

Code Review Stack Exchange is a question and answer site for peer programmer code reviews. Join them; it only takes a minute:

Here's how it works:

Sign up

Anybody can ask
a question

Anybody can
answer

The best answers are
voted up and rise to the
top

First attempt at a Java Blackjack game

I just completed my first multi class program, Blackjack, and it works! It allows the user to play Blackjack against a single dealer, with no other players at the table. I was wondering, seeing as it is my first multi class program, if you could help me optimize it and/or give me any advice. I want to implement insurance and splitting, so any advice to help prepare the code for eventually adding those features would be really helpful! Finally, my main method is pretty long — I was wondering if this is typical of Java programs and, if not, how I can fix that.

Card

```
package Blackjack;

class Card {
    /*
     * Creates a playing card.
     */
    private int rank;//represents the rank of a card
    private int suit;//represents the suit of a card
    private int value;//represents the value of a card
    private static String[] ranks =
    {"Joker", "Ace", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine", "Ten", "Jack", "Queen"}

    private static String[] suits = {"Clubs", "Diamonds", "Hearts", "Spades"};
    /*
     * Created with an integer that represents a spot in the String array ranks and the String
     array suits. This represents
     * the rank and suit of an individual card.
     */
    Card(int suit, int values)
    {
        this.rank=values;
        this.suit=suit;
    }
    /*
     * Returns the string version of a card.
     */
    public String toString()
    {
        return ranks[rank]+" of "+suits[suit];
    }
    /*
     * Returns the rank of a card.
     */
    public int getRank()
    {
        return rank;
    }
    /*
     * Returns the suit of a card.
     */
    public int getSuit()
    {
        return suit;
    }
    /*
     * Returns the value of a card. If a jack, queen, or king the value is ten. Aces are 11
     for now.
     */
    public int getValue()
    {
        if(rank>10)
        {
            value=10;
        }
        else if(rank==1)
        {
            value=11;
        }
        else
        {
            value=rank;
        }
        return value;
    }
}
```

```

    * Sets the value of a card.
    */
    public void setValue(int set)
    {
        value = set;
    }
}

```

Deck

```

package Blackjack;
import java.util.ArrayList;
import java.util.Random;
/*
 * Creates and shuffles a deck of 52 playing cards.
 */
class Deck {
    private ArrayList<Card> deck;//represents a deck of cards
    Deck()
    {
        deck = new ArrayList<Card>();
        for(int i=0; i<4; i++)
        {
            for(int j=1; j<=13; j++)
            {
                deck.add(new Card(i,j));
            }
        }
    }
}
/*
 * Shuffles the deck by changing the indexes of 200 random pairs of cards in the deck.
 */
public void shuffle()
{
    Random random = new Random();
    Card temp;
    for(int i=0; i<200; i++)
    {
        int index1 = random.nextInt(deck.size()-1);
        int index2 = random.nextInt(deck.size()-1);
        temp = deck.get(index2);
        deck.set(index2, deck.get(index1));
        deck.set(index1, temp);
    }
}
/*
 * Draws a card from the deck.
 */
public Card drawCard()
{
    return deck.remove(0);
}
}

```

Dealer

```

package Blackjack;
import java.util.ArrayList;
import java.util.Arrays;
/*
 * Creates a dealer that the user plays against.
 */
class Dealer {
    ArrayList<Card> hand;//represents the dealer's hand
    private int handvalue=0;//value of the dealer's hand (starts at 0)
    private Card[] aHand;//used to convert the dealer's hand to an array
    private int AceCounter;//counts the aces in the dealer's hand
    Dealer(Deck deck)
    {
        hand = new ArrayList<>();
        aHand = new Card[]{};
        int AceCounter=0;
        for(int i=0; i<2; i++)
        {
            hand.add(deck.drawCard());
        }
        aHand = hand.toArray(aHand);
        for(int i=0; i<aHand.length; i++)
        {
            handvalue += aHand[i].getValue();
            if(aHand[i].getValue()==11)
            {
                AceCounter++;
            }
            while(AceCounter>0 && handvalue>21)
            {
                handvalue-=10;
                AceCounter--;
            }
        }
    }
}
/*
 * Prints the dealer's first card (the card face up at the beginning of a blackjack game).
 */
public void showFirstCard()
{
    Card[] firstCard = new Card[]{};
    firstCard = hand.toArray(firstCard);
    System.out.println("[ "+firstCard[0]+" ]");
}
}
/*
 * Gives the dealer another card and updates the value of his hand. Takes into account the
 * value of aces.
 */

```

```

public void Hit(Deck deck)
{
    hand.add(deck.drawCard());
    aHand = hand.toArray(aHand);
    handvalue = 0;
    for(int i=0; i<aHand.length; i++)
    {
        handvalue += aHand[i].getValue();
        if(aHand[i].getValue()==11)
        {
            AceCounter++;
        }
        while(AceCounter>0 && handvalue>21)
        {
            handvalue-=10;
            AceCounter--;
        }
    }
}
/*
 * Determines if the dealer wants to hit according to classic Blackjack rules.
 */
public boolean wantsToHit()
{
    if(handvalue<17)
    {
        return true;
    }
    return false;
}
/*
 * Returns true if the dealer has blackjack.
 */
public boolean hasBlackJack()
{
    if(hand.size()==2 && handvalue==21)
    {
        System.out.println("The dealer has blackjack!");
        return true;
    }
    return false;
}
/*
 * Prints the dealer's hand.
 */
public void showHand()
{
    System.out.println(hand);
}
/*
 * Returns the value of the dealer's hand.
 */
public int getHandValue()
{
    return handvalue;
}
/*
 * Determines if a dealer has busted.
 */
public boolean busted(int handvalue)
{
    if(handvalue>21)
    {
        System.out.println("The dealer busted!");
        return true;
    }
    return false;
}
/*
 * Takes the turn for the dealer and returns the value of his hand.
 */
public int takeTurn(Deck deck)
{
    while(wantsToHit())
    {
        System.out.println("The dealer hits");
        Hit(deck);
        if(busted(handvalue))
        {
            break;
        }
    }
    if(handvalue<=21)
    {
        System.out.print("The dealer stands.");
    }
    return handvalue;
}

```

Main

```

package Blackjack;
import java.util.*;
public class Blackjack {
    private static int cash;//cash the user bets with
    private static int bet;//how much the user wants to bet
    private static int AceCounter;//how many aces are in the user's hand
    private static ArrayList<Card> hand;//represents the user's hand
    private static int handvalue;//the value of the user's hand
    private static String name;//name of the user
    public static void main(String[] args){
        System.out.println("Hi! What is your name?");
        Scanner scan = new Scanner(System.in);
    }
}

```

```

name = scan.nextLine();
System.out.println("Hello, "+name+", lets play some Blackjack!");
System.out.println("How much cash do you want to start with?");
Scanner money = new Scanner(System.in);
cash = money.nextInt();
System.out.println("You start with cash: "+cash);
while(cash>0){
    Deck deck = new Deck();//initialize deck, dealer, hands, and set the bet.
    deck.shuffle();
    AceCounter=0;
    Dealer dealer = new Dealer(deck);
    List<Card> hand = new ArrayList<>();
    hand.add(deck.drawCard());
    hand.add(deck.drawCard());
    System.out.println("How much would you like to bet?");
    bet=Bet(cash);
    System.out.println("Cash:"+cash-bet);
    System.out.println("Money on the table:"+bet);
    System.out.println("Here is your hand: ");
    System.out.println(hand);
    int handvalue = calcHandValue(hand);
    System.out.println("The dealer is showing: ");
    dealer.showFirstCard();
    if(hasBlackJack(handvalue) && dealer.hasBlackJack())//check if both the user and
dealer have blackjack.
    {
        Push();
    }
    else if(hasBlackJack(handvalue))//check if the user has blackjack.
    {
        System.out.println("You have BlackJack!");
        System.out.println("You win 2x your money back!");
        cash=cash+bet;
        Win();
    }
    else if(dealer.hasBlackJack())//check if the dealer has blackjack.
    {
        System.out.println("Here is the dealer's hand:");
        dealer.showHand();
        Lose();
    }
    else
    {
        if(2*bet<cash)//check if the user can double down.
        {
            System.out.println("Would you like to double down?");//allows the user to
double down.
            Scanner doubledown = new Scanner(System.in);
            String doubled = doubledown.nextLine();
            while(!isyesorno(doubled))
            {
                System.out.println("Please enter yes or no.");
                doubled = doubledown.nextLine();
            }
            if(doubled.equals("yes"))
            {
                System.out.println("You have opted to double down!");
                bet=2*bet;
                System.out.println("Cash:"+cash-bet);
                System.out.println("Money on the table:"+bet);
            }
        }
        System.out.println("Would you like to hit or stand?");//ask if the user will
hit or stand
        Scanner hitorstand = new Scanner(System.in);
        String hitter = hitorstand.nextLine();
        while(!isHitorStand(hitter))
        {
            System.out.println("Please enter 'hit' or 'stand'.");
            hitter = hitorstand.nextLine();
        }
        while(hitter.equals("hit"))//hits the user as many times as he or she pleases.
        {
            Hit(deck, hand);
            System.out.println("Your hand is now:");
            System.out.println(hand);
            handvalue = calcHandValue(hand);
            if(checkBust(handvalue))//checks if the user busted
            {
                Lose();
                break;
            }
            if(handvalue<=21 && hand.size()==5)//checks for a five card trick.
            {
                fivecardtrick();
                break;
            }
            System.out.println("Would you like to hit or stand?");
            hitter = hitorstand.nextLine();
        }
        if(hitter.equals("stand"))//lets the user stand.
        {
            int dealerhand = dealer.takeTurn(deck);//takes the turn for the dealer.
            System.out.println("");
            System.out.println("Here is the dealer's hand:");
            dealer.showHand();
            if(dealerhand>21)//if the dealer busted, user wins.
            {
                Win();
            }
            else
            {
                int you = 21-handvalue;//check who is closer to 21 and determine
winner

```

```

        int deal = 21-dealerhand;
        if(you==deal)
        {
            Push();
        }
        if(you<deal)
        {
            Win();
        }
        if(deal<you)
        {
            Lose();
        }
    }
}

System.out.println("Would you like to play again?");//ask if the user wants to keep
going
Scanner yesorno = new Scanner(System.in);
String answer = yesorno.nextLine();
while(!isyesorno(answer))
{
    System.out.println("Please answer yes or no.");
    answer = yesorno.nextLine();
}
if(answer.equals("no"))
{
    break;
}
}

System.out.println("Your cash is: "+cash);//if user doesn't want to play or runs out
of cash, either congratulates them on their winnings or Lets them know
if(cash==0)
{
    System.out.println("You ran out of cash!");
}
else
{
    System.out.println("Enjoy your winnings, "+name+"!");
}
}
}
/*
 * Checks if the user has blackjack.
 */
public static boolean hasBlackJack(int handValue)
{
    if(handValue==21)
    {
        return true;
    }
    return false;
}
/*
 * Calculates the value of a player's hand.
 */
public static int calcHandValue(List<Card> hand)
{
    Card[] aHand = new Card[]{};
    aHand = hand.toArray(aHand);
    int handvalue=0;
    for(int i=0; i<aHand.length; i++)
    {
        handvalue += aHand[i].getValue();
        if(aHand[i].getValue()==11)
        {
            AceCounter++;
        }
        while(AceCounter>0 && handvalue>21)
        {
            handvalue-=10;
            AceCounter--;
        }
    }
    return handvalue;
}
/*
 * Asks the user how much he or she would like to bet.
 */
public static int Bet(int cash)
{
    Scanner sc=new Scanner(System.in);
    int bet=sc.nextInt();
    while(bet>cash)
    {
        System.out.println("You cannot bet more cash than you have!");
        System.out.println("How much would you like to bet?");
        bet=sc.nextInt();
    }
    return bet;
}
/*
 * Called if the user wins.
 */
public static void Win()
{
    System.out.println("Congratulations, you win!");
    cash=cash+bet;
    System.out.println("Cash: "+cash);
}
/*
 * Called if the user loses.
 */
public static void Lose()
{

```

```

        System.out.println("Sorry, you lose!");
        cash=cash-bet;
        System.out.println("Cash: "+cash);
    }
    /*
     * Called if the user pushes
     */
    public static void Push()
    {
        System.out.println("It's a push!");
        System.out.println("You get your money back.");
        System.out.println("Cash: "+cash);
    }
    /*
     * Adds a card to user's hand and calculates the value of that hand. Aces are taken into
     account.
     */
    public static void Hit(Deck deck, List<Card> hand)
    {
        hand.add(deck.drawCard());
        Card[] aHand = new Card[]{};
        aHand = hand.toArray(aHand);
        handvalue = 0;
        for(int i=0; i<aHand.length; i++)
        {
            handvalue += aHand[i].getValue();
            if(aHand[i].getValue()==11)
            {
                AceCounter++;
            }
            while(AceCounter>0 && handvalue>21)
            {
                handvalue-=10;
                AceCounter--;
            }
        }
    }
    /*
     * Determines if a user has input hit or stand.
     */
    public static boolean isHitorStand(String hitter)
    {
        if(hitter.equals("hit") || hitter.equals("stand"))
        {
            return true;
        }
        return false;
    }
    /*
     * Determines if a user has busted.
     */
    public static boolean checkBust(int handvalue)
    {
        if(handvalue>21)
        {
            System.out.println("You have busted!");
            return true;
        }
        return false;
    }
    /*
     * Determines if a user has input yes or no.
     */
    public static boolean isyesorno(String answer)
    {
        if(answer.equals("yes") || answer.equals("no"))
        {
            return true;
        }
        return false;
    }
    /*
     * Called if the user has a five card trick.
     */
    public static void fivecardtrick()
    {
        System.out.println("You have achieved a five card trick!");
        Win();
    }
}

```

java beginner object-oriented playing-cards

edited Jun 3 '15 at 19:15



200_success ♦

112k 14 128 366

asked Jun 3 '15 at 18:51



Barney Stinson

411 1 3

Follow-up question – 200_success ♦ Jun 4 '15 at 22:40

3 Answers

Class Structure

- A `Hand` class might be useful. It can calculate and store the hand value. This would also avoid the duplication you currently have (`calcHandValue` and `Hit`).

- Your `Dealer` class contains a lot of code that I would not place there. It contains the dealer AI (when does the dealer hit?), winning/losing condition check, printing, and counting. With a `Hand` class, you would already separate out some of it. I would also remove all the prints (they make code reuse difficult, and lead to bad code structure), and separate the AI logic to it's own class (this would make it easier to change the rules, because they are all in one place).
- Your `Blackjack` class also does way too much. It prints, it reads input, it handles input, it checks winning/losing condition (again, a duplication, see next point), etc. It is the player as well as the game, which violates the single responsibility principle.
- Whenever you copy/paste code, try to think of a better alternative. In this case, your `Dealer` and your `Blackjack` class contain a lot of duplication. Mainly because they both represent a blackjack player (the dealer and the player). A generic `Player` class might be helpful, from which `Dealer` and `HumanPlayer` extend.

So to summarize: I would add at a minimum a `Hand`, `Player` and `HumanPlayer` class. Possibly also an `Input / Output` interface and `ConsoleInput / ConsoleOutput` class (which would make it easy to add a GUI). There are more classes you could create, but this would be a good start.

Misc

- your whole `shuffle` function can be replaced by `Collections.shuffle(deck);` .
- Why does your `Dealer` class have `hand` and `aHand` ? This seems unnecessary and confusing.
- you have some style issues (position of curly bracket, indentation, spaces, etc). It seems mostly internally consistent (that's the important part), but does not really match what most Java programmers are used to. Most IDEs that support Java (eg Netbeans or Eclipse) format the code per default in a way most Java programmers recognize.
- variable and method names should start lowercase, so they are not confused with classes.
- `if (cond) return true else return false` can be written as `return cond` .

answered Jun 3 '15 at 19:53



tim

23.7k 2 21 73

okay i will try some of these things. thank you very much. I don't quite understand why I would add an Input/Output interface and what I would make it do. Do you mind explaining a little bit more about that point? – [Barney Stinson](#) Jun 3 '15 at 21:29

also, which class would you recommend I run the game in? – [Barney Stinson](#) Jun 3 '15 at 21:31

@Jared Input would just gather the action (which could be an enum; hit, split, etc), and output would output everything. The main reason is that it separates those things from the rest. It makes your code more readable, and especially more reusable. You can run your game in a `BlackJackGame` class, which should maintain the game loop (which would call other classes to deal a hand, get input, check end condition and apply results, after which it deals the next hand). All the rest should ideally happen elsewhere. – [tim](#) Jun 3 '15 at 21:48

if it isn't too much trouble, do you have any example code or any links to example code of a game or anything you would consider well structured? – [Barney Stinson](#) Jun 3 '15 at 22:23

Very nicely done, well deserved victory! +1 – [janos ♦](#) Jun 4 '15 at 4:51



Did you find this question interesting? Try our newsletter

Sign up for our newsletter and get our top new questions delivered to your inbox ([see an example](#)).

There's a quite a lot to improve on this. Here are a couple of tips to get your started.

Excessive comments

Do these comments add anything new that isn't clear already?

```
private int rank;//represents the rank of a card
private int suit;//represents the suit of a card
private int value;//represents the value of a card
```

They don't. In fact most of the other comments in the code don't add value either. The best code doesn't need comments. Look through all the comments in your code, if they are not needed, then remove them, if they are needed, then try to change the code in a way to not need comments.

Making `Card` immutable

Will it make sense for `rank`, `suit` and `value` to change in the lifetime of a `Card` instance? Probably not. So make these fields `final`. There is a `setValue` method, which you don't need either.

Review the other classes too. Make everything `final` that doesn't need to change or doesn't make sense to ever change. This practice can help you spot some design bugs.

Refer to types by interfaces

You declare several lists like this:

```
ArrayList<Card> hand;
```

Use the interface type instead:

```
List<Card> hand;
```

Consider `enums`

These variables are good candidates for `enum` `S`:

```
private static String[] ranks =
{"Joker", "Ace", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine", "Ten", "Jack", "Queen"}

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

Like this:

```
enum Suit {
    Clubs,
    Diamonds,
    Hearts,
    Spades
}
```

If you want to iterate over the possible suits, you can do `for (Suit suit : Suit.values()) { ... }`

Magic numbers


There are too many magic numbers in the code. 17, 10, 11, 4, ... It would be better to put these in `public static final` variables with descriptive names, to clarify the purpose of these values, have them together near the top of the code for easier control and flexibility to play with.

Formatting

The code doesn't follow the common formatting generated by the auto-format option of common IDEs like Eclipse and IntelliJ. I suggest to reformat the entire thing, to make the code look more familiar and easier to read for the majority of Java coders. In Eclipse the keyboard shortcut is Control-Shift-f

edited Jun 3 '15 at 19:53

answered Jun 3 '15 at 19:43

 **janos** ♦
86.2k 12 101 316

If i use auto format, is this the way programs are usually formatted? – [Barney Stinson](#) Jun 3 '15 at 19:45

Yes, that's the common way – [janos](#) ♦ Jun 3 '15 at 19:50

2 I'd add the the whole "value" attribute of Card is not needed, only the getValue() function is, which only needs the rank to work. – [C4stor](#) Jun 3 '15 at 22:58

Return the boolean directly

The following:

```
public static boolean isyesorno(String answer)
{
    if(answer.equals("yes") || answer.equals("no"))
    {
        return true;
    }
    return false;
}
```

should become:

```
public static boolean isyesorno(String answer) {
    return answer.equals("yes") || answer.equals("no");
}
```


The same goes for `public static boolean hasBlackJack(int handValue)` and `public static boolean isHitorStand(String hitter)` and `public static boolean checkBust(int handvalue)` for the latter you should move printing out of the function.

Use already existing wheels

You can shuffle the deck by using the built-in:

```
List<Cards> list = Arrays.asList(deck);
Collections.shuffle(list);
```

answered Jun 3 '15 at 19:37

 Caridorc

22k

3

29

94

-
- will shuffling this way shuffle my ArrayList deck, or give me a new list called list that is shuffled? – Barney Stinson Jun 4 '15 at 14:57
-
- @jared in place.. – Caridorc Jun 5 '15 at 17:32
-