War Game - Full Documentation & User Guide

1. Introduction

The War Game is a simple two-player card game where players compete to win all the cards. This document provides a detailed explanation of how the game works, how to install and play it, and an in-depth breakdown of the Java code that implements it.

2. Game Overview

The game is based purely on chance, where players draw cards and compare their ranks. The winner of each round takes both cards, and the game continues until one player has all the cards.

3. Rules of the Game

1. The game is played with a standard 52-card deck.
2. The deck is shuffled and split evenly between two players.
3. Each round, both players reveal their top card.
4. The player with the higher-ranked card wins both cards.
5. If both players have the same rank, a 'War' occurs, where additional cards are drawn.
6. The game continues until one player wins all 52 cards.

4. How to Install and Run the Game

1. Install Java Development Kit (JDK 17+).
2. Download and open the project in NetBeans or Eclipse.
3. Locate and run `WarMain.java` to start the game.

5. Detailed Gameplay Mechanics

Each round, players compare their top cards. The player with the higher card wins both. In case of a tie, a War occurs where multiple cards are drawn to determine a winner.

6. Code Structure and Implementation

The game is implemented in Java using an Object-Oriented approach:

- `WarMain.java`: Starts the game.
- `WarGame.java`: Controls the game logic.
- `WarPlayer.java`: Represents a player.
- `WarDeck.java`: Manages the deck.
- `WarCard.java`: Represents individual cards.

7. UML Class Diagram

Overview

The UML Class Diagram represents the structure of the War Game project. It includes classes and relationships that define the game's functionality.

Class Descriptions

1. **Card (Abstract Class)**
    - +toString(): String (Abstract Method)
2. **WarCard (Inherits from Card)**
    - -suit: String
    - -rank: int
    - +WarCard(suit: String, rank: int)
    - +getRank(): int
    - +toString(): String
3. **GroupOfCards**
    - -cards: ArrayList
    - -size: int
    - +shuffle(): void
    - +getCards(): ArrayList
4. **WarDeck (Extends GroupOfCards)**
    - +addCard(card: WarCard): void
    - +drawCard(): WarCard
5. **Player (Abstract Class)**
    - -name: String
    - +getName(): String
    - +play(): void (Abstract)
6. **WarPlayer (Extends Player)**
    - -deck: WarDeck
    - +playCard(): WarCard

- +hasCards(): boolean
- +getDeck(): WarDeck

7. **Game (Abstract Class)**
    - -name: String
    - -players: ArrayList
    - +play(): void (Abstract)
    - +declareWinner(): void (Abstract)
8. **WarGame (Extends Game)**
    - -player1: WarPlayer
    - -player2: WarPlayer
    - +WarGame(name: String, p1: WarPlayer, p2: WarPlayer)
    - +play(): void
    - +declareWinner(): void
9. **WarMain**
    - +main(args: String[]): void

Relationships in UML Diagram

- **Generalization (Inheritance)**: WarCard → Card, WarDeck → GroupOfCards, WarPlayer → Player, WarGame → Game.
- **Composition**: WarPlayer has WarDeck, WarGame has WarPlayer (player1 & player2).
- **Aggregation**: Game has multiple Player objects, GroupOfCards has an ArrayList.

8. Git Repository Usage

1. Clone the repository: `git clone <repository_url>`
2. Create a new branch: `git checkout -b new-branch`
3. Commit changes: `git commit -m 'Message'`
4. Push updates: `git push origin new-branch`

9. Design Considerations

The game follows Object-Oriented Design principles:

- **Encapsulation**: Data is kept private and accessed through methods.
- **Delegation**: Different classes handle different aspects of the game.
- **Flexibility**: The code is structured for easy modifications.

10. Conclusion

The War Game is a simple, yet engaging game implemented in Java. This guide provides a step-by-step breakdown of its setup, gameplay, and internal code structure.