CS 233

Project Assignment: Online Food Delivery System (Java)

Objective:

This project aims to design and implement a simplified online food delivery system using Java. It will serve as a practical exercise to demonstrate your understanding of object-oriented programming (OOP) principles, data structures, and file I/O.

System Requirements

The online food delivery system should encompass the following core functionalities:

1. **Ordering**
    - Customers can place orders for hamburgers, fries, and drinks.
    - Each menu item should have a name and price.
    - Orders should store details about the customer, selected items, and order status (e.g., placed, accepted, in-progress, delivered).
2. **Delivery**
    - Drivers are *assigned* orders by the system based on the logic in `Driver Assignment`.
    - Drivers can then mark their assigned orders as 'in-progress' and 'delivered'.
    - The system keeps track of basic driver information (e.g., name, availability status).
3. **Driver Assignment**
    - The system must maintain a pool of *available* drivers.
    - When a new order is placed and 'accepted' by the system, it must be assigned to the available driver with the **highest average rating**.
    - You must implement an **efficient mechanism** to manage the pool of available drivers that allows the system to quickly retrieve the one with the highest rating.
    - When a driver completes a delivery, they should be marked as 'available' and returned to the pool of available drivers.
4. **Rating**
    - Customers can rate drivers on a scale of 1 to 5 after an order is delivered.
    - Each driver can have a maximum of 10 ratings stored at a time. When a new (11th) rating comes in, it must replace the oldest one.
    - A driver's *average rating* should be re-calculated every time a new rating is added.
5. **Order Processing**
    - Orders must be processed in the order they are received (First-In, First-Out). This applies to the *initial acceptance* of orders before they are assigned to drivers.

6. **Data Persistence**
    - The system must load its menu (items and prices) from a configuration file (e.g., `menu.txt`) at startup.
    - The system must also read and write all user data (e.g., `customers.txt`, `drivers.txt`, `admins.txt`) to files. This includes login credentials and driver ratings.
    - **All new orders** must be appended to a persistent file (e.g., `orders.txt`) as they are created. This file will serve as a log of all transactions.
    - This ensures all key data persists between application runs.
7. **User Authentication & Dashboards**
    - The system must support three distinct user roles: **Admin**, **Customer**, and **Driver**.
    - A login mechanism is required.
    - After logging in, each user type should see a simple "dashboard" (a menu of options) specific to their role:
        - **Admin:** Can manage the menu (add/remove/update items) and (optionally) view all orders.
        - **Customer:** Can place a new order, view their order history, and rate a completed delivery.
        - **Driver:** Can view their assigned order(s) and update an order's status (e.g., to 'in-progress' or 'delivered').

Tasks

To successfully complete this project, you will need to undertake the following tasks:

1. **Design**
    - Create comprehensive class diagrams to model the system's structure. This must include classes for the different user types, menu items, orders, etc.
    - Clearly define the relationships between these classes (e.g., inheritance).
    - Ensure your design adheres to OOP principles.
2. **OOP Principles**
    - Provide a detailed explanation of how your design incorporates the four fundamental OOP principles:
        - **Encapsulation**
        - **Abstraction**
        - **Inheritance** (Hint: Consider the similarities and differences between your user types).
        - **Polymorphism**
3. **Data Structures and Algorithms**
    - Describe the data structures you will use to implement the system's functionalities.

    - You must **justify your choices** based on the specific requirements and efficiency. Pay special attention to the data structures you select for:
        - **Order Processing:** How will you ensure orders are processed First-In, First-Out?
        - **Driver Assignment:** What data structure will you use to efficiently manage available drivers and select the one with the highest rating?
        - **Driver Ratings:** What data structure will you use to store only the 10 most recent ratings, ensuring the oldest is replaced by the newest?
    - Describe the algorithm you will use to read from and write to text files for all persistent data (users, menu, and orders).
4. **Implementation**
    - Write the Java code to implement the system based on your design.
    - The system may be fully operable through a basic **command-line interface (CLI)**.
    - Include necessary methods for:
        - **User login** and role-based dashboards.
        - **Placing orders** (and saving them to the `orders.txt` file).
        - **Assigning orders** to drivers based on your chosen algorithm.
        - Calculating order totals.
        - Managing and calculating driver average ratings.
        - **File I/O:** Reading/writing all persistent data (users, menu, and orders).
    - You may utilize GitHub for collaborative development and version control.
    - **(Optional Extension):** If all core requirements are met and time permits, feel free to develop a graphical user interface (GUI) using JavaFX or Swing. This is not required but will be considered for bonus credit.
5. **Demonstration**
    - You will be required to demonstrate your working system (via the CLI or GUI(if you built one)).

Submission and Deadlines

- **Tuesday, October 28, 2025 (100 points):**
    - Submit a PDF document named "CS233_GROUP_NAME.pdf".
    - This document should include:
        - The first version of the system design (Class diagrams).
        - Explanations of how you applied OOP principles.
        - Descriptions and justifications for your chosen data structures and file I/O strategy.
    - One person from each group should submit the document.
- **Wednesday, October 29, 2025 (80 points):**

    - Submit a revised version of the system design and implementation based on the feedback received.
- **Wednesday, November 19, 2025 (150 points):**
    - Demonstrate the current system code with some core classes implemented (e.g., login, file I/O, placing an order).
- **Wednesday, December 5, 2025 (200 points):**
    - Submit the final code and demonstrate the complete working system (via CLI or optional GUI).

Grading Rubric (for first submission)

- **Design (60%):** The system is well-designed using OOP principles. Class diagrams are clear and accurately represent the complete system structure (including users, orders, etc.).
- **Data Structures & Algorithms (30%):** Appropriate data structures are chosen, justified, and effectively utilized for all major requirements (e.g., order processing, driver assignment, and rating storage). File I/O logic is clearly described.
- **Document Clarity (10%):** The submitted document is clear, concise, and easy to read.