Project Documentation : Card Deck Simulation

Table of Contents :

1. Introduction

➢ Purpose

The purpose of this project is to simulate a deck of playing cards in Java. It includes functionalities like deck initialization, shuffling, drawing cards, and sorting.

➢ Scope

This project focuses on creating a flexible and extensible card deck simulation. It provides a Deck class with methods for shuffling, drawing cards, and sorting based on custom criteria.

➢ Dependencies

- Java Standard Library (No external dependencies)

2. Code Overview

➢ Classes and Enums

- **Card**: Represents a playing card with a suit and rank.
- **Suit**: Enum for card suits (SPADE, CLUB, HEART, DIAMOND).
- **Rank**: Enum for card ranks (ACE, TWO, ..., KING).
- **CardComparator**: Custom comparator for sorting cards based on color, suit, and value.
- **Deck**: Represents a deck of cards with methods for initialization, shuffling, drawing cards, and sorting.

➢ Deck Initialization

The **Deck** class initializes a deck with 52 cards by combining all suits and ranks.

➢ Shuffling

The **shuffle()** method uses the Fisher-Yates shuffle algorithm to randomize the order of cards in the deck.

➢ Drawing Cards

- **drawCard()**: Draws a single card from the top of the deck.
- **drawRandomCards(int count)**: Draws a specified number of random cards from the deck.

➢ Sorting

The **drawnCards** list is sorted using the **CardComparator** before displaying.

3. Unit Tests

➢ Test 1: Deck Initialization

- Verify that the deck is initialized with 52 cards.

➢ Test 2: Shuffling

- Confirm that shuffling alters the order of cards in the deck.

➢ Test 3: Drawing Cards

- Ensure that drawing a card reduces the deck size by 1.
- Confirm that drawing multiple cards returns the correct number of cards.

➢ Test 4: Sorting

- Verify that sorting the drawn cards results in a sorted order based on the custom comparator.

➢ Test 5: Deck Size

- Check that the deck size is accurate after drawing cards.

4. Conclusion

➢ Summary

The card deck simulation project successfully provides functionalities for deck initialization, shuffling, drawing cards, and sorting. Unit tests ensure the correctness of these functionalities.

➢ Future Improvements

- Implement more advanced sorting strategies.
- Add additional functionalities like dealing hands for card games.
- Enhance error handling and input validation.

Note:

The unit tests can be implemented using a testing framework like JUnit. For each test case, assertions will be made to validate the expected behavior. If you would like to see the actual implementation of these unit tests, please specify a testing framework, or I can provide a general outline for incorporating JUnit tests.