

CS 202: IT Workshop I Lab Exam, 2025

Total Duration: 2 hours

Total Marks =65

Instructions:

- 1) Read the question carefully before start coding.
- 2) Classes are defined with instance members and methods. These are primary requirements during implementation of the classes. However, you are free to add additional member and methods as per your convenience.
- 3) Create your own test cases for evaluation. Creating nice test cases for better demonstration of your code is also a part of evaluation.
- 4) There are different components for marking. You will be offered marks based on the correctness of your code as well as your appropriate explanation.
- 5) **Don't copy from others or use any AI tool like chatgpt. We will check for plagiarism of your code. If your code is found to be similar with others, then immediately both will be offered zero in the lab examination.**
- 6) **Your internet should be off during the entire examination. Otherwise, your exam will be cancelled immediately.**
- 7) **During examination, you should open either of the two following screens in your system: the pdf containing question or your editor. In case, if you open any to other tab or screen, your exam may be cancelled.**

Question:

You are required to develop a multithreaded, menu-driven Online Food Order System using Java. The system must allow customers to place food orders and also allow the system to display all placed orders. Multiple customers may attempt to place orders at the same time or may attempt to display at the same time. Only one thread is allowed in such cases. Other threads must wait based on specific conditions using wait() and notify() /notifyAll() methods.

[Marks = 10 +20+ 10 + 20 + 5 =65]

The following classes need to be implemented.

A) Food Item Hierarchy

- 1) abstract class FoodItem with:
 - protected String name;
 - protected double price;

Methods:

- Parameterized constructor
 - public abstract void display(): should display the name and price of the item
- 2) Create three main categories (sub-classes) extending FoodItem: pizza, Burger, Biryani
 - 3) Each category must have **two sub-types** i.e.,
 - pizza -> VegPizza, ChickenPizza
 - Burger -> VegBurger, CheeseBurger
 - Biryani -> VegBiryani, ChickenBiryani

Note that each subclass must initialize its own price and Override display() to print item details.

B) Order Class

- private String customerName;
- private FoodItem item;
- private int quantity;
- private double finalAmount; (It is equal to price of item*Quantity ordered)

Methods:

- Constructor to initialize order details and calculate final price (finalAmount)
- public void applyDiscount() → apply **10% discount** if quantity > 3. Note that discount is automatically applied while adding orders by a customer. Print a message showing the original amount and the final amount after discount for the order.
- public void printOrderDetails() -> Prints the order details (Name of Item and the final amount for an order)

Note that for a particular customer, his orders for different FoodItems are treated separately (for separate Order objects). However, if the customer orders same FoodItem with quantity more than once, don't create separate objects. Rather update the corresponding object by setting the appropriate quantity ordered.

C) OrderManager – Singleton Class & Critical Section: Create a Singleton class OrderManager that stores and manages orders.

- private static OrderManager instance;
- private ArrayList<Order> orderList (Maintains list of orders)
- private static int totalOrders (maintains count of total orders)
- private boolean isDisplayInProgress (It shows that status of ongoing display)

Methods and Constructors:

- private constructor (Initializes orderList and isDisplayInProgress)
- **public static synchronized OrderManager getInstance()** -> Returns the singleton instance of the OrderManager class.

Note that the following two methods will be invoked by two separate threads (customer thread and display thread), which are explained in the next page.

- **public synchronized void addOrder(Order o)** → adds an order. If another thread is currently displaying orders (by calling displayOrders() explained next), the current thread must wait until display is finished. Once display finishes, it adds the order, updates necessary totals, and then notifies the other threads to wake them up.
- **public synchronized void displayOrders():** This method is responsible for displaying all the orders stored in the system. If the list is currently empty, the thread must wait until at least one order is added. Once the display begins, it sets the *isDisplayInProgress* flag to true, shows all orders, then resets the flag back to false and finally notifies the blocked threads so that they can continue.
- Also, include methods for search by customer name, show highest priced order, calculate average price and show customers with discounts with discounted prices.

D) Multithreading:

- Create CustomerThread extending Thread that Attempts to place order through addOrder() during addition of an order (Options 1,2,3 of the menu).
- Create DisplayThread extending Thread that Calls displayOrders() during option 4 of the menu.

E) The driver code must provide menu driven options which are as follows.

- 1) Order Pizza,
- 2) Order Burger,
- 3) Order Biryani
- 4) Show all orders (DisplayThread)
- 5) Show total order count
- 6) Search order by customer name. Print all details related to the order if found.
- 7) Discounted customers (show customers who got discounts and the original and final discounted amount)
- 8) Show highest priced order.
- 9) View average order price.
- 10) Exit