CS161: Introduction to Computer Science
Homework Assignment 7
Due: 3/21 by 11:59pm

Comparing Objects, Loops, and ArrayList

In this week's assignment, you'll get practice comparing objects, using `while` loops, and using the `ArrayList` class. Since this is a substantial assignment, there is no section for written exercises.

This assignment asks you to write three Java classes: `Card`, `Deck`, and `DeckController`. The `Card` class represents a single playing card. The `Deck` class represents a standard set of 52 playing cards. The `DeckController` class will have a `main()` method that will create a new deck of cards, shuffle it, and then deal the top 3 cards of the deck.

Download the starter code from the course webpage. The starter code contains a `Card` class and `Deck` class for you to finish.

Remember that any repeated code should be pulled out into a private method!

The `Card` Class

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.

The `Card` class should have the following methods:

- A constructor that takes a suit and a face value as input arguments.
- Accessor methods for the suit and face value
- `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
- `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.
- A `toString()` method. Your `toString()` method *must 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 `Deck` Class

The `Deck` class should use an `ArrayList` to store a standard deck of 52 playing cards: 13 spades, 13 clubs, 13 diamonds, and 13 hearts. This is similar to the `BookCase` example we are working on in class.

The constructor should initialize the deck by creating and adding to the `ArrayList` each of the 52 cards. In addition to the constructor, you should add the following methods

- A method `shuffle()` that shuffles the cards in the deck. This method should not print anything and should not return anything. It simply shuffles the cards in the deck. (Hint: You'll need a `Random` object to shuffle.)
- A `dealCard()` method that **removes** the top card from the deck (i.e. the card in position 0 in the `ArrayList`) and returns it. This means the return type of the `dealCard()` method should be a `Card`.
- Finally, add a `toString()` method that prints out all of the cards in the deck.

The `DeckController` Class

Finally, the `DeckController` class is where you will actually use your `Deck` class! Create a class named `DeckController` with a `main()` method. Inside the `main()` method, write code to

- Create a deck of cards and print it to the screen
- Shuffle the deck of cards and print it to the screen
- Deal the top 3 cards of the deck. Print each card after it is dealt

The next page shows an example of what my `DeckController` prints when executed.

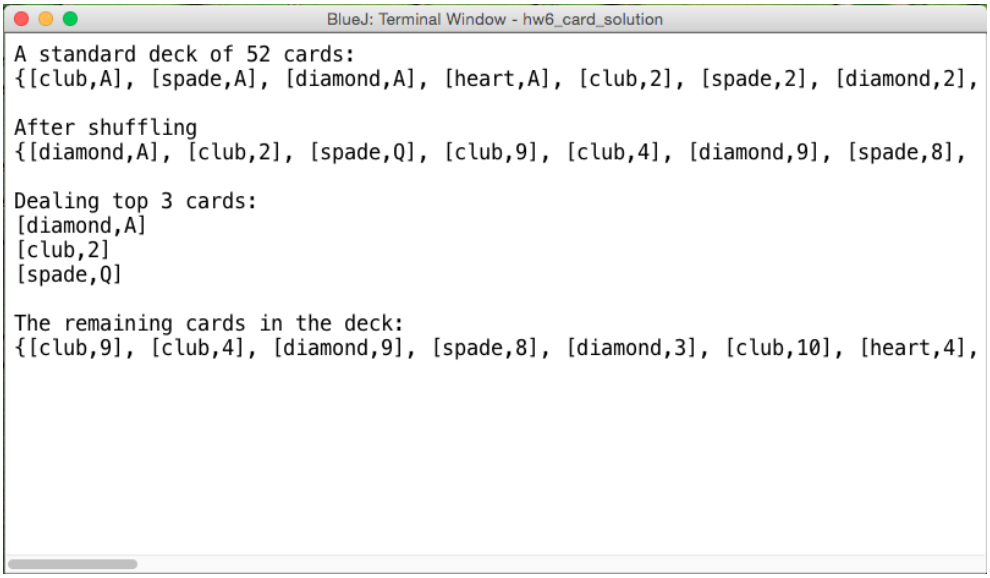
Style Guide

Before you submit your assignment, go through the checklist below and make sure your code conforms to the style guide.

Checklist

- ☐ All unused variables are deleted
- ☐ All instance variables are used in more than one method (if not, make them local)
- ☐ All instance variables are declared private
- ☐ All instance variables are initialized in the constructor
- ☐ Javadoc comment for all classes
- ☐ All methods have Javadoc comments (except for the `main` method)
- ☐ Proper capitalization of variables (final and non-final variables), methods, and classes
- ☐ All numbers have been replaced with constants (i.e. no magic numbers)
- ☐ Use white space to separate different sections of your code
- ☐ Code is correctly indented to improve readability

Read the "Style Guide" (under "Resources" on the course website) for more information.



Submitting your assignment

You should submit your `hw7` folder with your `Card`, `Deck`, and `DeckController` class via Moodle.