CS161: Introduction to Computer Science
Homework Assignment 7
Due: 3/20 by 11:59pm

Playing Cards

In this assignment, you will use conditionals to implement a Java class representing a single playing card.

A playing card has the following attributes:

- A face value which is an integer ranging from 1 to 13. A face value of 1 corresponds to an Ace. A face value of 11, 12, or 13 corresponds to a Jack, Queen, or King respectively. These three are known as *face* cards because they usually have a face drawn on them.
- A suit which is either: `diamond`, `heart`, `spade`, or `club`. Diamonds and hearts are red in color. Spades and clubs are black in color.

You will also write a second Java class (with a `main()` method) to test your playing card class.

This homework assignment is meant to give you practice writing methods that compare objects (e.g. the `equals()` method), using private methods, conditionals, and constants (i.e. using the `final` keyword). For a guiding example of how to write an `equals()` method, please see the `Die` class posted on the course webpage for 3/3.

When you're done, go through the style guide for the course (you can find this in your course packet or on the course webpage under "Resources") and make sure your code adheres to the style guide.

The `Card` class

Download the starter code from the course webpage. The starter code contains a `Card` class for you to finish. The `Card` class should have 2 constructors:

- A constructor that takes no input arguments. Instead, randomly generate both a face value and a suit for the card
- A constructor that takes both a suit and a face value as input arguments. Now that we can use conditionals, we can check that the user passed correct values to the constructor. If the suit or the face value passed by the user are not legal, go ahead and generate random values.

Remember that repeated code should be pulled out into a private method! Your class should also have the following methods:

- Accessor methods for the suit and face value
- A `toString()` method. Your `toString()` method should return a string in this precise format:

  ```
  [suit, faceValue]
  ```

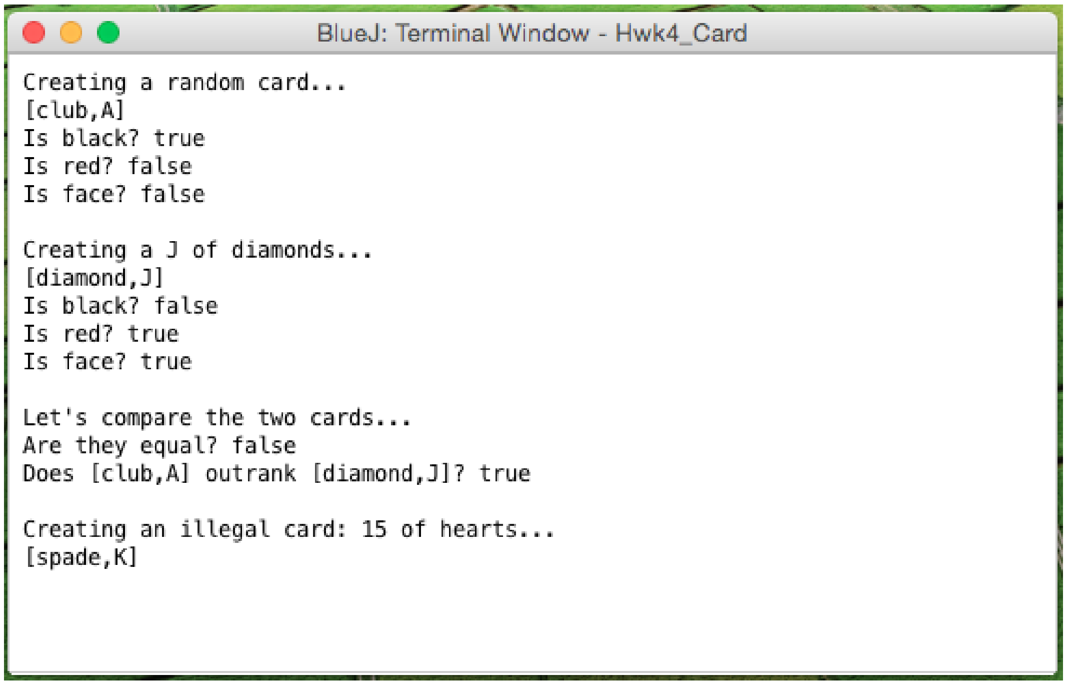
  where a face value of 1 is converted to an "A", an 11 to "J", a 12 to "Q" and a 13 to "K". For example, an ace of spades should return "[spade, A]" and a three of clubs should return "[club, 3]". (My tester code relies upon your `toString()` method returning this exact format.)

- The following methods should return a boolean
  - `isBlack()` returns whether the card is black
  - `isRed()` returns whether the card is red
  - `isFaceCard()` returns whether the card is a face card
  - `hasSameFaceValue(Card other)` returns whether this card and the other card have the same face value
  - `hasSameSuit(Card other)` returns whether this card and the other card have the same suit
  - `equals(Card other)` returns whether this card and the other card are equal (i.e. have the same suit and the same face value).
  - `outRanks(Card other)` returns true if this card has a strictly greater face value than the other card. The only exception is a face value of 1 (i.e. an Ace) which outranks all other face values.

Testing Your Class

Write a second class called `CardTester` that creates objects of type `Card` and uses the dot operator to call methods from the card class. Since you have two constructors, create at least two instances of the `Card` class.

Below is an example of what my `CardTester` prints when executed:



```
Creating a random card...
[club,A]
Is black? true
Is red? false
Is face? false

Creating a J of diamonds...
[diamond,J]
Is black? false
Is red? true
Is face? true

Let's compare the two cards...
Are they equal? false
Does [club,A] outrank [diamond,J]? true

Creating an illegal card: 15 of hearts...
[spade,K]
```

Notice that when I pass in an illegal value to the constructor (a face value of 15), it instead randomly initializes the card resulting in a King of spades.

Extras

Bored and looking for something to do over spring break? Well, now that you know about loops and the `ArrayList` class, you can do a lot more! Here are some ideas:

- Create a new Java class called `Deck` that represents a standard deck of 52 playing cards: 13 hearts, 13 diamonds, 13 clubs, and 13 spades. The playing cards can be held inside of a variable with type `ArrayList<Card>`. To initialize the deck of cards, you'll definitely want to use a while loop!
- Add a method to the `Deck` class that shuffles the cards in the array list. To shuffle, you can use a `Random` object to randomly choose two cards in the list and then swap those two cards. Do this repeatedly to shuffle the entire deck.
- What other methods could you add to the `Deck` class? What are the types of actions people usually perform on a deck of cards?

Submitting your assignment

You should submit your `hw7` folder with your `Card` class and your `CardTester` class.