Activity 7.2 - Quick Draw

Overview

In this activity, you will create a multi-class program to construct and use a class that represents a standard 52-card deck of playing cards. You will test your class by using it to play a simple card game.

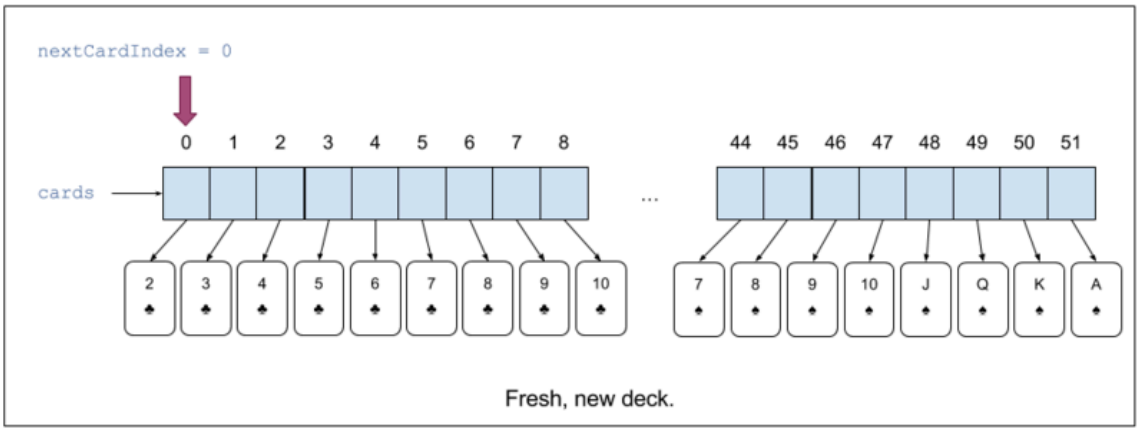
Instructions

Getting Started

1. Create "`Activity7.2`" in VScode.
2. Import enums: `Suit.java` and `Rank.java`
3. Import interface: `DeckOfCardsInterface.java`
4. Import class: `Card.java`
5. Create a `DeckOfCards` class that implements the `DeckOfCardsInterface`
6. Create a `QuickDraw` class with a `main` method. This will be your driver class for testing your `DeckOfCards`.

Part 1: Write a DeckOfCards class

The `DeckOfCards` class will contain and manage an **array** containing the 52 unique Card objects.



- Create a new class called `DeckOfCards` that implements the `DeckOfCardsInterface`. Add the unimplemented methods (you can just leave them as "stubs" for now). **NOTE:** The *dealtCards()* and *remainingCards()* methods are not used in this activity and may return *null*
- A `DeckOfCards` must have the following instance variables *or constants*.
    - A *deck size* (a standard deck is *always* 52 cards).
    - An array of *cards* in the deck.
    - A *next card index* to keep track of where we are in the deck.
- Write a constructor that instantiates all of the instance variables needed to create a new deck of cards. The constructor will need a nested loop to go over all combinations of Suit and Rank values to create a card for each of the 52 possibilities.
    - The following nested for-each loop will make sure that you are creating a card with each suit and face value

      ```
      for(Suit s : Suit.values()) {
       for(Rank r : Rank.values()) {
           //Create a new card and add it to your deck.
        }
      }
      ```
    - You will also need a variable to keep track of where you are inserting your next card in the cards array e.g.

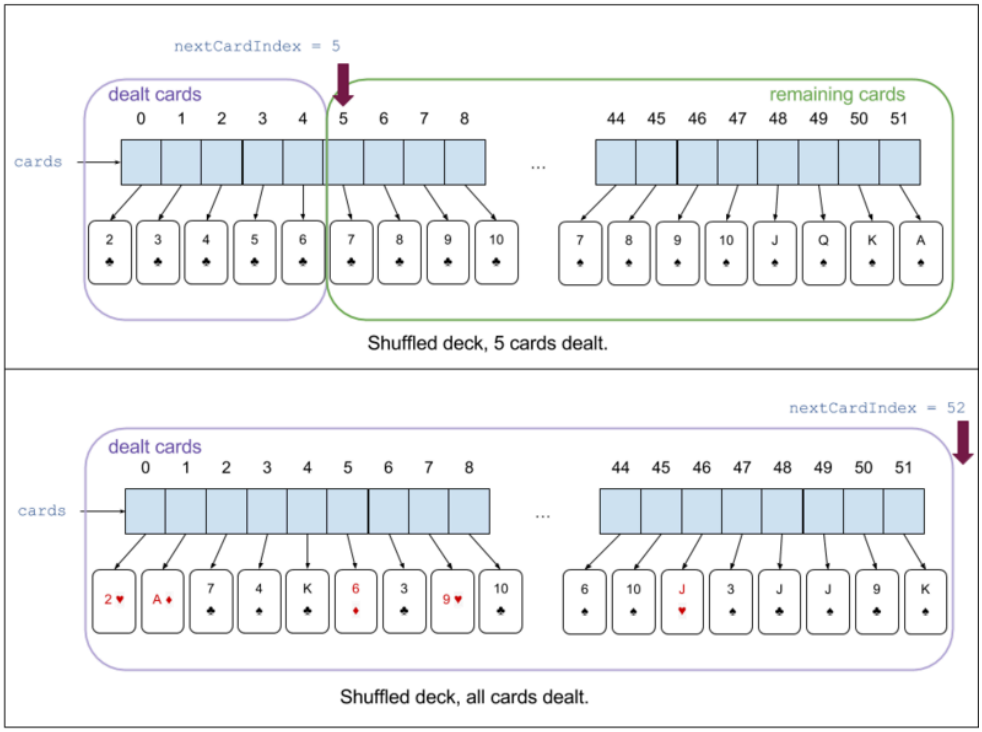
      ```
      cards[i] = new Card(s, r);
      ```

- Write a `toString` method that will return a well-formatted String including all the cards in the deck.
- Test that your `DeckOfCards` is working by creating a new `DeckOfCards` object named `deck` in the main method of your `QuickDraw` class. Call the `toString` and verify that it prints all the cards in the deck.

- Implement the `shuffle()` method. The table below provides *pseudocode* for a basic shuffle algorithm (note: it isn't a perfectly random shuffle, but is a simple one. If you have time later, try to come up with a better one!). Make sure to reset the next card index back to the first card in the deck after shuffling.

```
function shuffle():
      for( i = 0; i < cards.length; i++)
            j = random integer from 0 to cards.length
            swap cards[i] and cards[j]
       reset next card index
```

- Test your `shuffle()` method by shuffling the `DeckOfCards` in `QuickDraw`, then print the `deck` after the shuffle to verify that they are randomized.
- Implement the `draw()` method. The draw method should draw one Card from the **top of the deck**. After 52 calls to `draw()` the deck will be "empty" and the method should return `null`. To make this work, you will keep track of the index in the card array that separates the dealt cards from the undealt cards using your `nextCardIndex` instance variable.



- Implement the `numCardsRemaining()` and `numCardsDealt()` methods. You will calculate these values based on the `nextCardIndex`.
- Test your draw method by drawing the first 3 cards off the top of the deck. Print each card after it is drawn as well as the number of cards dealt and number of cards remaining and verify that you get the correct results. (**HINT: should sum to 52**)

Note:

- **Help! Deck Of Cards Unicode characters showing up as (?) on my windows computer. Click here for how to fix this issue.**

Part 2: Create a simple game using DeckOfCards class

1. Modify the `QuickDraw` class so that it requires the user to provide the following **two command-line arguments that specify**:
    - Player1 name
    - Player2 name
2. Check the number of items (`length`) in the args array to make sure that it contains the proper number, otherwise display an error message to the user and exit.
3. Store the `String` values from `args[0]` and `args[1]` to `String` variables named `player1` and `player2` respectively
4. Draw and display a card for player1 from the `deck`
5. Draw and display a card for player2 from the `deck`
6. Use the `compareTo()` method from the `Card` objects to determine which player wins the draw. Print a nice Winner message to the console displaying the winning player's name. Don't forget to handle a tie.

Terminology Identification

In your code add comments identifying examples of the following: command-line arguments, validating arguments, array of objects. These should be identified with an @keyterm tag within the comment.

Code Review

When you are finished with this activity, pair up with a classmate and review each other's code to make sure it meets all the requirements.

Submission

After completing the assignment, use the assignment link in Canvas and follow the submission instructions there. You will upload your `QuickDraw.java` and `DeckOfCards.java` files and put your reflection in the "Comments" box.

Reflection Requirements

Write a one paragraph reflection describing your experience with this activity. The reflection should also include the name of your code review partner AND something interesting you found in their code. Please review the activity rubric for more details.