Najee Walls

CS 306 – Data Structures

A demonstration of the ArrayList: Blackjack program.

A list is a data structure that is a collection of objects where the order of the objects is important. An array list is a list that is implemented using an array. So the List is the name of the data structure but the array is just the way that the list is working.

In this demonstration case the scenario is the card game Blackjack. In blackjack one or more players and a dealer draws cards with the goal of getting as close to 21 in total score without going over. The names of the classes will be capitalized

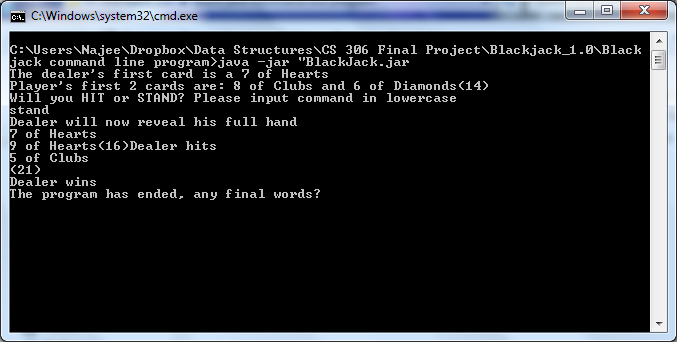
There is a DECK of CARDS that each PLAYER draws from. Each PLAYER checks the value of the CARD from the CARD’s rank. They add up this score and recite what cards they have before they decide to hit or stand. If a PLAYER hits they draw another card, if a PLAYER stands the dealer, which is also a PLAYER will automatically hit until they reach a score of over 17. At that point if their score is higher than the user’s while still being under 21, the dealer wins. In other cases the dealer either ties or loses to the user.

The DECK is a generated array list of CARDS. A CARD is composed of a suit and rank. The PLAYER also has an arraylist of CARDS called a hand that was pulled from the DECK. The PLAYER can also determine the value a CARD to a particular game.

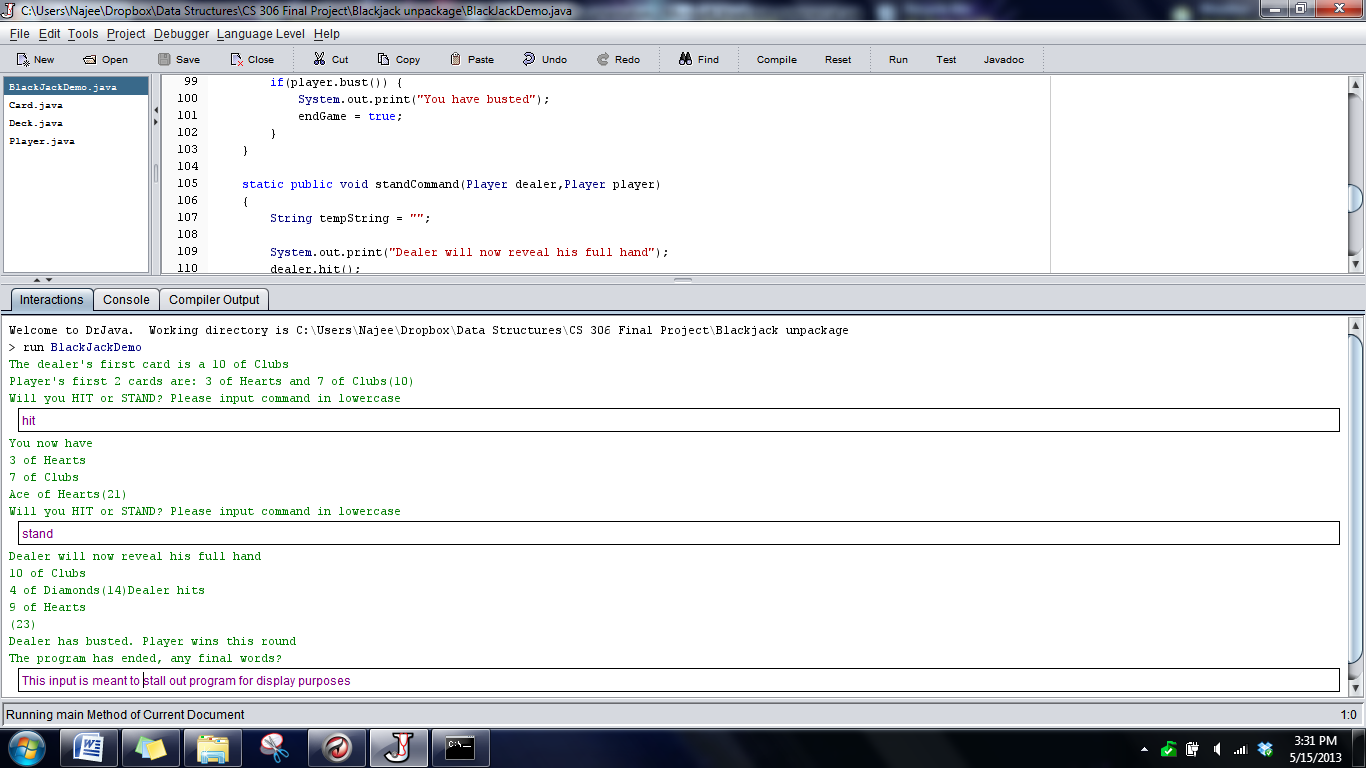
The program, as it is, is only concerned about generating a list and then removing items from a list.

There are other ways to implement a List such as a Linked List, the array is particularly efficient at accessing a particular element. Something like a “getEntry” method would be an efficient process for an ArrayList. However this is that this is not what the program is doing. It is removing the first card (index 0) from the deck and adding it to the player’s hands. Both of the arrays from player and deck are resized in the process. Removing an entry means re-ordering the indexes of entries. Adding an entry means copying the contents of the array to a larger array. Both of these things can be a time drain on larger sized arrays. For this kind of application, where items in a list are added to and removed constantly, a linked list or perhaps a stack list would have been a more efficient approach. That way the whole collection of items isn’t moved every time a player draws from the deck. This is to say, although it was done in this way because of my own preference, there is a better performing data structure for this scenario and I think that is worth noting.

Command Line Output



Dr Java Output



Class BlackJackDemo

/\*\*

\*

\* **@author** nawalls

\*/

**import** java**.**io**.**Console**;**

**import** java**.**util**.**Scanner**;**

**import** java**.**util**.**ArrayList**;**

public class BlackJackDemo **{**

private static Deck bjDeck**;**

private static boolean endGame **=** **false;**

/\*\*

\* **@param** args the command line arguments

\*/

public static void main**(**String**[]** args**)** **{**

/\*Program Testing Area -----------------------------------

Deck myDeck = new Deck();

myDeck.generateDeck(1);

myDeck.shuffle();

String hand = "";

/\*

for(int i = 0; i < myDeck.getSizeDeck(); i++)

System.out.println(myDeck.cards.get(i));

\*/

/\*

String s = myDeck.cardList();

System.out.println(s);

for(int i = 0; i < 5; i++)

{

hand += myDeck.draw().cardString() + "\n";

}

System.out.println(hand);

Program Testing Area End --------------------------------- \*/

//Instance and Generate Cards for deck

bjDeck **=** **new** Deck**();**

bjDeck**.**generateDeck**(**1**);**

//Randomize Deck Order

bjDeck**.**shuffle**();**

//player and gambler starts with deck.

Player player1 **=** **new** Player**(**bjDeck**);**

Player dealer **=** **new** Player**(**bjDeck**);**

//Dealer reveals first card

dealer**.**hit**();**

System**.**out**.**println**(**"The dealer's first card is a " **+** dealer**.**hand**.**get**(**0**).**cardString**());**

//Player gets intial 2 cards

player1**.**hit**();**

player1**.**hit**();**

System**.**out**.**println**(**"Player's first 2 cards are: " **+** player1**.**hand**.**get**(**0**).**cardString**()**

**+** " and " **+** player1**.**hand**.**get**(**1**).**cardString**()+** "("**+** player1**.**handValue**()** **+** ")" **);**

//Request for user's input

Scanner command **=** **new** Scanner**(**System**.**in**);**

**while(!**endGame**)**

**{**

System**.**out**.**println**(**"Will you HIT or STAND? Please input command in lowercase"**);**

String teststring **=** command**.**next**();**

**if(** "hit"**.**equals**(**teststring**))**

**{**

hitCommand**(**player1**);**

**}**

**else** **if** **(**"stand"**.**equals**(**teststring**))**

**{**

standCommand**(**dealer**,**player1**);**

**}**

**}**

System**.**out**.**println**(**"The program has ended, any final words?"**);**

command**.**next**();**

**}**

static public void hitCommand**(**Player player**)**

**{**

player**.**hit**();**

System**.**out**.**println**(**"You now have " **+** player**.**printCardName**()** **+** "(" **+** player**.**handValue**()** **+**")" **);**

**if(**player**.**bust**())** **{**

System**.**out**.**print**(**"You have busted"**);**

endGame **=** **true;**

**}**

**}**

static public void standCommand**(**Player dealer**,**Player player**)**

**{**

String tempString **=** ""**;**

System**.**out**.**print**(**"Dealer will now reveal his full hand"**);**

dealer**.**hit**();**

System**.**out**.**print**(**dealer**.**printCardName**()** **+** "(" **+** dealer**.**handValue**()+**")"**);**

//dealer must always hit until the value of it's cards reaches 17

**while** **(**dealer**.**handValue**()** **<** 17**)**

**{**

dealer**.**hit**();**

System**.**out**.**println**(**"Dealer hits"**);**

//Add the last card to tempString for display

tempString **+=** dealer**.**hand**.**get**(**dealer**.**hand**.**size**()** **-** 1**).**cardString**()** **+** "\n"**;**

System**.**out**.**println**(**tempString **+** "(" **+** dealer**.**handValue**()+**")"**);**

**}**

//If the dealer busts.

**if** **(** dealer**.**bust**())** **{**

System**.**out**.**println**(**"Dealer has busted. Player wins this round"**);**

endGame **=** **true;**

**}**

//If there's a tie.

**else** **if(**dealer**.**handValue**()** **==** player**.**handValue**())**

**{**

System**.**out**.**println**(**"A tie"**);**

endGame **=** **true;**

**}**

//If the player has a higher hand

**else** **if** **(**dealer**.**handValue**()** **<** player**.**handValue**())**

**{**

System**.**out**.**println**(**"Player wins"**);**

endGame **=** **true;**

**}**

//The final condition; dealer has a higher hand

**else**

**{**

System**.**out**.**println**(**"Dealer wins"**);**

endGame **=** **true;**

**}**

**}**

**}**

Class Card

/\*\*

\*

\* **@author** nawalls

\*/

public class Card **{**

int rank**;**

int suit**;**

private static String **[]** suits **=** **{**"Spades"**,**"Diamonds"**,**"Hearts"**,**"Clubs"**};**

private static String**[]** ranks **=** **{**"Ace"**,**"2"**,**"3"**,**"4"**,**"5"**,**"6"**,**"7"

**,**"8"**,**"9"**,**"10"**,**"Jack"**,**"Queen"**,**"King"**};**

public Card**()**

**{**

**}**

public Card**(**int suit**,** int rank**)**

**{**

**this.**rank **=** rank**;**

**this.**suit **=** suit**;**

**}**

public String cardString**()**

**{**

**return** ranks**[**rank**]** **+** " of " **+** suits**[**suit**];**

**}**

public int getRank**()**

**{**

**return** rank**;**

**}**

public int getSuit**()**

**{**

**return** suit**;**

**}**

**}**

Class Deck

**import** java**.**util**.**ArrayList**;**

**import** java**.**util**.**Random**;**

/\*\*

\* Reference

\* http://www.dreamincode.net/forums/topic/110380-deck-of-cards-using-various-methods/

\* This was a great help

\*

\* **@author** nawalls

\*/

public class Deck **{**

private ArrayList**<**Card**>** cards**;**

Deck**()** **{**

**}**

public void generateDeck**(**int n**)** **{**

cards **=** **new** ArrayList**<**Card**>();**

//Complexity for Deck generation is (52n)

**{**

**for(**int i **=** 0**;** i **<** n**;** i**++){**

**for** **(**int s **=** 0**;** s **<=** 3**;** s**++)** **{**

**for** **(**int r **=** 0**;** r **<=** 12**;** r**++)** **{**

cards**.**add**(new** Card**(**s**,** r**));**

**}**

**}**

**}**

**}**

**}**

//Returns each string in the deck each on it's own line

public String cardList**()**

**{**

String cardString **=** ""**;**

Card tempCard**;**

**for** **(**int i **=** 0**;** i **<** cards**.**size**();** i**++)**

**{**

tempCard **=** cards**.**get**(**i**);**

cardString **+=** tempCard**.**cardString**()** **+** "\n"**;**

**}**

**return** cardString**;**

**}**

public Integer getSizeDeck**()** **{**

**return** cards**.**size**();**

**}**

public Card draw**()** **{**

//Remove card from top of deck

//Return the name of the card

**return** cards**.**remove**(**0**);**

**}**

public void shuffle**(){**

int index1**;**

int index2**;**

Card tempCard**;**

Random x **=** **new** Random**();**

//swaps the places of two indexes 100 times.

**for(**int i**=**0**;** i**<**200**;** i**++)**

**{**

//get indexs of two cards

index1 **=** x**.**nextInt**(**cards**.**size**()** **-** 1**);**

index2 **=** x**.**nextInt**(**cards**.**size**()** **-** 1**);**

//Card1 moves to tempcard

tempCard **=** cards**.**get**(**index2**);**

//Card2 moves to Card1

cards**.**set**(**index2**,** cards**.**get**(**index1**));**

//Tempcard(which was Card1) moves to Card2

cards**.**set**(**index1**,** tempCard**);**

**}**

**}**

**}**

Class Player

**import** java**.**util**.**ArrayList**;**

**import** java**.**util**.**Iterator**;**

/\*\*

\*

\* **@author** Najee The player class should be able to: -Check the value of cards

\* -Add them -Check for busts

\*/

public class Player **{**

Deck bjDeck**;**

ArrayList**<**Card**>** hand**;**

Card card**;**

Integer handValue**;**

public Player**(**Deck deck**)** **{**

hand **=** **new** ArrayList**();**

bjDeck **=** deck**;**

handValue **=** 0**;**

**}**

//gets the value of the incoming cards for this game of blackJack

public Integer getValue**(**Card tempcard**)** **{**

/\*

//If the card is an Ace and it would cause a bust,

//Change the value to 1;

if (tempcard.rank == 0 & currentValue > 10) {

return 1;

}

\*/

**switch** **(**tempcard**.**getRank**())** **{**

**default:**

**return** 0**;**

**case** 0**:**

**return** 11**;**

**case** 1**:**

**return** 2**;**

**case** 2**:**

**return** 3**;**

**case** 3**:**

**return** 4**;**

**case** 4**:**

**return** 5**;**

**case** 5**:**

**return** 6**;**

**case** 6**:**

**return** 7**;**

**case** 7**:**

**return** 8**;**

**case** 8**:**

**return** 9**;**

**case** 9**:**

**return** 10**;**

**case** 10**:**

**return** 10**;**

**case** 11**:**

**return** 10**;**

**case** 12**:**

**return** 10**;**

//ace transfer case

**case** 13**:**

**return** 1**;**

**}**

**}**

public void hit**()** **{**

//Add a card from the deck to the

hand**.**add**(**bjDeck**.**draw**());**

handValue **=** handValue**();**

**}**

public boolean bust**()** **{**

**if** **(**handValue **>** 21**)** **{**

**return** **true;**

/\*Foe each card in the hand

for (int i = 0; i < hand.size(); i++) {

if (hand.get(i).rank == 0)

//change it to a value of 1 (as seen from 14)

hand.get(i).rank = 14;

}\*/

**}** **else** **{**

**return** **false;**

**}**

**}**

public Integer handValue**()** **{**

int tempValue **=** 0**;**

int realValue **=** 0**;**

//If the value is under 21, even with 11-Ace, just add the hand up

**for** **(**int i **=** 0**;** i **<** hand**.**size**();** i**++)** **{**

tempValue **+=** getValue**(**hand**.**get**(**i**));**

**}**

//Subtract 10 for each ace in the hand if the value is over 21

**if** **(**tempValue **>** 21**)** **{**

**for** **(**int i **=** 0**;** i **<** hand**.**size**();** i**++)** **{**

**if** **(**hand**.**get**(**i**).**rank **==** 0**)** **{**

tempValue **=** tempValue **-** 10**;**

**}**

**}**

**}**

//If after all of this it's over 21, it's busted.

**return** tempValue**;**

**}**

public String printCardName**()** **{**

String printString **=** ""**;**

//Attempted Iterator loops.Need to fine tune to get strings out of class Card

/\*

Iterator itr = hand.iterator();

while(itr.hasNext()){

printString += itr.next() + "\n";

\*/

**for** **(**int i **=** 0**;** i **<** hand**.**size**();** i**++)** **{**

printString **+=** "\n" **+** hand**.**get**(**i**).**cardString**();**

**}**

**return** printString**;**

**}**

**}**