopDetails Class:`** Stores information about the shop like its name, address, and contact details.

Below is an overview of key functionalities:

1. Admin Features
- Login Authentication: Admin must enter a valid password to access the system.
- Menu Management:
    - Add, update, or delete items.
    - Update item prices and stock.
    - View the current menu.
- Sales Report Management:
    - View and review sales reports based on customer orders.
    - Clear sales reports.
    - View customer information.

- Order Stock Management:
    - Automatically update stock when a customer places an order.

2. Customer Features
- View Menu: Customers can view the menu and check available items.
- Place Order: Customers can select items from the menu and specify the quantity.
- Payment Options: Supports both card and cash payments, calculating change for cash payments.
- Order History: Orders are saved to a file for future reference and reporting.
- Customer information: Customers information are saved to a file for future reporting.

3. Data Persistence

File handling is used to save and retrieve data:

- `menu.txt` stores menu items (name, price, stock).
- `order.txt` records all customer orders. This ensures that data is retained across multiple sessions of the system, making it reliable for long-term use.

customer.txt stores customers information.

N.B. The UML Diagram is given in the next page.

Person

Person

-name:String

+Person(name:String)
+getName():String
+setName(name: String):void
+toString(): String

Admin

-adminPassword: String
-menu: Menu
-orderReport: File

+Admin()
+addItem()
+deleteItem()
+updateItemPrice()
+updateItemStock()
+viewSalesReport()
+viewCustomreInfo()
+clearSalesReport()
+getAdminPassword(): String

Customer

-phoneNumber : String
- customerList : list<String>

+Customer(name :String, phoneNumber :String)
+getPhoneNumber() :String
+createOrder(takeOrder: TakeOrder)
+saveCustomerInfo()
+addCustomerInfoToFile(customerInfo:
ArrayList<String>):void

1

Menu

-menuFile: File

+Menu()
+displayStock()
+addMenuItemToFile()
+deleteItemFromFile()
+updateMenuItemPriceInFile()
+updateMenuItemStockInFile()

ShopDetails

-shopName:String
-shopAddress: String
-phoneNumber: int
-open: boolean

+ getShopName(): String
+getShopAddress(): String
+getPhoneNumber(): int
+isOpen(): boolean
+toString(): String

TakeOrder

*

-orderFile: File
-menuFile:File
-shopDetails: ShopDetails

+takeOrder()
+createOder(itemName:String,quantity:int,
customerName:String):void
+paymentOption(bill:double): void
+updateMenu(menuItem:
MenuItem,itemName:String ,finalStock :int):void
+findMenuItem(itemName:String):MenuItem

MenuItem

-itemName: String
-itemPrice: double
-itemStock: int

+MenuItem(itemPrice: double,
itemStock :int)
+getItemName():String
+getItemPrice():double
+getItemStock():int
+toString(): String