In this project you will use Object Oriented Programming to implement a simplified version of the card game "Blackjack".

**Main Objective of the Game**
The main objective is to get as close a score to twenty-one as possible without going over twenty-one. There are two people involved in playing this game, a dealer and a player.

**Card Values**
Every card has a set value. A “TWO” is worth 2 points; a “THREE” is worth 3 points and so on. All the picture cards, “KING”, “QUEEN” and “JACK” are worth 10. For simplicity an “ACE” will be worth 1 point.

**Playing the Game**
The dealer deals 2 cards to the player and 2 cards to himself. Both the player and the dealer calculate their score by adding the value of their cards. Both the dealer and the player then have two choices:

- **They can stand:** If either the player or the dealer chooses to stand it means that they do not want to draw any more cards.

- **They can draw:** If either the player or the dealer chooses to draw then they will be dealt another card from the top of the deck. The player can choose to draw any number of cards. If the player’s score, however, exceeds 21 points then he immediately loses. The dealer, on the other hand, can only draw another card if his current score is below 17 points.

**Scoring**
Once the player finally chooses to stand, both him and the dealer reveal their cards. The one with the higher score wins. If the player has a score above 21 then the player loses no matter what the dealer has. If the dealer only has a score above 21, then the player wins. If both the dealer and the player happen to have the same score it is a tie.
Sample Output

Here is what a sample run of your program should look like.

//Example of Dealer having a higher score
------------------ GAME OPTIONS -------------------------
Enter 'p' to play BlackJack
Enter 's' to show score history
Enter 'q' to quit
> p

Dealing first hand of cards...

Your cards are:     ACE   TWO
Your Score is:          3

Enter 'd' to draw another card
Enter 's' to stand
> d

Your cards are:     ACE   TWO   FIVE
Your Score is:           8

Enter 'd' to draw another card
Enter 's' to stand
> d

Your cards are:     ACE   TWO   FIVE   SIX
Your Score is:         14

Enter 'd' to draw another card
Enter 's' to stand
> s

Revealing all cards....

Your cards are:     ACE   TWO   FIVE   SIX
Your Score is:         14

The dealer's cards are     THREE   FOUR   FIVE   FIVE
The dealer's score is:        17

Game Over: YOU LOSE!
//Example of player having a higher score
------------------ GAME OPTIONS -------------------
Enter 'p' to play BlackJack
Enter 's' to show score history
Enter 'q' to quit
> p

Dealing first hand of cards...

Your cards are:     ACE   TEN
Your Score is:          11

Enter 'd' to draw another card
Enter 's' to stand
> d

Your cards are:     ACE   TEN   SEVEN
Your Score is:           17

Enter 'd' to draw another card
Enter 's' to stand
> s

Revealing all cards....

Your cards are:     ACE   TEN   SEVEN
Your Score is:          18

The dealer's cards are     THREE   FOUR   FIVE   FIVE
The dealer's score is:        17

Game Over: YOU WIN!
//Example of both the player and the dealer having the same score
------------------  GAME OPTIONS -------------------------
Enter 'p' to play BlackJack
Enter 's' to show score history
Enter 'q' to quit
> p

Dealing first hand of cards...

Your cards are: 
SEVEN   EIGHT   THREE
Your Score is: 18

Enter 'd' to draw another card
Enter 's' to stand
> s

Revealing all cards....

Your cards are: 
SEVEN   EIGHT   THREE
Your Score is: 18

The dealer's cards are 
SIX   FOUR   EIGHT
The dealer's score is: 18

Game Over: TIE!

//Example of the player's score going above 21
------------------  GAME OPTIONS -------------------------
Enter 'p' to play BlackJack
Enter 's' to show score history
Enter 'q' to quit
> p

Dealing first hand of cards...

Your cards are: 
SEVEN   EIGHT
Your Score is: 15

Enter 'd' to draw another card
Enter 's' to stand
> d

Your cards are: 
SEVEN   EIGHT   JACK
Your Score is: 25

Enter 'd' to draw another card
Enter 's' to stand
> s

Revealing all cards....

Your cards are: 
SEVEN   EIGHT   JACK
Your Score is: 25
The dealer's cards are NINE  TEN
The dealer's score is: 19

Game Over: YOU LOSE!

//Example of the dealer's score going above 21
------------------  GAME OPTIONS -------------------------
Enter 'p' to play BlackJack
Enter 's' to show score history
Enter 'q' to quit
> p

Dealing first hand of cards...

Your cards are: SEVEN  EIGHT
Your Score is: 15

Enter 'd' to draw another card
Enter 's' to stand
> s

Revealing all cards....

Your cards are: SEVEN  EIGHT
Your Score is: 15

The dealer's cards are NINE  SIX  TEN
The dealer's score is: 25

Game Over: YOU WIN!

//Example of both the player and the dealer going above 21
------------------  GAME OPTIONS -------------------------
Enter 'p' to play BlackJack
Enter 's' to show score history
Enter 'q' to quit
> p

Dealing first hand of cards...

Your cards are: SEVEN  EIGHT  NINE
Your Score is: 24

Enter 'd' to draw another card
Enter 's' to stand
> s
Revealing all cards....

Your cards are: SEVEN EIGHT NINE
Your Score is: 24

The dealer's cards are NINE SIX TEN
The dealer's score is: 25

Game Over: YOU LOSE!

//Example of showing the score history
------------------  GAME OPTIONS -------------------------
Enter 'p' to play BlackJack
Enter 's' to show score history
Enter 'q' to quit
> s

Total number of games played: 6
Number of games won: 2
Number of games lost: 3
Number of games tied: 1
Implementing the game
There are many different ways to implement this game. Feel free to implement the game in any manner you see fit. We do provide some hints and helpful guidelines on one possible way to arrange your classes. If you choose to design your own implementation for the game please make sure you make use of object oriented concepts in order to receive credit for the project. In other words your project must contain multiple classes.

To implement the Blackjack game one possible approach is to implement four classes, namely: Card, Deck, Player, and PlayBlackJack. A brief description of what each class might contain is given below.

```java
/* The Card class could define a card along with the attributes associated with it. A card has two attributes:
/* String Name: ACE, TWO, THREE, FOUR, FIVE……NINE, TEN, JACK, QUEEN and KING
/* int Value: 1, 2, 3, 5, 6, 7, 8, 9 and 10
/* You do not really need to worry about a card’s suit or color as they are of no importance in playing this game.
*/
public class Card {
    //String name stores the name of the card
    private String name;
    //Integer value stores the value of the card
    private int value;

    /* This is the Class’s constructor. It could take the card’s name and value as arguments. */
    public Card (String n, int v) { }

    /* This method could return the name of the card. The name is of type String. */
    public String getName() { }

    /* This method could return the value of the card. The value is of type int. */
    public int getValue() { }
}
```
public class Deck {

    //Array that stores the deck of cards
    private Card[] deckOfCards = new Card[52];

    //Counter that keeps track of the next card to be dealt out
    private int deckCounter;

    /* An empty constructor for the class. */
    public Deck() {
    }

    /* This method could create a deck of cards and could store it in the array deckOfCards */
    public void createDeck() {
        //create 52 cards and store them in the array deckOfCards
        //There will be 13 distinct cards (i.e. cards with different names and values)
        //There will be four cards with identical names and values
        //When creating the cards do not worry about the card’s suit or color
    }

    /* This method shuffles the deck of cards in any manner */
    public void shuffleDeck() {
    }

    /* This returns the next card to be drawn to one of the players from the deck */
    /* It could also increment any necessary counters */
    public Card drawNextCard() {
    }

} //End of class Deck
/* This class could define all the methods that are required to support a player */

public class Player {

    //Array that holds the cards drawn by a player
    public Card[] cardsArray;

    //Counter for number of cards drawn by a player
    public int counter;

    /* Empty constructor for the Player class */
    public void Player() {
    }

    /* This method draws a card for a player and puts it in the appropriate array while incrementing any necessary counters */
    public void drawCard() {
    }

    /* This method clears the array that holds the player's cards */
    /* This method is called every time a new game is started */
    public void clearCards() {
    }

    /* This method calculates and returns the player's score based on the cards that he drew. */
    public int calculateScore() {
    }

    /* This method prints out the names of the player's cards */
    public void printCards() {
    }

    /* This method prints out the player's score */
    public int printScore() {
    }

} //End of class Player
public class PlayBlackJack {

    // You could create an instance of the Deck class

    // You could also create two instances of the Player class. One for the user and one for the dealer

    /* Constructor for the PlayBlackJack class */
    public PlayBlackJack ( ) {  }

    /* This method implements the actual game
    ** This would most probably be the most extensive method in this project */
    public void playGame() {  }

    /* This method prints the score history */
    public void printScoreHistory() {  }

    /* This is the main method */
    public static void main(String args[]) {  }

} // End of class PlayBlackJack
Submitting your project:

You should include the following information in the beginning of your .java file.

1. Your name, SPIRE id and e-mail address and all the information of your partner.
2. A brief description of the project (~ 1 paragraph).
3. Any special features of your program that we should look out for.

If your program includes multiple files, for example, if you used the EasyIn.java we discussed in the class, you can upload the multiple source files to the server. For a group project, please make sure that every group member needs to submit one separate copy of the program to PROG3 directory. Please use ftp to submit your programs, and please follow the detailed instruction on the course website.

Grading:

There are 100 points total for this project.
1. Your program should be compiled successfully. It is your responsibility to include all the source files when you hand in the project. Please make sure you also include the name of your partner. (40 points)
2. Basic functions of the program. It should implement the game correctly, yield correct results and have a user-friendly interface. (50 points)
3. Ten points are reserved for the overall impression. (10 points)

Extra credits (20 points):
Implement a GUI of your choice for the game.