Project Assignment: Developing a Text-Based UNO Game

(Important!! Due Date 05/01/2025)

Objective:

Through this project, students will demonstrate their ability to apply core OOP concepts such as encapsulation, inheritance, abstraction, and polymorphism while developing a functional, interactive program in Java.

Project Description:

In this assignment, students will develop a console-based version of the UNO game, where players can interact with the system through text commands.

The system will simulate the game mechanics, including card drawing, turn management, and rule enforcement. Players will be able to play cards by matching colors or numbers, and the game will feature special cards such as *Skip*, *Reverse*, and *Draw Two*. This project will be an opportunity to practice designing modular systems, understanding game logic, and implementing clean, maintainable code using OOP principles.

Assignment Requirements

1. Functional Requirements

1. **Game Initialization:**
    - Design a system to initialize the deck of cards, shuffle it, and distribute cards to players.
    - Allow the program to accommodate 2–4 players, with the option of human or automated opponents.
2. **Gameplay Mechanics:**
    - Implement turn-based gameplay, ensuring that players can:
        - Play a card that matches the current top card by color or number.
        - Draw a card if they cannot play any cards.
    - Enforce rules for special cards like *Skip*, *Reverse*, and *Draw Two*.
    - Optionally, include wild cards (*Wild*, *Wild Draw Four*).
3. **Game State Management:**
    - Display the current game state, including:
        - The top card on the discard pile.
        - The player's hand (for the human player).
        - Which player’s turn it is.
    - Detect when a player has won the game (i.e., they have no cards left).

4. **Interactive Console Interface:**
    - Develop a clear, user-friendly text interface for player interactions.
5. **Game Rules Enforcement:**
    - Validate all player actions to ensure they adhere to the rules of UNO.

2. Object-Oriented Design Requirements

Students are expected to design the system using Object-Oriented Programming principles. This includes:

- **Encapsulation:** Ensuring that data and methods are logically grouped within classes, with controlled access.
- **Inheritance:** Utilizing a hierarchy where appropriate to minimize redundancy and promote code reuse.
- **Abstraction:** Abstracting common behaviors and properties to simplify code structure.
- **Polymorphism:** Designing flexible systems where behavior can vary depending on context (e.g., different player types or card actions).

Suggested Class Design

While students are encouraged to develop their own design, they may consider the following:

1. **Card Class:** Represents individual cards, storing attributes such as `color`, `type`, and `value`.
2. **Deck Class:** Manages the collection of cards, including shuffling, drawing, and resetting.
3. **Player Class:** Represents players, storing their `hand` and managing their actions.
4. **Game Class:** Handles the core game logic, including turn management, rule enforcement, and the game state.

Deliverables

1. **Code Implementation:**
    - A Java program that implements the described functionality and adheres to OOP principles.
2. **Design Documentation:**
    - A brief report explaining the class structure and how key design decisions align with OOP principles.
    - A UML class diagram representing the system architecture. (optional)
3. **Sample Game Run:**
    - Include a documented/video walkthrough of a sample game, showcasing the program in action (video is preffered).
4. **Reflection:**
    - A short reflection on challenges encountered during development and how they were overcome.

Evaluation Criteria

1. **Correctness (40%):**
   - Does the program follow UNO rules and handle game scenarios correctly?
2. **OOP Implementation (30%):**
   - Are the principles of encapsulation, inheritance, abstraction, and polymorphism applied correctly?
3. **Game Flow (20%):**
   - Is the game interactive, user-friendly, and free of major bugs?
4. **Bonus (10%):**
   - Additional features such as:
     - Wild cards implementation.
     - Persistent storage to save game progress.

Group Work Instructions

Students are required to form **groups of four (4)** to collaborate on this project. Each group will work collectively on the design, implementation, and testing of their UNO game. Collaboration should include:

- **Task Distribution:** Clearly assign roles and responsibilities to ensure equitable contribution.
- **Regular Communication:** Conduct regular group discussions to track progress and address challenges.
- **Team Reflection:** Include a short section in the final report detailing the group dynamic, task allocation, and individual contributions.