

Mock SSF Assessment 2

Study the following REST endpoint at <https://deckofcardsapi.com/> . No registration is required

The 2 endpoint of interest are

- Endpoint 1 - Shuffle the Cards, which creates and shuffles a new deck of card
- Endpoint 2- Draw a Card, draws one or more cards from the created deck

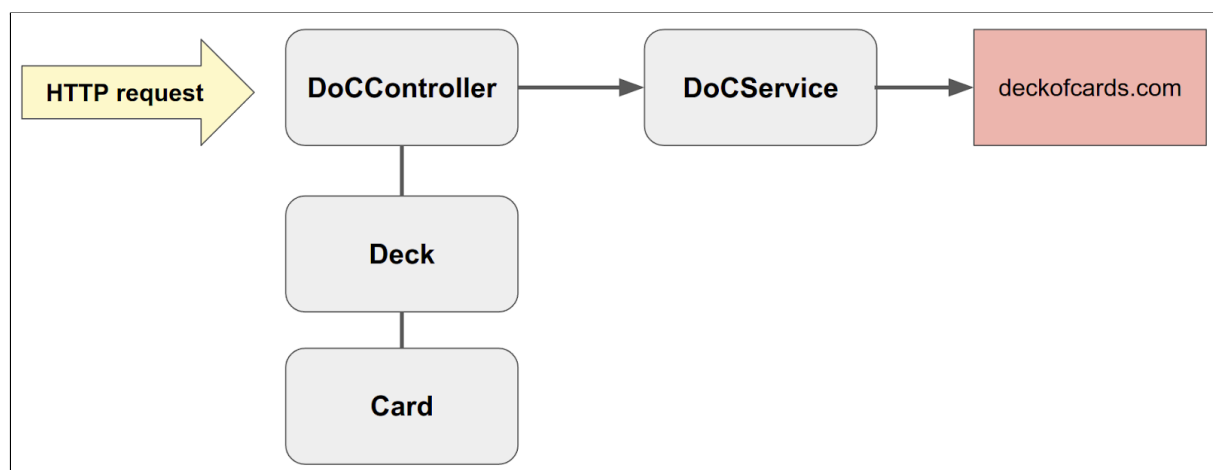
Write a Spring Boot MVC application to allow players to create and draw cards from a deck

Create the following 3 classes

- `DoCController` - handles all request from the user
- `DoCService` - communicates with the deckofcardsapi endpoint
- `Card` - model of a card
- `Deck` - model of a deck of cards

You may use other classes in the assessment, but the above 3 classes are mandatory; how they are used will be described in the following tasks.

The following diagram shows their interaction



Task 1

Write a landing page (index.html) with a single button with the label 'Create a new Deck'

The button will send the following HTTP request to the DoCController

```
POST /deck
```

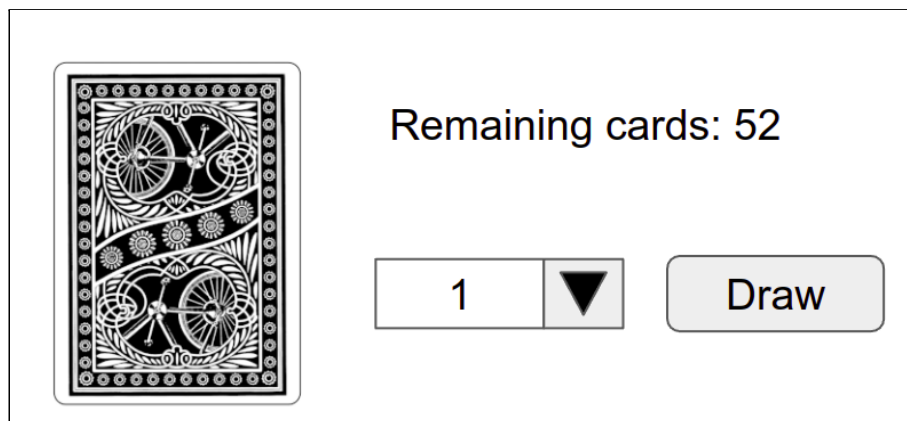
Write a method in DoCService called `createDeck()` that makes a request to the Endpoint 1 with `deck_count = 1`. `createDeck()` should return a `Deck` object.

Deck object should contain all attributes returned by Endpoint 1 except the success attribute

The signature of `createDeck()` is as follows

```
public Deck createDeck()
```

Once the deck has been created, return the following view to the client



The view consist of

- Back image of the deck - <https://deckofcardsapi.com/static/img/back.png>
- Number of remaining cards in the deck

- A HTML `<select>` that allows the user to select the number of cards to be drawn. This number should not be greater than the remaining cards. This should default to 1
- A button to perform the draw

You are free to style the view according to your taste but the above mentioned 4 items must be present

Task 2

When the Draw button is pressed (assume the user has decided to draw 3 cards), the following HTTP request will be send to `DoCController`

```
POST /deck/<deck_id>
```

where `<deck_id>` is returned by `deckofcards.com` from Task 1.

`DoCController` will use `DoCService` to draw cards from the given deck with Endpoint 2.

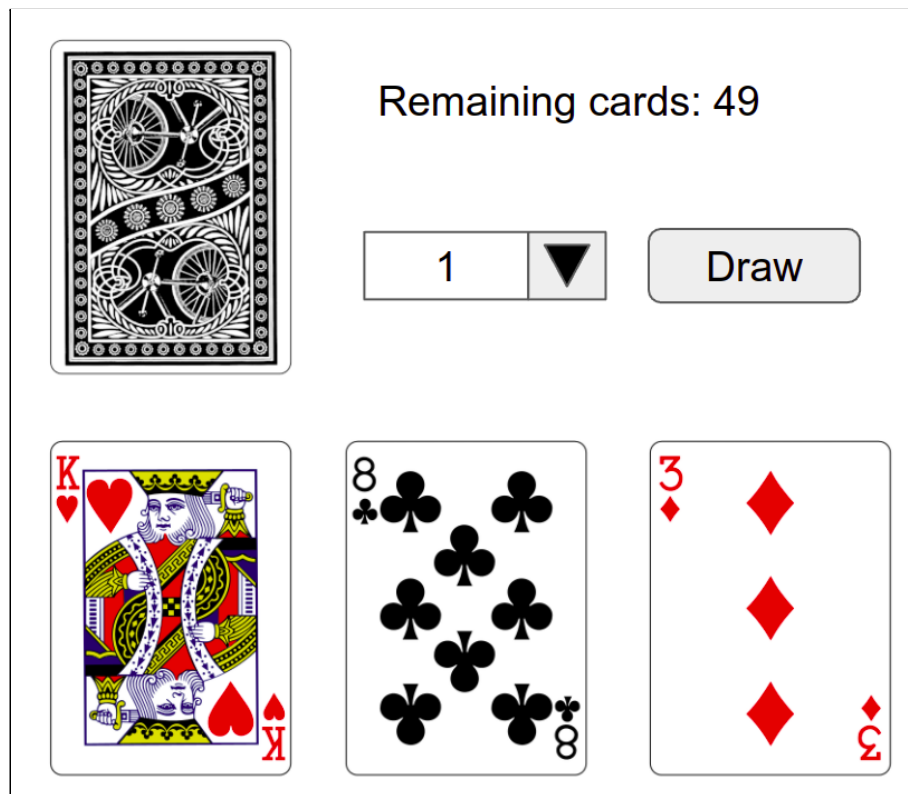
Write a method in `DoCService` called `drawCards()` ; the method will return a `Deck` object. `drawCards()` method signature is as follows

```
public Deck drawCards(String deckId, Integer count)
```

where `deckId` is the `<deck_id>` and `count` is the number of cards to draw

The returned `Deck` object should contain all attributes returned by Endpoint 1 except the `success` attribute.

Render the following view back to the user



The view should reflect the number of remaining cards in the deck and also should only allow the user to draw cards equal to or less than the remaining cards.

Subsequent draws will display the newly drawn cards replacing the cards from previous draws.


When the remaining cards is 0, the view should

- Not display the back image
- The draw button should be disabled
- HTML `<select>` should display 0

Task 3 (Optional)

When you draw new cards, they should be added to what is already drawn; for example, if you draw 3 cards, followed by 2, then all 5 cards should be displayed instead of the most recent as required by Task 2.

See the following figure



Remaining cards: 47

1

▼

Draw

