

# Black Jack Code

The document is about the code of a game, written in JAVA, which depicts the famous card game - *Black Jack*. The documents covers different areas relating to the game and the code. The code was written by the Computer Science class at EF Academy in the year 2016. The mentor for the project is Dr.Schultz and the ones who followed his guidance into building this project are the students of Grade 11, including me. The project was done cooperatively using GitHub thus giving more efficiency to our work.

```
1 import java.util.*;
2 import java.util.Scanner;
3
4 public class Cards
5 {
6
7
8     static int count=52; //the count represents the number of cards remaining in the
    deck
9
10    public static int rand(int high)
11    {
12        { return (int) (high*Math.random()+1);
13        }
14    }
15
16    public static void shuffle(String[] the_deck, int switches)
17    {
18        { String temp; //As the name suggests, it is temporary space provided to the
    switching elements when shuffling.
19        int a, int b;
20        for(int i=0; i<switches; i++)
21        {
22            a = rand(52);
23            b = rand(52);
24            temp = the_deck[a-1];
25            the_deck[a-1] = the_deck[b-1];
26            the_deck[b-1] = temp;
27        }
28    }
29
30    public static String deal(String[] the_deck){
31        count=count-1;
32        return the_deck[count];
33    }
34
35    public static int aces(String the_card){
36        if(the_card.charAt(0)=='A'){
37            return 1;
38        }
39        else{
40            return 0;
41        }
42    }
43
44    public static int aces(String[] the_hand)
45    {
46        int sum=0;
47        for(int i=0; i<the_hand.length;i++)
48        {
49            sum = sum + aces(the_hand[i]);
50        }
51        return sum;
52    }
53
54    public static int aces(ArrayList the_hand)
55    {
56        int sum=0;
57        for(int i=0; i<the_hand.size();i++)
58        {
59            sum = sum + aces(the_hand.get(i).toString());
60        }
61        return sum;
62    }
63
64    public static int value(String the_card)
65    {
66        char first = the_card.charAt(0);
67        if (first=='1'|first=='J'|first=='Q'|first=='K')
68        {
69            return 10;
70        }
71        else if(first=='A')
72        {
73            return 11;
74        }
75        else
76        {
77            return Character.getNumericValue(first);
78        }
79    }
80
81    public static int value(String[] the_hand)
82    {
83        int sum=0;
84        for(int i=0; i<the_hand.length;i++)
85        {
86            sum = sum + value(the_hand[i]);
87        }
88        return sum;
89    }
90
91    public static int value(ArrayList the_hand)
92    {
93        int sum=0;
```

/Users/NEEL/Desktop/JAVA/Cards.java



Academy

INTERNATIONAL BOARDING SCHOOLS

## ***Overview***

- I. Introduction to Cards.
- II. About Black Jack and its rules.
- III. Analysis of different sections of the code.
- IV. Output of the code and Instructions for using the game.

## Introduction to Cards

A playing card is a piece of specially prepared heavy paper, thin cardboard, plastic-coated paper, cotton-paper blend, or thin plastic, marked with distinguishing motifs and used as one of a set for playing card games. Playing cards are typically palm-sized for convenient handling.

A complete set of cards is called a pack (UK English), deck (US English), or set (Universal), and the subset of cards held at one time by a player during a game is commonly called a hand. A pack of cards may be used for playing a variety of card games, with varying elements of skill and chance, some of which are played for money (e.g., poker and blackjack games at a casino).

In playing cards, a suit is one of several categories into which the cards of a deck are divided. Most often, each card bears one of several pips (symbols) showing to which suit it belongs; the suit may alternatively or additionally be indicated by the colour printed on the card. The rank for each card is determined by the number of pips on it. Ranking indicates which cards within a suit are better, higher or more valuable than others, whereas there is no order between the suits unless defined in the rules of a specific card game. Unless playing with multiple decks, there is exactly one card of any given rank in any given suit.[1] A deck may include special cards that belong to no suit, often called jokers.

The deck of 52 French playing cards is the most common deck of playing cards used today. It includes thirteen ranks of each of the four French suits: clubs (♣), diamonds (♦), hearts (♥) and spades (♠), with reversible "court" or face cards. Some modern designs, however, have done away with reversible face cards. Each suit includes an ace, depicting a single symbol of its suit; a king, queen, and jack, each depicted with a symbol of its suit; and ranks two through ten, with each card depicting that many symbols (pips) of its suit.

## **About Black Jack and its rules.**

Blackjack, also known as twenty-one, is the most widely played casino banking game in the world. Blackjack is a comparing card game between a player and dealer, meaning players compete against the dealer but not against other players. It is played with one or more decks of 52 cards.

The object of the game is to beat the dealer in one of the following ways:

- Get 21 points on the player's first two cards (called a blackjack), without a dealer blackjack;
- Reach a final score higher than the dealer without exceeding 21; or
- Let the dealer draw additional cards until his or her hand exceeds 21.

The player or players are dealt a two-card hand and add together the value of their cards. Face cards (kings, queens, and jacks) are counted as ten points. A player and the dealer can count an ace as 1 point or 11 points. All other cards are counted as the numeric value shown on the card. After receiving their first two cards, players have the option of getting a "hit", or taking an additional card. In a given round, the player or the dealer wins by having a score of 21 or by having the higher score that is less than 21. Scoring higher than 21 (called "busting" or "going bust") results in a loss. A player may win by having any final score equal to or less than 21 if the dealer busts. If a player holds an ace valued as 11, the hand is called "soft", meaning that the player cannot go bust by taking an additional card; 11 plus the value of any other card will always be less than or equal to 21. Otherwise, the hand is "hard". Players win by not busting and having a total higher than the dealer's. The dealer loses by busting or having a lesser hand than the player who has not busted. If the player and dealer have the same total, this is called a "push", and the player typically does not win or lose money on that hand.

## Analysis of different sections of the code.

The coding language used is JAVA. It is written by the members of the Computer Science class. The code contains the whole game of Black Jack, excluding of course Graphics and other high end user interface objects....

The code has 10 different methods which serve different function in the building of the game. The 10 methods include the main method where the code finally gets executed. Before proceeding further, please ensure you have a JAVA IDE and compiler so that you can actually run the code. (The code is shown in PDF format which may not reflect the code exactly as written so it can give error if you literally copy the code from PDF.)

-----  
*Starting from the top:*

```
1  import java.util.*;
2  import java.util.Scanner;
3
4  public class Cards
5  {
6
7      static int count=52; //the count represents the number of cards remaining in the
8      deck
9  }
```

We can see that two libraries are imported from outside the code. The program may lead to an error if you don't import these libraries. The name of the code as seen here is

→ **"Cards"**.

In the Line 8, there is an integer specified which is supposed to be used later in the code as it gives the count of the initial number of cards in the deck which can be later changed to show that the cards have been given out.


-----  
*Shuffling Part of the code:*

This part of the code does the shuffling of the cards. It consists of two methods, one that generates a **random number** between 1 to 52 and the other part picks up the card from the deck which is in the order number as the generated random number and switches their places. The hard part about this code is when we are switching two elements, we need to provide temporary space for one of the elements to be and then change their places.

```
10  public static int rand(int high)
11  {
12      return (int) (high*Math.random()+1);
13  }
14
15  public static void shuffle(String[] the_deck, int switches)
16  {
17      String temp; //As the name suggests, it is temporary space provided to the
18      switching elements when shuffling.
19      int a; int b;
20      for(int i=0; i<switches; i++)
21      {
22          a = rand(52);
23          b = rand(52);
24          temp = the_deck[a-1];
25          the_deck[a-1] = the_deck[b-1];
26          the_deck[b-1] = temp;
27      }
28  }
29
30
```

### Third Part of the Code:

Cards are dealt to the player and the number of the cards from the deck is reduced by the number of cards dealt. As you can see that two of the methods have the same name but the **arguments provided are different** so that it doesn't give an error. Char is used to give out a list so here it increases our efficiency of the code. The methods combined return the elements of the deck thus giving out any card and as follows checks for Ace. And at the end, the sum is counted for the first card which is dealt.



```
30
31 public static String deal(String[] the_deck){
32     count=count-1;
33     return the_deck[count];}
34
35 public static int aces(String the_card){
36     if(the_card.charAt(0)=='A'){
37         return 1;}
38     else{
39         return 0;}
40     }
41
42 public static int aces(String[] the_hand)
43 {
44     int sum=0;
45     for(int i=0; i<the_hand.length;i++)
46     {
47         sum = sum + aces(the_hand[i]);
48     }
49     return sum;
50 }
51
```

The diagram shows a vertical arrow pointing upwards from the `aces(the_hand[i])` call in the `aces(String[] the_hand)` method to the `return the_deck[count];` line in the `deal(String[] the_deck)` method, indicating that the value returned by `deal` is used as an argument for `aces`.

---

### Card Identifier:

In this section of the code, the methods given below identify the values of the cards and assign them an integer value. The integer value is then used to calculate the sum of all the other cards you had before and then adds the sum. The Face cards here are given values of 10. And the value of the Ace is set as 11. The value of the sum is returned as the final product which we will use later on.

```

51
52 public static int aces(ArrayList the_hand)
53 {
54     int sum=0;
55     for(int i=0; i<the_hand.size();i++)
56     {
57         sum = sum + aces(the_hand.get(i).toString());
58     }
59     return sum;
60 }
61
62 public static int value(String the_card)
63 {
64     char first = the_card.charAt(0);
65     if (first=='1'|first=='J'|first=='Q'|first=='K')
66     {
67         return 10;
68     }
69     else if(first=='A')
70     {
71         return 11;
72     }
73     else
74     {
75         return Character.getNumericValue(first);
76     }
77 }
78
79 public static int value(String[] the_hand)
80 {
81     int sum=0;
82     for(int i=0; i<the_hand.length;i++)
83     {
84         sum = sum + value(the_hand[i]);
85     }
86     return sum;
87 }
88
89

```

---

### *Ace Value Changer:*

There is one more rule that needs to be added before moving to the final method. If you have an ace and the total sum exceeds 21, then the value of one of the ace changes to 1 according to BlackJack's Rules. This is done in these two methods.

```

89
90 public static int value(ArrayList the_hand)
91 {
92     int sum=0;
93     int num_aces=aces(the_hand);
94     for(int i=0; i<the_hand.size();i++)
95     {
96         sum = sum + value(the_hand.get(i).toString());
97     }
98     while(num_aces>0 && sum>21)
99     {
100         sum=sum-10;
101         num_aces=num_aces-1;
102     }
103     return sum;
104 }
105
106

```



When the value of Ace is changed to 1, the program subtracts 10 from the final sum of the values of cards.

---

The main method starts by creating a fresh deck of 52 cards. It is an Array which has four sub arrays each one containing one suit of cards. The sequence of the suits is Clubs,Diamonds,Hearts and Spades. After filling all the subarrays, it starts giving cards their value, for example: it assigns the cards 0,10,11,12 as Ace,Jack,Queen,King respectively. The code contains some commented out sections which print the deck and then we use the Shuffling method to shuffle the deck 1000 times. It again prints out the shuffled deck but its commented out just to make it look neat at the end.

Now, the dealer deals a hand, which is nothing but takes the topmost card from the deck and presents it to the player. This process is done twice.

Finally, the game logic enters here, the dealer picks up his own cards and shows one of them, and the game shows our two cards. The next question asked by the new Scanner line will determine if the game goes forward or should it show the results... If we continue the game and take more card, as per the rules, we go above 21 and we lose, we have less score than the dealer and we lose, and so on. This whole process happens in a while loop allowing the the program to run until opted out which in turn triggers the counting system of the result. When opted to take a hit, the dealer as well as the player a dealt a hand, given new cards and changing the sum according to that.

-----



## **Output of the code and Instructions for using the game.**

Before running the code, you will need to compile it in JAVA which will make another file with the extension .class . When you run the program, the output will show you your cards and the cards that the dealer has. And at the end of the output it will be questioned whether you want to 'Hit' or 'Stay' by pressing the choice key, it will again ask if the sum is less than 21 but if you go above 21 then you lose and it will be stated as 'BUST!!!!'. Also if you opt for 'Stay', the code will see if the sum is greater than of the dealer, if not, you lose.