Design Document Template for War Card Game

1. Project Background and Description

This project implements the classic card game "War" using a Java-based Object-Oriented design. It involves two players who draw cards from their decks and compare them, with the higher-ranked card winning the round.

2. Design Considerations

This section explains the key OO principles applied in the game design:

- **Encapsulation**: Each class encapsulates specific attributes and behaviors relevant to its function. For example, Card encapsulates the properties of a card (suit and rank) and methods to retrieve them.
- **Cohesion**: The classes are designed with high cohesion. Each class has a clear, single responsibility—Deck manages a collection of cards, Player handles the player's hand, and WarGame manages the overall game flow.
- **Coupling**: The code maintains low coupling by using clear interfaces between classes. WarGame interacts with Player and Deck objects without knowing their internal details.
- **Inheritance**: In the base implementation, no inheritance is used, but the code can be extended using inheritance if necessary (e.g., creating specialized Card classes for different game types).
- **Aggregation and Composition**: The relationships between classes use aggregation (e.g., WarGame has Player objects) and composition (e.g., Deck contains Card objects).
- **Flexibility/Maintainability**: The modular design allows easy extension or modification of rules without impacting the entire codebase.

3. Updated Class Diagram

Include an updated class diagram showing:

- Card as the basic entity.
- Deck containing Card objects.
- Player managing a hand of Card objects.
- WarGame managing gameplay with Player and Deck interactions.

4. Implementation Details

- **Classes Overview**: Provide a summary of each class and its role in the game.
- **Data Structures**:

    - **List<Card>**: Used in Deck to manage the collection of cards.
    - **Queue<Card>**: Used in Player for managing the player's hand, allowing efficient draw and play operations.
- **Key Algorithms**: Explain the critical game logic:
    - **Card Comparison**: Compares the rank of two cards to determine the winner.
    - **War Resolution**: A "war" scenario where additional cards are drawn and compared to resolve ties.

6. Testing Strategy

- Unit tests are provided in WarGameTest.java.
- Testing covers:
    - Card drawing and shuffling.
    - Game round mechanics.
    - War scenarios.
    - Edge cases (e.g., game ending conditions).

7. Future Enhancements

Outline potential improvements, such as adding a GUI, creating multiplayer support, or extending game rules for variations of War.