# BlackJack Game Project Description

### Project Overview

Create a console-based BlackJack game where a single player can play against the dealer (computer). The game will follow the standard rules of BlackJack, where the objective is to get as close to 21 points as possible without exceeding it. The player can choose to "hit" to receive another card or "stand" to keep their current total.

### Objectives

- Develop an understanding of arrays, loops, and conditionals in Java.
- Practice implementing card deck shuffling and drawing.
- Learn how to handle user input and display output in a console application.

### Requirements

1. **Game Setup:**
    - The game should use a standard 52-card deck.
    - The deck should be shuffled at the start of each game.
    - Both the player and the dealer are dealt two cards initially.
2. **Gameplay:**
    - The player should be able to choose to "hit" or "stand".
    - If the player "hits", they receive another card.
    - If the player's total exceeds 21, they bust, and the game ends.
    - After the player stands, the dealer reveals their hidden card and continues to draw cards until they reach at least 17 points.
3. **Winning Condition:**
    - The game ends when the player busts or chooses to stand.
    - The player wins if their total is closer to 21 than the dealer's without exceeding 21.
    - The dealer wins if the player busts or if the dealer's total is closer to 21.

## Step-by-Step Guide

### Step 1: Define Constants and Variables

Define constants and variables to represent the deck of cards, player and dealer hands, and other necessary information.

### Step 1.1: Define Constants

- **Description:** Define constants for the suits and ranks of the cards, and the maximum points (21) and dealer's minimum points (17).

**Step 1.2: Define Variables**

- **Description:** Define variables to represent the deck of cards, player and dealer hands, and other necessary game state information.

**Step 2: Initialize the Game**

Create methods to initialize the deck, shuffle it, and deal cards to the player and dealer.

**Method 2.1: `initializeDeck`**

- **Description:** Initialize the deck with 52 cards, each represented by a combination of a rank and a suit.

**Method 2.2: `shuffleDeck`**

- **Description:** Shuffle the deck using a random number generator.

**Method 2.3: `dealInitialCards`**

- **Description:** Deal two cards each to the player and the dealer at the start of the game.

**Method 2.4: `drawCard`**

- **Description:** Draw a card from the deck and remove it from the deck.

**Step 3: Calculate Points**

Create methods to calculate the points for the player's and dealer's hands.

**Method 3.1: `calculatePoints`**

- **Description:** Calculate the total points for a given hand, considering the values of cards and handling aces.

**Step 4: Player's Turn**

Create methods to handle the player's turn, including hitting and standing.

**Method 4.1: `playerTurn`**

- **Description:** Handle the player's turn, allowing them to hit or stand.

**Step 5: Dealer's Turn**

Create methods to handle the dealer's turn, where the dealer draws cards until reaching at least 17 points.

**Method 5.1:** `dealerTurn`

- **Description:** Handle the dealer's turn, drawing cards until reaching at least 17 points.

**Step 6: Determine Winner**

Create methods to determine the winner based on the points of the player and dealer.

**Method 6.1:** `determineWinner`

- **Description:** Determine the winner based on the points of the player and dealer.

**Step 7: Main Game Loop**

Create the main game loop to initialize the game, handle turns, and determine the winner.

**Method 7.1:** `playGame`

- **Description:** The main game loop to initialize the game, handle turns, and determine the winner.