



4Leaf Co.

DESIGN

Jordan Quick (PM)

Frances Coronel

Calvin Chambers

Anesha Passalacqua



Table of Contents

1.0 Introduction

1.1 Overview	3
--------------	---

2.0 Design

2.1 Overview	3
--------------	---

3.0 Method of Implementation

3.1 Program 1: Database	3
3.2 Program 2: Hands Probability	5
3.3 Program 3: Strategy Table	12

4.0 Desktop App

4.1 Overview	15
--------------	----

5.0 Visual Interface

5.1 Overview	19
--------------	----

6.0 Tracking and Control Mechanisms

6.1 Document Revision History	21
-------------------------------	----



1.0 Introduction

1.1 Overview

4Leaf Co. has been tasked with implementing a program that will correctly teach a user how to play Blackjack.

This program will be used as both a learning experience and interactive activity to provide users with a better understanding of this fun game.

Applying Web Tools and Databases with an Interactive Desktop Environment will result in the creation of an efficient and interactive game of Blackjack.

2.0 Design

2.1 Overview

Throughout the product development cycles, the product will undergo many changes. A prototype of the product will be designed in the end but ultimately each phase of the product will be developed as if it was the end prototype.

Each prototype will be checked for inconsistencies and flaws. Constant updates and software patches will reach this prototype as we get near to the end of the product's development.

3.0 Method of Implementation

3.1 Database

The database will compute and store n combinations for statistical analysis.

The number of supported decks are 1, 2, 4, 6, and 8. For each supported deck, the database will formulate n -combinations. The two algorithms that will be implemented are the frequency of hands and then the probability of winning depending on the current hand.


The probabilities are formatted using 8 decimal points given that the probabilities are very small.

For example, the database is formatted below with four different scenarios for the dealer and user hand. The probabilities for each user choice (split, double down, hit, stand) is then associated that way.

Dealer's Hand	User's Hand	Split	Double Down	Hit	Stand
3	10	Probability of the user choice leading to a Blackjack scenario based on dealer and user hand P (user_choice [dealer hand, user hand])			
6	5, 5				
10	8				

Table 1: Database Probabilities

The database is built using MySQL statements programmed within MySQL Workbench.



The MySQL database is communicating with Java code by having certain statements within the Java program that specifically communicate with the MySQL database.

Example of how this code looks like in the program in order to communicate with the database is shown below:

```
// connecting to MySQL Driver
static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
// specifying database URL
static final String DB_URL = "jdbc:mysql://localhost:3306/bj";
// initializing username for database
static final String USER = "root";
// initializing password for database
static final String PASS = "4leafco";
```

3.2 Program 2 - Hands Occurrence Probabilities

The program used to generate the frequencies is written in Java and communicates with the database explained in more detail in Section 3.1.


Otherwise known as the frequency table assignment, the second program uses the database and calculates the frequency that a hand appears out of n hands for the dealer's one card and the user's first two cards. Below is a screenshot taken from the console output of the Frequency program that communicates with the database.

	2	3	4	5	6	7	8	9	10	A
8	120520	20661	19833	20490	19533	20837	21111	20703	20491	19791
9	11280	1256	1264	1288	1366	1104	1368	1432	1136	1240
10	11344	1057	1435	1127	1339	1351	1316	1281	1218	1358
11	11440	1368	1428	1584	1382	1530	1542	1506	1344	1557
12	12085	2235	2237	2170	2201	2575	2125	2170	2450	2285
13	11964	1720	1980	1884	1980	1928	1892	1880	1964	1904
14	11344	1353	1188	1299	1356	1200	1314	1275	1284	1236
15	1822	818	830	764	786	808	792	834	742	786
16	1389	375	398	379	296	384	400	373	379	350
17	135383	35686	35273	35818	34416	36973	36542	35904	35398	35236
2,2	12952	3261	2629	2989	2970	2918	2937	2826	2862	2305
3,3	13451	2873	2924	3451	3179	3400	3519	3400	3451	3162
4,4	12528	2560	2576	2864	2720	2960	3008	2688	2912	3040
5,5	12265	1800	1830	1695	1710	1635	1950	2085	1890	1980
6,6	12679	2660	2562	2226	2492	2674	2380	2744	2296	2534
7,7	12421	2239	2718	2198	2537	2185	2393	2418	2864	2216
8,8	11956	1668	1752	2052	1632	1800	1836	1800	2268	1932
9,9	11727	1749	1760	1496	1716	1815	1914	1815	1881	1925
10,10	11980	1680	1840	1420	1810	1740	1650	1520	1770	1722
A,2	11854	2151	1980	2007	1854	2025	2160	2349	1674	1827
A,3	11160	1240	1247	1361	1224	1241	1488	1288	1097	1272
A,4	11316	1365	1216	1624	1396	1341	1324	1514	1120	1425
A,5	11232	1326	1156	1296	1105	1152	1331	1235	1032	1082
A,6	11134	975	995	885	900	1076	935	965	960	905
A,7	1684	600	708	868	752	788	756	640	700	621
A,8	1486	556	602	585	549	565	479	574	514	623
A,9	1436	500	448	386	414	508	410	470	538	396
A,10	1230	237	244	244	230	258	272	243	265	265
A,A	13452	3414	3373	3426	3334	3512	3516	3481	3434	3364

Count: 1000007

Table 2. Frequency Table Generated From Program (1 million hands).

In order to verify this frequency table, an Excel sheet was created to manually calculate the probabilities of each hand occurrence in the frequency table. This file can be referenced under "Hand Frequency - Manual Calculation.xlsx". This file has two different sheets - the Probabilities themselves and then the Formulas used to calculate the



probabilities. These manual calculations in and of themselves also aid in creating a basic strategy table.

This file stands for the probability of each possible starting hand. The player could start with anything from a 5 up to 21, or a Blackjack. Since the Frequency Table starts with 8, that's where we will start. You could also have a soft, which is anything from a 13 up to 21, or a Blackjack. For both cards the user has, the cards could be a 2 through an Ace.

To calculate the occurrence, (refer to rows 59 to 70), it is simply $(1/13)$ squared since we have two cards and each has a $1/13$ or $4/52$ chance in getting selected.

When there's a face card involved however, then the probability becomes $1/13$ times $4/13$ since each card has a $1/13$ chance of getting selected and there is a $16/52$ of a face card/Ace getting selected. This same process is repeated for a soft hand, hard hand, and a hand pair. To serve as a reference, there is a screenshot of the frequencies themselves showcased below as well as the formulas used to generate those probabilities:

One Million Hands Count Frequency													
		2	3	4	5	6	7	8	9	10	A	Total	Hands
8		2,731	2,731	2,731	2,731	2,731	2,731	2,731	2,731	10,084	1,891	33,822	1,000,000
9		2,731	2,731	2,731	2,731	2,731	2,731	2,731	2,731	10,084	1,891	33,822	
10		2,731	2,731	2,731	2,731	2,731	2,731	2,731	2,731	10,084	1,891	33,822	
11		3,641	3,641	3,641	3,641	3,641	3,641	3,641	3,641	13,445	2,521	45,096	
12		6,372	6,372	6,372	6,372	6,372	6,372	6,372	6,372	23,529	4,412	78,919	
13		6,372	6,372	6,372	6,372	6,372	6,372	6,372	6,372	23,529	4,412	78,919	
14		5,462	5,462	5,462	5,462	5,462	5,462	5,462	5,462	20,167	3,781	67,645	
15		5,462	5,462	5,462	5,462	5,462	5,462	5,462	5,462	20,167	3,781	67,645	
16		4,552	4,552	4,552	4,552	4,552	4,552	4,552	4,552	16,806	3,151	56,371	
17		11,834	11,834	11,834	11,834	11,834	11,834	11,834	11,834	43,696	8,193	146,563	
2,2		455	455	455	455	455	455	455	455	1,681	315	5,637	
3,3		455	455	455	455	455	455	455	455	1,681	315	5,637	
4,4		455	455	455	455	455	455	455	455	1,681	315	5,637	
5,5		455	455	455	455	455	455	455	455	1,681	315	5,637	


Table 3. One Million Hands Count Frequency (Snapshot)

Please refer to “Hand Frequency - Manual Calculation.xlsx”.

Strategy Table Order						
	2	3	4	5	6	7
8	0.002730996814	0.002730996814	0.002730996814	0.002730996814	0.002730996814	0.002730996814
9	0.002730996814	0.002730996814	0.002730996814	0.002730996814	0.002730996814	0.002730996814
10	0.002730996814	0.002730996814	0.002730996814	0.002730996814	0.002730996814	0.002730996814
11	0.003641329085	0.003641329085	0.003641329085	0.003641329085	0.003641329085	0.003641329085
12	0.006372325899	0.006372325899	0.006372325899	0.006372325899	0.006372325899	0.006372325899
13	0.006372325899	0.006372325899	0.006372325899	0.006372325899	0.006372325899	0.006372325899
14	0.005461993628	0.005461993628	0.005461993628	0.005461993628	0.005461993628	0.005461993628
15	0.005461993628	0.005461993628	0.005461993628	0.005461993628	0.005461993628	0.005461993628
16	0.004551661356	0.004551661356	0.004551661356	0.004551661356	0.004551661356	0.004551661356
17	0.01183431953	0.01183431953	0.01183431953	0.01183431953	0.01183431953	0.01183431953
2,2	0.0004551661356	0.0004551661356	0.0004551661356	0.0004551661356	0.0004551661356	0.0004551661356
3,3	0.0004551661356	0.0004551661356	0.0004551661356	0.0004551661356	0.0004551661356	0.0004551661356
4,4	0.0004551661356	0.0004551661356	0.0004551661356	0.0004551661356	0.0004551661356	0.0004551661356
5,5	0.0004551661356	0.0004551661356	0.0004551661356	0.0004551661356	0.0004551661356	0.0004551661356
6,6	0.0004551661356	0.0004551661356	0.0004551661356	0.0004551661356	0.0004551661356	0.0004551661356
7,7	0.0004551661356	0.0004551661356	0.0004551661356	0.0004551661356	0.0004551661356	0.0004551661356
8,8	0.0004551661356	0.0004551661356	0.0004551661356	0.0004551661356	0.0004551661356	0.0004551661356
9,9	0.0004551661356	0.0004551661356	0.0004551661356	0.0004551661356	0.0004551661356	0.0004551661356
10,10	0.00728265817	0.00728265817	0.00728265817	0.00728265817	0.00728265817	0.00728265817
A,A	0.0004551661356	0.0004551661356	0.0004551661356	0.0004551661356	0.0004551661356	0.0004551661356

Table 4. Strategy Table Order (Snapshot)

Please refer to “Hand Frequency - Manual Calculation.xlsx”.



The frequency table in this program forms the basis for the strategy table in program three. Below is the pseudocode for this frequency table which was used to develop Program 2.

```
import statements

/** DECLARE & INITIALIZE ARRAYS */

// array containing player cards
Array[i]
// array containing dealer cards
Array[j]
// array containing card counts for frequency table
Array[i][j] //frequency table

/**
 * Location Details
 * Sum  Array[i][j] = Array [0-9][0-8]
 * Pair Array[i][j] = Array[10-18][0-8]
 * Aces Array[i][j] = Array[19-27][0-9]
 */

/** DECLARATIONS */
Boolean pair, ace;
int sum = 0, tally = 0;

/** ASK FOR NUMBER OF DECKS TO USE */
// 1, 2, 4, 6, 8

/** GRAB VALUE OF CARDS FROM DATABASE */
```

```

/** USE DATABASE TO GRAB 1st 4 CARDS WITH A HAND AT ANY SINGLE
TIME */
/**
 * CARDS BEING READ
 * 1 - Player 1st Card
 * 2 - Dealer 1st Card (Up Card)
 * 3 - Player 2nd Card
 * 4 - Dealer 2nd Card (Down Card)
 *
 * User's 2 cards 1, 3
 * Dealer's up card 4
 * Card #2 is down
 */

/**
 * LOOP STARTING NOW
 */

/**
 * get sum of cards
 * get value of dealer card
 * determine if there is a pair
 * determine if there is an ace
 * if there is no pair or ace tally at location array[i][j]
 */

/** LOCATIONS OF i and j */
[i] = 0-9
[j] = 0-9

/**
 * IF USER HAS SUM
 * AND DEALER HAS ACE

```

```

    * TALLY AT LOCATIONS
*/
[i]= 0-9
[j]= 9

/** DETERMINE IF CARDS ARE PAIR */
if (value of usrCard1 == value of usrCard2)
    pair is true;

/** IF CARDS ARE PAIR , CHANGE TALLY LOCATIONS*/
if (there is a pair go to the pair range in array)
    Array[i] locations: 10-18

/**
    * DETERMINE PAIR USER HAS
    * GET CARD VALUES
    * GET SUM OF DEALER CARD
    * TALLY AT LOCATION ARRAY[i][j]++
    */

//if user has a pair and dealer has an ace
tally at location array[i][j]++;

/** DETERMINE WHETHER THERE IS AN ACE OR NOT */

// if user has an ace and a value of 2nd card and dealer has a
sum
tally at location array[i][j]++;

// if user has an ace and value and dealer has an ace
tally at location array[i][j]++;

// if user has a pair of aces and dealer has a sum
tally at location array [i][j]++;

```

```

// if User has a pair of aces the dealer has an ace
tally at location array[i][j]++;

// Ace is true
// tally at locations
[i] = 19-27
[j] = 9

/**
 * ONCE HAND IS RETRIEVED FROM DATABASE
 * INCREMENT PLACE IN ARRAY
 * REPEAT PROCESS
 */
Array[i][j]++;


/** LOOP ENDS HERE */

/** PRINT OUT ARRAY IN TABLE FORMAT */
print array[i][j]

```

3.3 Program 3 - Create Strategy Table

To generate the strategy table, we will evaluate the probabilities generated from Program 2 that cross-reference the database. As you play Blackjack, exact probabilities of outcomes of the dealer's hand and exact expected values for your hand(s) are displayed. The numbers displayed in the generated CSV are the exact probabilities (0 = impossible, 1 = certain) of each possible outcome of the dealer's hand. These probabilities take into account all cards dealt from the shoe.



To finalize the table, only the highest probabilities are taken from hand scenario on both the dealer/player side. This would lead to the most statistically ideal likelihood that a player will score Blackjack assuming the player references the strategy table at all times.

The program that creates the strategy table creates 4 different tables accounting for the four player moves (Split, Double Down, Hit, and Stand). The program selects the highest probability from the four tables so if the highest probability across the tables for that specific player and dealer hand is for standing, then for that specific hand there will be a S standing for Stand. This process will be repeated for each specific player/deal hand instance until an action is created for each instance for the Strategy Table.

All five tables will be converted to an Excel compatible format.

This strategy table is then input into the Blackjack Console game where it will be used to generate prompts for the player suggesting what move they should have made if they didn't choose a move that was the most statistically ideal (i.e. player chooses to stand when they should have hit so console game interpret strategy table based on hands and tells player they should have hit in that scenario).

The initial Java code can be referenced below:

```
public class StrategyMethod {  
  
    public static void main(String[] args) {
```

```
public static char StrategyMethod(int playerCard1Value, int
playerCard2Value, int totalDealer) {

    /**
     * P for split
     * D for double down
     * H for hit
     * S for stand
     */
    char bestMove = ' ';
    // int ace = 11 || 1;

    int totalPlayer = playerCard1Value + playerCard2Value;
    System.out.println("Dealer Total is: " + totalDealer);
    System.out.println("Player's 1st Card is: " +
playerCard1Value);
    System.out.println("Player's 2nd Card is: " +
playerCard2Value);
    System.out.println("Player Total is: " + totalPlayer);

    /**
     * if player hand = 8
     * double on dealer hands 5 to 6
     * otherwise hit
     */
    // int dealerHandfor8 = 5 || 6;

    if (playerCard1Value <= 8 || totalPlayer <= 8) {
        if (totalDealer == 5 || totalDealer == 6)
            bestMove = 'D';
        else
            bestMove = 'H';
    }
}
```

```

/**
 * repeat if statement for each hand scenario
 * define best move
 */

/**
 * return best move
 */
return bestMove;
}

}
}

```

Ultimately, the strategy table will look a lot like Figure 1 below which is just the table taken from the client's original instructions.

		Dealer's Card										
		2	3	4	5	6	7	8	9	10	A	
Player's Cards	8	H	H	H	H	H	H	H	H	H	H	
	9	H	D	D	D	D	H	H	H	H	H	
	10	D	D	D	D	D	D	D	D	H	H	
	11	D	D	D	D	D	D	D	D	D	H	
	12	H	H	S	S	S	H	H	H	H	H	
	13	H	H	S	S	S	H	H	H	H	H	
	14	S	S	S	S	S	H	H	H	H	H	
	15	S	S	S	S	S	H	H	H	H	H	
	16	S	S	S	S	S	H	H	H	H	H	
	17	S	S	S	S	S	S	S	S	S	S	
	2,2	H	H	P	P	P	P	H	H	H	H	
	3,3	H	H	P	P	P	P	H	H	H	H	
	4,4	H	H	H	H	H	H	H	H	H	H	
	5,5	D	D	D	D	D	D	D	D	H	H	
	6,6	H	P	P	P	P	H	H	H	H	H	
	7,7	P	P	P	P	P	P	H	H	H	H	
	8,8	P	P	P	P	P	P	P	P	P	P	
	9,9	P	P	P	P	P	S	S	P	S	S	
	10,10	S	S	S	S	S	S	S	S	S	S	
	A,2	H	H	H	D	D	H	H	H	H	H	
	A,3	H	H	H	D	D	H	H	H	H	H	
	A,4	H	H	D	D	D	H	H	H	H	H	
	A,5	H	H	D	D	D	H	H	H	H	H	
	A,6	H	D	D	D	D	H	H	H	H	H	
	A,7	S	D	D	D	D	H	H	H	H	H	
	A,8	S	S	S	S	S	S	S	S	S	S	
	A,9	S	S	S	S	S	S	S	S	S	S	
	A,A	P	P	P	P	P	P	P	P	P	P	

H

 Hit

S

 Stand

D

 Double Down

P

 Split





Figure 1. Example of how strategy card will ultimately end looking like. For each dealer and player hand scenario, most statistically ideal player move is identified.

4.0 Desktop App

4.1 Overview

The desktop app will be executed through the console application. The console application is written in Java and can be ran on any desktop computer with the Java SDK that can be downloaded from the Oracle website. The console application does not use the database and it is purely Java code. However, it does communicate with an array of values that acts as the optimal strategy table for the Blackjack game. This strategy table is used to show the player what the optimal choice would be for each play they go through, regardless of whether their own playing choice is optimal or not.

This strategy table that is integrated into the game to showcase the helper text was generated from previously completed programs that involved creating a database of hands to produce the most optimal player decision. However, to clarify once again, there is no active database used in the console/desktop application. It consists of a Java file that uses the strategy table generated from past programs that involved databases to help the player make the most optimal decision in order to win at Blackjack.



Ultimately, the console app is a basic Java program showing record I/O and utilizing numerous classes as a part of Java. The project bargains a blackjack hand, and checks the player's choices (split, double down, hit, stand) against the right procedure in the strategy card. The strategy card is laid out in the plain text file which ought to live in the same index as the project.

The system doe keep track of who's winning in its present structure as well as checking the player's data (split, double down, hit, stand) against the right choice from the methodology card. At the point when the player parts a double, one and only of the split hands is played. Type in 'q' at the command line to leave the game.

The strategy card text file can be altered to change what the "right" choice is for a given hand.

The system comprises of three java classes, Deck.java, Hand.java, and Blackjack.java.

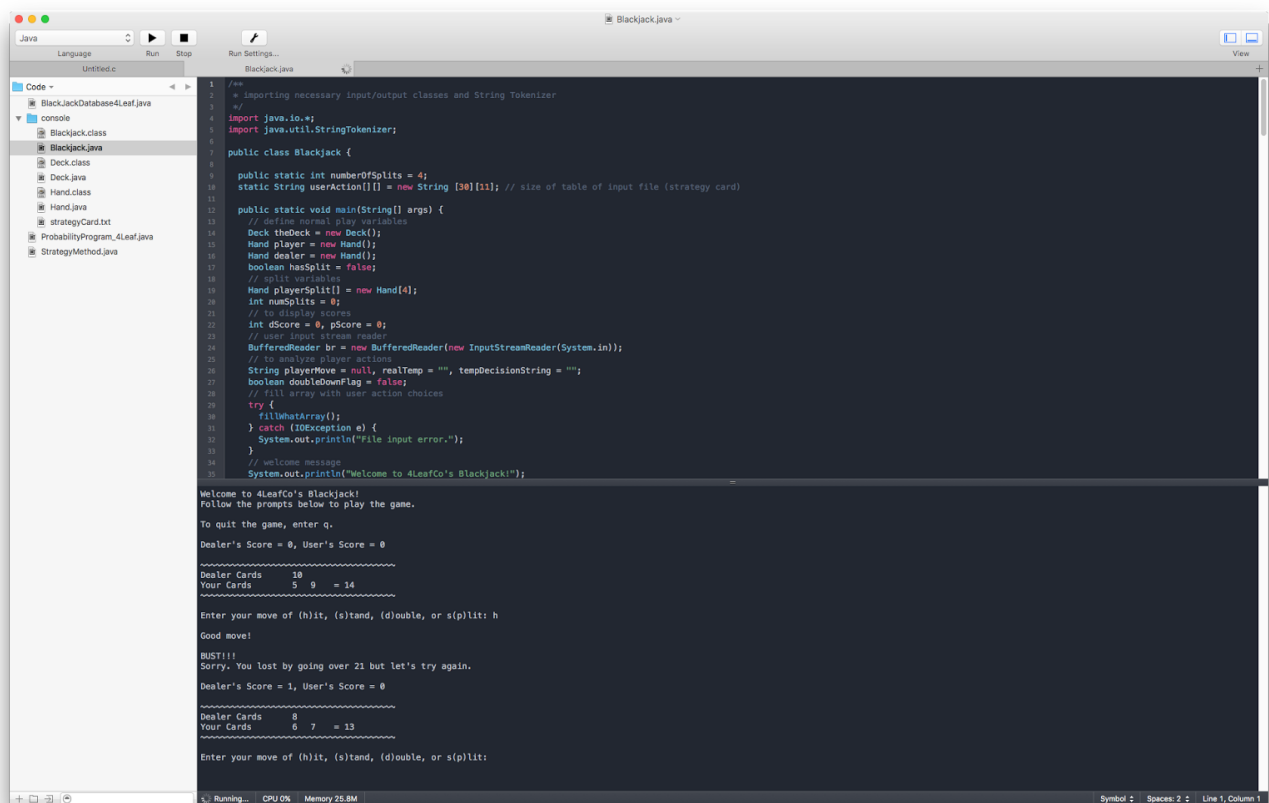
Deck.java is a 52 card deck. The deal() function gives back a string comprising of a numeric worth, or J, Q, K, or A, speaking to an irregular card. At the point when a card is managed, the deck denotes that card as lost. The deck won't reshuffle consequently, and will fail if more than 52 cards are managed. The shuffle() function reshuffles the deck.

Hand.java speaks to a man's hand (either the player or the merchant). The hit() capacity takes a string comprising of a numeric quality, or J, Q, K, or An, and adds it to it's variety of cards. The other functions return different aspects of the a hand.

Blackjack.java is the place the main() loop is located. The only real member variable it has consists of the array that holds the correct choices depending on the hand situation. To make the code DRY (non-repetitive), various methods were created that could be called for any hand scenario.

This system can be incorporated on any stage since it is composed simply in Java.

Refer to the figures below for an idea of how the console app looks like when run.



The screenshot shows a Java IDE with the file 'Blackjack.java' open. The code defines a 'Blackjack' class with a 'main' method. The console output shows the game's execution, including a welcome message, initial scores, and a game round where the user busts.

```
1  /*  
2  * Importing necessary input/output classes and String Tokenizer  
3  */  
4  import java.io.*;  
5  import java.util.StringTokenizer;  
6  
7  public class Blackjack {  
8  
9      public static int numberOfSplits = 4;  
10     static String userAction[] = new String [30][11]; // size of table of input file (strategy card)  
11  
12     public static void main(String[] args) {  
13         // define normal play variables  
14         Deck theDeck = new Deck();  
15         Hand player = new Hand();  
16         Hand dealer = new Hand();  
17         boolean hasSplit = false;  
18         // split variables  
19         Hand playerSplit[] = new Hand[4];  
20         int numSplits = 0;  
21         // to display scores  
22         int dScore = 0, pScore = 0;  
23         // user agent stream reader  
24         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));  
25         // to analyze player actions  
26         String playerMove = null, realTemp = "", tempDecisionString = "";  
27         boolean doubleDownFlag = false;  
28         // fill array with user action choices  
29         try {  
30             fillUserArray();  
31         } catch (IOException e) {  
32             System.out.println("File input error.");  
33         }  
34         // welcome message  
35         System.out.println("Welcome to 4LeafCo's Blackjack!");  
36  
37         Welcome to 4LeafCo's Blackjack!  
38         Follow the prompts below to play the game.  
39  
40         To quit the game, enter q.  
41  
42         Dealer's Score = 0, User's Score = 0  
43  
44         ~~~~~  
45         Dealer Cards      10  
46         Your Cards        5 9 = 14  
47         ~~~~~  
48  
49         Enter your move of (h)it, (s)and, (d)ouble, or s(p)lit: h  
50  
51         Good move!  
52  
53         BUST!!!  
54         Sorry. You lost by going over 21 but let's try again.  
55  
56         Dealer's Score = 1, User's Score = 0  
57  
58         ~~~~~  
59         Dealer Cards      8  
60         Your Cards        6 7 = 13  
61         ~~~~~  
62  
63         Enter your move of (h)it, (s)and, (d)ouble, or s(p)lit:
```

Figure 2. Running Console App in CodeRunner

	2	3	4	5	6	7	8	9	10	A
8	H	H	H	H	H	H	H	H	H	H
9	H	H	H	H	S	H	H	H	H	H
10	H	D	H	H	H	H	H	S	H	H
11	H	D	S	H	S	H	D	S	H	D
12	S	D	H	H	H	H	H	H	H	H
13	D	D	H	S	H	S	H	H	H	H
14	D	H	H	H	H	S	H	H	S	H
15	H	H	H	H	H	H	H	H	S	H
16	S	H	H	H	H	S	H	H	H	D
17	S	H	H	H	S	S	S	S	H	S
2,2	S	S	H	H	S	H	H	H	D	H
3,3	H	S	H	H	H	S	H	H	H	H
4,4	S	H	S	H	H	S	H	D	S	H
5,5	S	H	H	S	D	S	H	H	H	H
6,6	H	H	H	H	H	H	H	H	H	H
7,7	S	S	H	S	S	H	H	S	S	S
8,8	H	P	H	P	S	P	P	H	P	H
9,9	S	H	S	H	S	H	S	H	H	H
10,10	S	H	S	S	H	S	S	S	S	S
A,2	D	S	S	H	S	H	H	H	H	H
A,3	H	H	H	S	H	H	D	H	H	H
A,4	H	D	H	H	H	H	H	S	S	H
A,5	H	H	H	H	H	H	H	H	H	H
A,6	S	H	H	H	H	H	H	H	H	H
A,7	H	H	H	H	H	H	H	H	S	H
A,8	S	S	S	S	H	S	S	S	S	H
A,9	S	S	S	S	H	H	H	S	S	S
A,10	S	H	H	S	S	S	S	S	H	S
A,A	S	H	P	P	P	P	D	P	P	H

Figure 3. Format for Strategy Card

5.0 Visual Interface

5.1 Overview

The visual interface for the desktop app is limited to the console. This means the visual interface is limited to all the text prompts for the following indications:

- Intro prompt to game and how to play


- i.e. "Welcome to the game of 4Leaf Co's Blackjack. You will now be asked how many points you'd like to bet before you can start a game."
- Cards the player has at hand and the sum of the cards
 - i.e. "Your cards are [player cards]. Your total is [total here]."
- Card the dealer has shown up
 - i.e. "Dealer is showing the [dealer card showing up]."
- Asking which choice the player wants to take
 - i.e. "Enter P to split, D to double down, H to hit, or S to stand."
- Making sure player enters a legitimate option if they don't enter a specified letter
 - i.e. "Please make a choice by entering P (to split), D (to double down), H (to hit), or S (to stand)."
- Confirmation of the choice they decided to play
 - i.e. "User hits."
- Whether or not the option they choose (Hit, Stand, Double Down, Split) was the most optimal solution after they make their choice
 - i.e. "You should have chosen to [hit, stand, double down, split] instead of [player choice]."
- Result of user's play
 - i.e. "Your card is the King. Now your cards are 4, 2, King. Your total is now 16."
- Results of the dealer's play and total (not initial)

- i.e. "Dealer's cards are 6 and 8. Dealer's total is 14."
- Dealer's play
 - i.e. "Dealer hits and gets the Jack"
- How you won/lost
 - i.e. "Dealer busted by going over 21. You win."
- Your number of wins
 - i.e. "Wins = 1"
- Number of points you have to bet
 - i.e. "You have 2 points."
- Prompt once player/dealer wins
 - i.e. "You won!"

6.0 Tracking and Control Mechanisms

6.1 Document Revision History

Version	Implemented By	Approved By	Date	Reason
1.0	Calvin Chambers Frances Coronel	Jordan Quick	1/21/16	Iteration 1
2.0	Frances Coronel	Jordan Quick	2/11/16	Iteration 2
3.0	Frances Coronel	Jordan Quick	3/3/16	Iteration 3



4.0	Frances Coronel	Jordan Quick	3/31/16	Iteration 4
5.0	Frances Coronel	Jordan Quick	4/26/16	Iteration 5