Locked: Content, Points, Due Dates & Availability Dates

Individual Card Game Part 1 Assignment #1

Locked | Published | Edit | ⋮

Card Game Part #1

This two-part assignment is meant to serve as a Java review. We will be building a console program that lets the user play a simplified version of Blackjack the card game (https://en.wikipedia.org/wiki/Blackjack (https://en.wikipedia.org/wiki/Blackjack)).

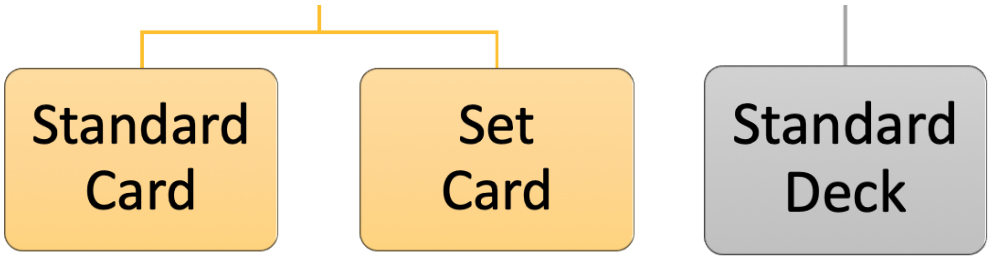
Objectives

Module

- To write classes that exist in a class hierarchy.
- To write a program that uses object-oriented features learned from an introductory sequence in Java, including: control-flow statements, inheritance & abstraction, and basic data-structures.

Building the Pieces of a Card Game

To create a card game, we must be able to represent cards and groups of cards (a deck of cards). We will begin by creating the following class hierarchies:



Here are descriptions of the classes in the "Card" hierarchy:

- **Card -** represents a playing card in any type of card game
- **StandardCard -** represents a standard playing card, with a suit and rank, that might be played in a Poker or Blackjack game
- **SetCard -** represents a card in the Set card game (https://en.wikipedia.org/wiki/Set_(card_game) (https://en.wikipedia.org/wiki/Set_(card_game)))
  - *Note: This card type has been added merely for practice purposes. The goal here is work with a class hierarchy in a small programming example.*

Here are descriptions of the classes in the "Deck" hierarchy:

- **Deck -** a deck represents two groups of cards, a dealer pile and a discard pile. Cards are dealt to a player from the dealer pile and then placed in the discard pile afterwards.
- **Standard Deck -** this represents a standard 52-card deck of playing cards (https://en.wikipedia.org/wiki/Standard_52-card_deck (https://en.wikipedia.org/wiki/Standard_52-card_deck))

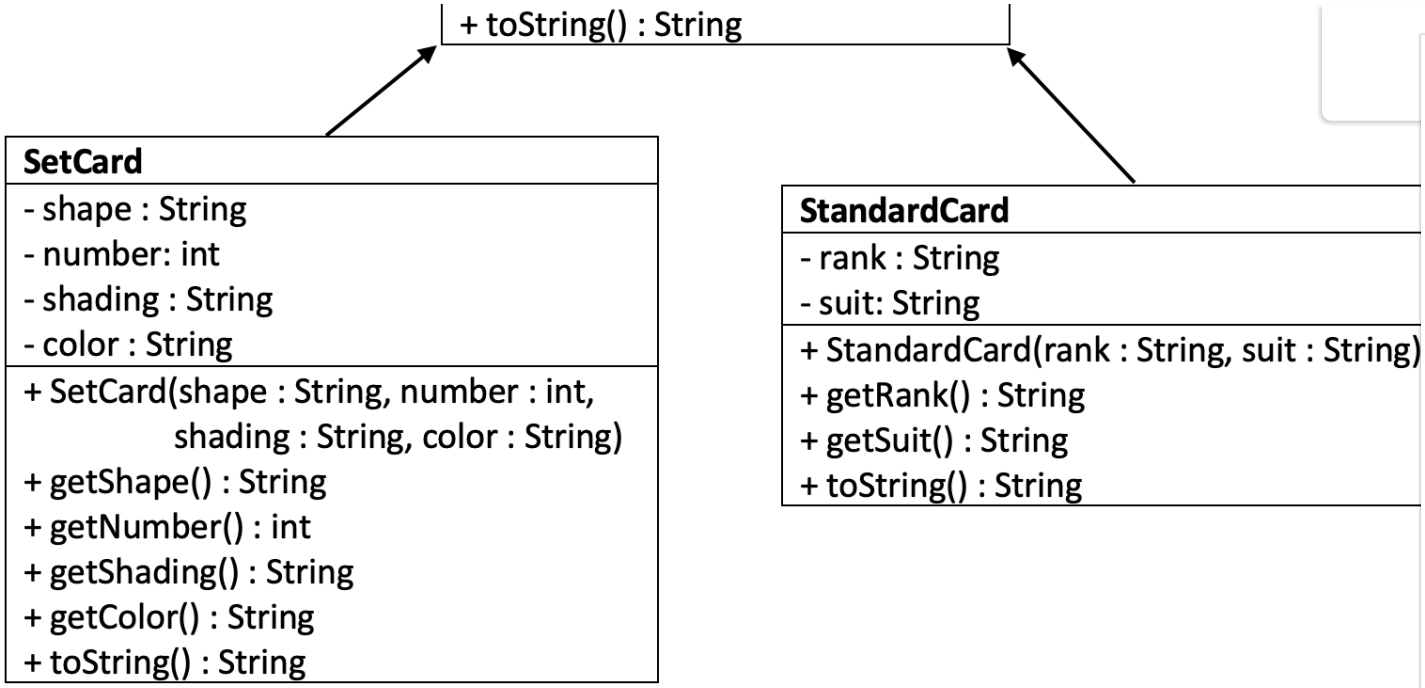
You should focus on implementing these classes and writing **jUnit** tests for the Card and the StandardCard classes, for an extra challenge you could write jUnit test classes for Deck and StandardDeck.

To meet this assignment's requirements, you should write two **jUnit** test methods for the Card and StandardCard class.

1) A @Test method to test the constructor for the Card and a @Test method to test the toString() of the Card

2)A @Test method to test the constructor for the StandardCard and a @Test method to test the toString() of the StandardCard

Card
- cardText : String
+ Card(cardText : String) + getCardText() : String



Card

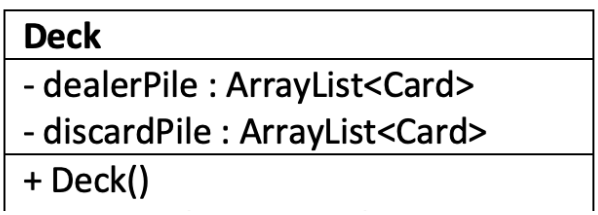
- Card(): creates a new card with a description text, for example "**Ace of Spades**" for a standard card or "**dashed-blue diamond three**" for a card from the Set Game. This text will be used to represent the card on the Java console.
- getCardText(): returns the description text
- toString(): also returns the description text

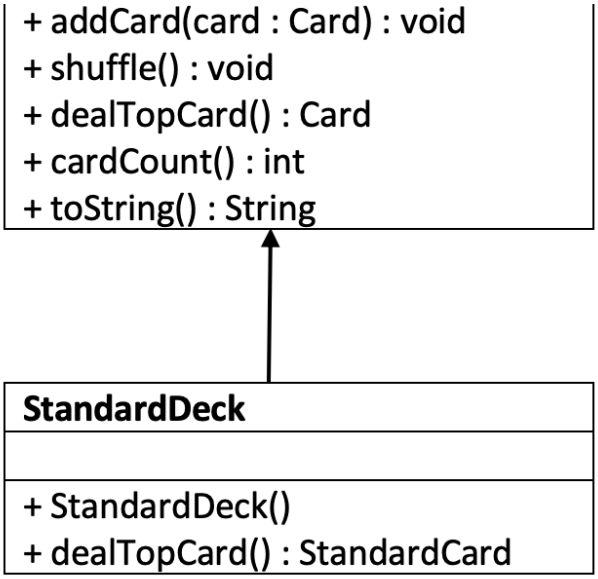
SetCard

- SetCard(): creates a new card for the Set Game given a number, shape, shade and color.
- getShape(), getNumber(), getShading(), getColor(): getters for the attribute of a card from the Set Game.
- toString(): returns the description text for a card from the Set Game

StandardCard

- StandardCard(): creates a new standard playing card with a rank and suit
- getRank(), getSuit(): getters for the attributes of a standard playing card
- toString(): returns the description text for a standard playing card





Deck

- Deck(): creates a new Deck object
- addCard(): adds a card to the dealer pile
- shuffle(): moves all cards from the discard pile to the dealer pile. Then randomizes the position of all cards in the dealer pile.
  - *Note: You should not be using Collections.shuffle() here. Instead write the algorithm as described below.*
- dealTopCard(): removes the "top" card from the dealer pile and places it in the discard pile. Then returns the removed card.
- cardCount(): returns the number of cards in the dealer pile. (this is useful as it tells you when you no longer have cards to deal to a player)
- toString(): returns a string representation of the deck using the format seen below.

```
45 cards in deck
**********
6 of spades
5 of spades
4 of spades
3 of spades
```

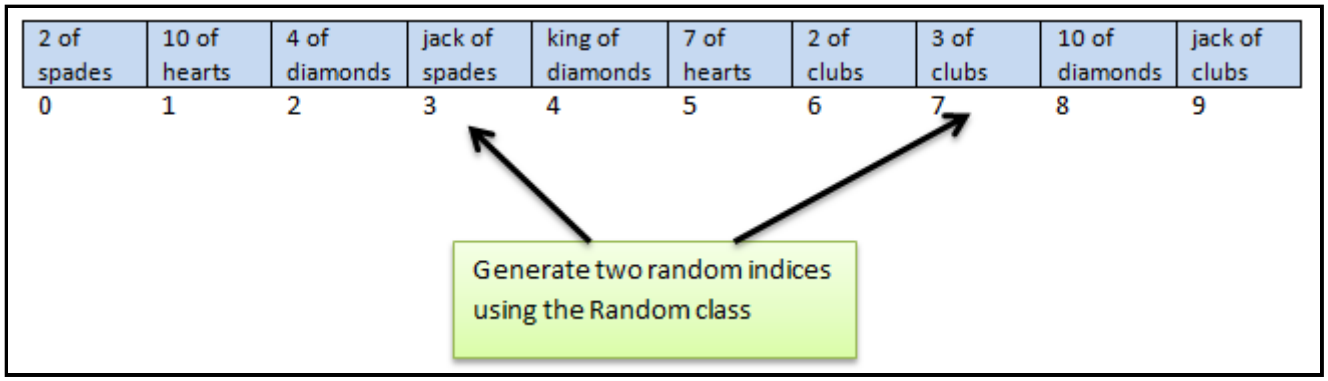
```
5 of spades
...

7 cards in discard
**********
4 of clubs
9 of hearts
Ace of hearts
...
```

StandardDeck

- StandardDeck(): creates a standard 52-card deck with each combination of the rank and suit values
  - Here are the ranks in a standard deck - 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, King, Ace
  - Here are the suits in a standard deck - Hearts, Diamonds, Spades, Clubs
- dealTopCard(): this method hides the Deck.dealTopCard() method. It retrieves the card from the parent class by calling dealTopCard() on the parent class and then casts the Card object to a StandardCard object for ease of use.

Implementation Hints

To write the shuffle method you can loop over each index of the array list of cards and swap that position with another random position in the list. This can be done using the Random class to repeatedly pick pairs of numbers and swapping their positions. For example:



spades	hearts	diamonds	clubs	diamonds	hearts	clubs	spades	diamonds	clubs
0	1	2	3	4	5	6	7	8	9

Writing Informal Tests

You will be provided with a test program for your classes: **DeckClient.java** (you are welcome to change/modify it) Here is the output of an informal test contained in the DeckClient.java file. You are welcome to add additional tests.

```
Creating deck...

52 cards in deck
******************
2 of clubs
2 of hearts
2 of spades
2 of diamonds
3 of clubs
3 of hearts
3 of spades
3 of diamonds
4 of clubs
...

0 cards in discard
******************
```

```
Shuffled deck...

52 cards in deck
******************
6 of diamonds
Jack of diamonds
3 of clubs
4 of hearts
...
4 of spades
9 of clubs
King of diamonds
6 of spades

0 cards in discard
******************
```

```
Dealing a few cards...

Dealt a 6 of spades
Dealt a King of diamonds
Dealt a 9 of clubs
```

```
Shuffled deck again...

52 cards in deck
******************
```

```
49 cards in deck
**********
6 of diamonds
Jack of diamonds
3 of clubs
4 of hearts
...
4 of spades

3 cards in discard
**********
6 of spades
King of diamonds
9 of clubs
```

```
Ace of clubs
5 of spades
10 of diamonds
7 of spades
...
Jack of clubs
Jack of diamonds
4 of clubs
2 of hearts

0 cards in discard
**********
```

Make sure you test the following scenarios:

- Create a new standard deck
- Print out the cards to verify all 52 cards are present
- Shuffle the deck
- Print out the cards to verify they are in random order
- Deal a few cards
- Verify the dealt cards are then in the discard pile
- Shuffle the deck
- Print out the cards to verify that the discard pile was shuffled back into the dealer pile

Points	50
Submitting	a file upload
File Types	zip

Due	For	Available from	Until
Oct 4	Everyone	-	Oct 11 at 11:59pm

Card Game #1 Rubrics

Criteria	Ratings		
Card, SetCard and StandardCard classes have all fields, constructors and methods presented in the class diagrams above. A string representation of both card types are passed from the child class to the parent class. jUnit tests are provided to thoroughly test these classes.	**10.0 pts Full Marks**	**0.0 pts No Marks**	10.0 pts
A new deck of cards can be created with the Deck class. Cards can be added to the dealer pile with the addCard() method and moved from the dealer pile to the discard pile with the dealTopCard() method. The cardCount() method correctly reports the number of cards in the dealer pile.	**5.0 pts Full Marks**	**0.0 pts No Marks**	5.0 pts
Calling shuffle() on a Deck object will randomize the position of the cards in the deck. At least 52 cards must be adjusted for the shuffle() to be valid.	**5.0 pts Full Marks**	**0.0 pts No Marks**	5.0 pts
The StandardDeck class creates a 52-card deck with each combination of suit and rank possible. The dealTopCard() method is overridden and correctly returns a StandardCard object.	**5.0 pts Full Marks**	**0.0 pts No Marks**	5.0 pts
Each class has a toString() method that is available with appropriate output. All Deck objects should have a toString() method with the format seen in the screenshots above.	**5.0 pts Full Marks**	**0.0 pts No Marks**	5.0 pts
An informal test program is provided that includes the sequence described above: deck creation, printing the deck, shuffling the deck, dealing cards and then reshuffling the deck - This will now be given to you so you can focus on creating JUnit test classes	**0.0 pts Full Marks**	**0.0 pts No Marks**	0.0 pts
The submission follows all style considerations described in the Java style guide provided on Canvas.	**5.0 pts Full Marks**	**0.0 pts No Marks**	5.0 pts
The program has a comment header and Javadocs as formal documentation.	**5.0 pts Full Marks**	**0.0 pts No Marks**	5.0 pts
JUnit Test Classes are provided for the Card Class and the Standard Card Class.	**10.0 pts Full Marks**	**0.0 pts No Marks**	10.0 pts

Criteria	Ratings	
		Total Points: 50.0