ACKNOWLEDGEMENT

We are greatly indebted to our mini project guide **Ms.Nitu L Pariyal** for her able guidance throughout this work. It has been an altogether different experience to work with her and we would like to thank her for her help, suggestions and numerous discussions.

We gladly take this opportunity to thank **Dr.Mrs. Rajurkar A. M**. (Head of Computer Science & Engineering, MGM' s College of Engineering, Nanded).

We are heartily thankful to **Dr. Mrs. Lathkar G. S**. (Director, MGM's College of Engineering, Nanded) for providing facilities during the progress of the mini project in java, also for her kindly help, guidance and inspiration. Last but not least we are also thankful to all those who help directly or indirectly to develop this mini project and complete it successfully.

With Deep Reverence,
Shivam Jatale [115]

ABSTRACT

Card shuffle game

This Java program simulates a simple card game where two players draw random cards. It picks a random rank and suit for each player, compares the ranks, and declares the winner. The code uses arrays for suits and ranks, and the `Random` class for card selection. It is a basic implementation without complex structures, making it easy to understand for beginners.

This Java program creates a simplified two-player card game using basic arrays and random selection. The game has two key arrays: `suits` (Hearts, Diamonds, Clubs, Spades) and `ranks` (2 to Ace). Using the `Random` class, it selects a rank and a suit for each player's card.

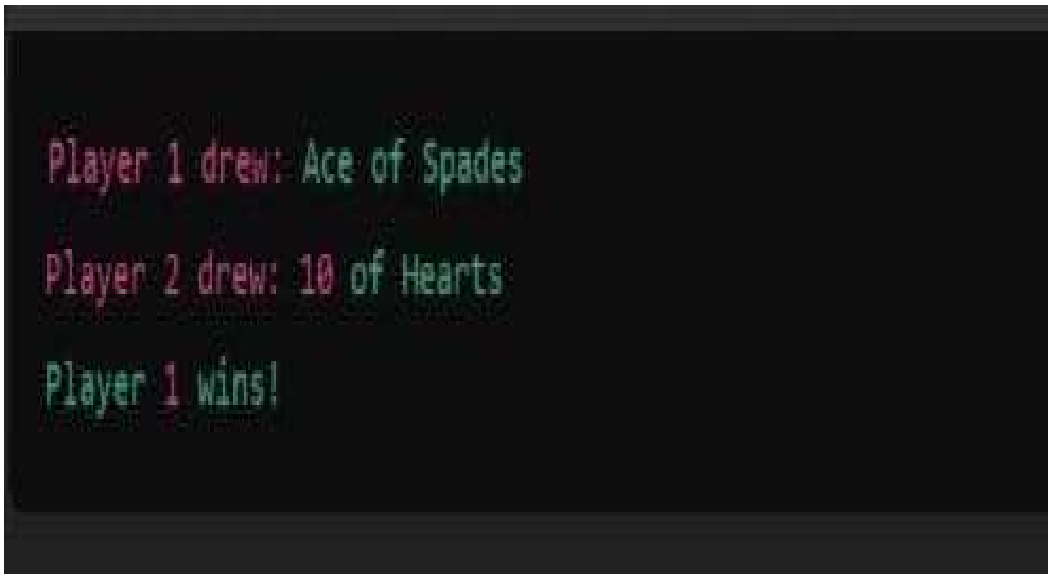
1.UML Class Diagram

```
+-------------------+
|   SimpleCardGame  |
+-------------------+
| - suits: String[] |
| - ranks: String[] |
+-------------------+
| + main(String[] args): void       |
| + compareRanks(String, String, String[]): int |
| + getRankValue(String, String[]): int         |
+-------------------+
```

2.Code of Card Suffle Game

```
import java.util.Random;

public class SimpleCardGame {
    public static void main(String[] args) {
        // Define the ranks and suits
        String[] suits = {"Hearts", "Diamonds", "Clubs", "Spades"};
        String[] ranks = {"2", "3", "4", "5", "6", "7", "8", "9", "10",
"Jack", "Queen", "King", "Ace"};

        // Create a random number generator
        Random random = new Random();

        // Player 1 draws a card
        String player1Rank = ranks[random.nextInt(ranks.length)];
        String player1Suit = suits[random.nextInt(suits.length)];
        System.out.println("Player 1 drew: " + player1Rank + " of " +
player1Suit);

        // Player 2 draws a card
        String player2Rank = ranks[random.nextInt(ranks.length)];
        String player2Suit = suits[random.nextInt(suits.length)];
        System.out.println("Player 2 drew: " + player2Rank + " of " +
player2Suit);

        // Compare the cards
        int winner = compareRanks(player1Rank, player2Rank, ranks);
        if (winner == 1) {
            System.out.println("Player 1 wins!");
        } else if (winner == 2) {
            System.out.println("Player 2 wins!");
        } else {
            System.out.println("It's a tie!");
        }
    }

    // Method to compare card ranks
    public static int compareRanks(String rank1, String rank2,
String[] ranks) {
        int rank1Value = getRankValue(rank1, ranks);
        int rank2Value = getRankValue(rank2, ranks);

        if (rank1Value > rank2Value) {
            return 1;
        } else if (rank2Value > rank1Value) {
            return 2;
        }
        return 0;
    }

    // Method to get the rank value
    public static int getRankValue(String rank, String[] ranks) {
        for (int i = 0; i < ranks.length; i++) {
            if (ranks[i].equals(rank)) {
                return i;
            }
        }
        return -1;
    }
}
```

3.Output



```
Player 1 drew: Ace of Spades

Player 2 drew: 10 of Hearts

Player 1 wins!
```

4. Working of code

This Java program is a simple implementation of a two-player card game where each player draws a random card, and the program compares the cards to determine the winner. It uses basic Java features like arrays and the `Random` class, avoiding complex structures, making it suitable for beginners.

Code Structure

1. **Main Class** (SimpleCardGame**)**:
    - Contains the main method, which is the entry point of the program.
    - Initializes arrays for suits and ranks:
        - **Suits**: Hearts, Diamonds, Clubs, Spades.
        - **Ranks**: Numbers 2 to 10, followed by Jack, Queen, King, and Ace.
2. **Card Drawing**:
    - Uses the Random class to select a random rank and suit for each player's card.
    - Prints the card drawn by both players.
3. **Comparison Method (compareRanks)**:
    - Takes the ranks of both players' cards and compares their positions in the ranks array.
    - Uses the helper method getRankValue to determine the index of each rank.
    - The player with the higher rank index wins. If both have the same rank, it's a tie.
4. **Helper Method (getRankValue)**:
    - Iterates through the ranks array to find the index of a given rank.
    - Returns the index, which is used for comparison.
5. **Output**:
    - The program prints the cards drawn by both players and announces the result (Player 1 wins, Player 2 wins, or it's a tie).

5.Explaination of code in Detail

This Java card game simulates a very simple card-drawing game between two players, where the cards are randomly drawn from a standard deck, and the player who draws the card with the highest rank wins.

Key Concepts:

1. **Deck of Cards**:
   - A deck of cards contains 52 cards divided into four suits: **Hearts**, **Diamonds**, **Clubs**, and **Spades**.
   - Each suit contains 13 ranks: **2**, **3**, **4**, ..., **10**, **Jack**, **Queen**, **King**, and **Ace**.
   - In this game, the suits are not considered when determining the winner. The only criterion for determining the winner is the rank of the card.
2. **Card Rank Comparison**:
   - The ranks of cards are compared based on their **value**. In this game:
     - **2** is the lowest rank.
     - **Ace** is the highest rank.
   - Cards are drawn randomly for each player. The rank of each drawn card determines who wins. For example, if Player 1 draws an **Ace** and Player 2 draws a **7**, Player 1 wins because the Ace has a higher rank than the 7.
3. **Random Card Selection**:
   - The game uses randomness to simulate the drawing of a card. This means that each time a card is drawn, the rank and suit are chosen randomly from the deck.
   - For both players, a random rank and suit are selected to represent the card that they draw.
4. **Determining the Winner**:
   - After both players have drawn a card, their ranks are compared:
     - If **Player 1's card** has a higher rank, **Player 1 wins**.
     - If **Player 2's card** has a higher rank, **Player 2 wins**.
     - If both cards have the same rank, it is a **tie**.
5. **Game Flow**:

    - Both players draw one card each.
    - The program compares the ranks of the two drawn cards.
    - It then prints out which player won based on the comparison or if it’s a tie.

Step-by-Step Flow of the Game:

1. **Initialize the Deck**: The program creates two arrays, one for the **suits** (Hearts, Diamonds, Clubs, and Spades) and one for the **ranks** (2, 3, 4, ..., Ace). These arrays represent a simplified version of a deck of cards.
2. **Draw Cards**:
    - Each player randomly selects a rank and suit. The randomness ensures that the selection of a card is unpredictable.
    - For each player, a rank and suit are picked randomly from the defined lists. These are then displayed as "Player 1 drew: [rank] of [suit]" and "Player 2 drew: [rank] of [suit]".
3. **Rank Comparison**:
    - After both players have drawn their cards, the ranks of the two cards are compared.
    - The ranks are assigned values based on their position in the **rank array** (e.g., 2 = 0, 3 = 1, ..., Ace = 12).
    - The player whose card has a higher rank wins. If both cards have the same rank, the game ends in a tie.
4. **Print the Result**:
    - The program then outputs the winner (or if it is a tie) based on the rank comparison.
    - If Player 1’s rank is greater than Player 2’s, Player 1 wins. If Player 2’s rank is greater, Player 2 wins. If both ranks are the same, it's a tie.

Example Scenario:

Imagine a scenario where Player 1 and Player 2 draw their cards:

- Player 1 draws a **King of Hearts**.
- Player 2 draws a **9 of Clubs**.

When the ranks of the two cards are compared:

- **King** has a higher rank than **9** (based on the defined order in the game).
- Therefore, **Player 1 wins**.

If both players had drawn cards of the same rank (e.g., both draw a **7**), the result would be a tie.

Purpose and Scope of the Game:

- **Educational Purpose**: This card game is a basic demonstration of how to work with random values, arrays, and comparisons in Java. It's a simple introduction to understanding how games can be implemented programmatically, and it helps illustrate key concepts like arrays, loops, and randomization in a fun way.
- **Scope**: The game is kept simple by focusing only on the rank of the cards, and suits are not considered. There is no complex logic like a full deck of 52 cards, hand management, or multiple rounds of play. The focus is on learning fundamental programming concepts.

6.Conclusion

The **Java card game** implemented in the provided code is a **simple simulation** of a two-player card game that randomly draws a card for each player, compares the ranks of the cards, and determines the winner.The **Java card game** code serves as a basic example of using randomization and comparison in a game-like setting. It is an excellent starting point for those who want to learn the fundamentals of programming with Java while also exploring how to structure simple games.