CS-1004 Object Oriented Programming Fall - 2023

[Project] **[Total Marks 100]**

Instructions.

- **Plagiarism is strictly prohibited. If any instance of plagiarism is detected, a consequence of receiving a score of zero will be applied to all assessments. Additionally, the matter will be forwarded to the Disciplinary Committee. No exceptions or excuses will be entertained.**
- Maximum 3 students are allowed per group. No cross-section group is allowed. No group changes are allowed during project execution.
- Combine all your work in one folder. The folder must contain only the **CPP files and header files**.
- Rename the folder as ROLL-NUM_SECTION (e.g. 22i-1234_22i-1235_22i-1236_A) and compress the folder as a zip file. (e.g. 22i-1234_22i-1235_22i-1236_A.zip).
- Do not submit .rar file.
- Submit the .zip file on Google Classroom (Lab) within the deadline.
- Submission other than Google classroom (e.g. Email etc.) Will not be accepted.
- The student is solely responsible to check the final zip files for issues like corrupt file, virus in the file, mistakenly exe sent.
- If the instructor cannot download the file from Google classroom due to any reason it will lead to zero marks in the Project.
- Deadline of Project is **2nd December 2023 11:59 PM.**
- Deadline to **submit** Project is **3rd December 2023 11:59 PM. (a whole one day for submission only).**
- Projects submitted after the deadline will be marked DIRECT ZERO.
- Correct and timely submission of the Project is the responsibility of every student; hence no relaxation will be given to anyone.
- For the timely completion of the Project, start as early as possible.
- Comments: Comment on your code properly. Write your name and roll number (as a block comment) at the beginning of the solution to each problem.
- You must do proper allocation and deallocation of the memory where necessary.
- All programs must be generic.
- For the timely completion of the Project, start as early as possible.
- Your code should be modular.
- Note: Follow the given instructions to the letter, failing to do so will result in a zero.

Cafe Management System (CMS)

In this project, you will develop a comprehensive application designed to manage menu items, process orders, handling payments, tracking inventory, and offers distinct user roles for customers, cafe staff, and administrators. The system aims to streamline the process of ordering. Users, including students, faculty and non-faculty will benefit from the various features and functionalities provided by the application.

Inventory Control:
Cafe staff can manage the menu, update item details, and track stock levels. Users can easily browse and filter through available items, viewing details like ingredients, prices, and availability status.

Order Processing:
Customers can place orders, modify selections, and confirm purchases through an intuitive interface. Staff can oversee and manage incoming orders, update order statuses, and handle requests swiftly. Note: The menu and prices are different for students, Faculty and Non-Faculty. Students cannot order from Faculty menu, but faculty can order from any menu. Non-Faculty customers will have different menus and prices and they cannot order from the Students and Faculty menu.

Staff and User Roles:
Distinct user roles are established, offering functionalities such as menu management, order processing, and inventory control for staff. Customers have access to placing orders, viewing order history etc.

Feedback and Ratings:
Users can rate menu items, leave comments, and provide feedback, enriching the system with valuable insights for menu improvements.

Table Reservation:
Customers can reserve tables or order ahead, enhancing the dining experience. Staff manage and confirm reservations, ensuring efficient table allocation and customer satisfaction.

The system comprises three core user types:

- **Administrators (Café Manager)**
- **Staff (Café Staff)**
- **Customers (Students, Faculty, Non-Faculty)**

Each assigned specific privileges and tools tailored to their roles within the cafe environment.

Major entities along with their functionalities are described below:

Administrator:
Inherits from User
Methods: ViewMenu(), AddMenuItem(), RemoveMenuItem(), ManageInventory(), ManageDiscountPromotion(), ViewOrderHistory(), AddNotification(), RemoveNotification(), RateMenuItem(), DisplayFromFile(), ViewUsersFromFile(), ViewMenuItemsFromFile(), ViewOrdersFromFile(), ViewRatingsFromFile(), ViewPaymentsFromFile(), CalculateMonthlyEarnings(), Logout()

- ViewMenu(): Displays the cafe's available items for users to see.
- AddMenuItem(): Allows administrators to include new items on the cafe's menu.
- RemoveMenuItem(): Permits administrators to delete items from the cafe's menu.
- ManageInventory(): Offers administrators the ability to oversee and update the cafe's stock.
- ManageDiscountPromotion(): Allows the Administrator to create, update, or remove discounts or promotional offers on menu items, enabling dynamic pricing adjustments or promotional campaigns.
- ViewOrderHistory(): Shows a record of past orders made within the system.
- AddNotification(): Permits administrators to create and send messages to staff or customers.
- RemoveNotification(): Allows administrators to delete previously sent messages.
- RateMenuItem(): Enables users to provide ratings and feedback for items on the menu.
- DisplayFromFile(): Reads and presents specific stored data from a file for administrative viewing.
- ViewUsersFromFile(): Retrieves and displays user data from a file for administrative review.
- ViewMenuItemsFromFile(): Fetches and presents menu item details stored in a file for administrative viewing.
- ViewOrdersFromFile(): Retrieves and displays order information from a file for administrative inspection.
- ViewRatingsFromFile(): Fetches and shows ratings and feedback for menu items stored in a file.
- ViewPaymentsFromFile(): Retrieves and presents payment details or transaction records from a file for administrative reference.
- CalculateMonthlyEarnings(): Calculates and displays the total earnings or revenue generated within a month based on payment records stored in a file.
- Logout(): Ends the current session and exits the system for the user.

User:
Attributes: UserID, UserName, Password, UserType
Methods: Register(), Login()

- Register(): Lets new users create an account by providing essential details.
- Login(): Allows registered users to access the system by verifying their credentials.

Customer: (Students, Faculty, Non-Faculty)
Note: The menu and prices are different for students, Faculty and Non-Faculty. Students cannot order from Faculty menu, but faculty can order from any menu. Non-Faculty customers will have different menus and prices and they cannot order from the Students and Faculty menu.
Inherits from User
Methods: ViewMenu(), PlaceOrder(), ViewOrderHistory(), Logout()

- ViewMenu(): Shows the available items in the cafe's menu. Different according to type of user.
- PlaceOrder(): Lets users select items and confirm their order.
- ViewOrderHistory(): Displays a record of past orders made by the user.
- Logout(): Ends the user's current session and exits the system.

CafeStaff:
Inherits from User
Have access to all the menus. Can edit and view all menus
Methods: ViewMenu(), ProcessOrder(), AddMenuItem(), RemoveMenuItem(), Logout()

- ViewMenu(): Displays the cafe's available items for users to see.
- ProcessOrder(): Enables staff to handle and fulfill incoming orders.
- AddMenuItem(): Lets staff include new items to the cafe's menu.
- RemoveMenuItem(): Allows staff to delete items from the cafe's menu.
- Logout(): Ends the current session and exits the system for the user.

MenuItem:
Attributes: ItemID, ItemName, ItemDescription, Price, QuantityInStock
Methods: UpdateStock()
- UpdateStock(): Allows modification of the quantity of an item in the cafe's inventory.

Order:
Attributes: OrderID, CustomerID, ItemsOrdered[], TotalPrice, OrderStatus
Methods: AddItem(), RemoveItem(), CalculateTotal(), ConfirmOrder(), CancelOrder()

- AddItem(): Adds selected items to the order.
- RemoveItem(): Deletes specific items from the order.
- CalculateTotal(): Computes the total price of items in the order.
- ConfirmOrder(): Finalizes and confirms the placed order. And should call process payment.
- CancelOrder(): Aborts and cancels the ongoing order process.

Rating:
Attributes: RatingID, MenuItemID, CustomerID, Score, Comment
Methods: Rate()

Payment:
Attributes: PaymentID, OrderID, Amount, PaymentStatus
- Methods: ProcessPayment()

File Manager:
Attributes: FileName, User, MenuItem, Order, Rating, Payment

Methods: SaveUserToFile(), SaveMenuItemToFile(), SaveOrderToFile(), SaveRatingToFile(), SavePaymentToFile()

- SaveUserToFile(): Saves user data to a file for storage.
- SaveMenuItemToFile(): Stores menu item details in a file.
- SaveOrderToFile(): Records order information into a file for future reference.
- SaveRatingToFile(): Saves ratings and feedback for menu items to a file.
- SavePaymentToFile(): Stores payment details or transaction records in a file for reference or tracking.

Functionality and Menu (it's just an overview)

Menus are tailored for each user type, offering specific functionalities:

Customer Menu

- View Menu
- Place Order
- View Order History
- Rate Item
- Logout (When logout it should store the data in file automatically)

Cafe Staff Menu

- View Menu
- Process Order
- Add/Remove Menu Item
- Logout (When logout it should store the data in file automatically)

Administrator Menu

- View Menu
- Add/Remove Menu Item
- Manage Inventory
- Manage Discount/Promotion
- View Order History
- Add/Remove Notification
- Rate Menu Item
- Display from File
- View Users from File
- View Menu Items from File
- View Orders from File
- View Ratings from File
- View Payments From File
- Calculate Monthly Earnings
- Logout (When logout it should store the data in file automatically)

- You are tasked to identify and create separate classes for the entities and implement it. Add appropriate functionalities, as mentioned in the roles of the user.

- You are required to identify appropriate relationships between different classes and apply polymorphic behavior between them.

- A proper menu driven is required from your end. Add as much detail and functionality to the project on your own as needed.

- Your implementation should demonstrate file handling, operator overloading, polymorphism, and virtual functions, as well as all other concepts taught in the course.

- Output should be clear and easy to follow. Write your driver program such that you can easily demonstrate all the features.

- Bonus marks will be awarded if you implement any features extraordinarily well and thoroughly, and for original thought and design.