This practice exercise will assess your knowledge of concepts from COP2551C and COP2800C.

French-suited playing cards (https://en.wikipedia.org/wiki/French-suited_playing_cards) are one of the most common types of cards in use today. These cards consist of 4 suits (categories) in a 52-card deck as described in the following table:

Suit Name	Symbol	Ranks (Number Card Values)	Face Cards
Clubs	♣	Ace (1), 2 through 10	Ace*, Jack, Queen, King
Diamonds	♦	Ace (1), 2 through 10	Ace, Jack, Queen, King
Hearts	♥	Ace (1), 2 through 10	Ace, Jack, Queen, King
Spades	♠	Ace (1), 2 through 10	Ace, Jack, Queen, King

Design and implement a Java program which represents a standard 52-card deck of playing cards using the above information.

*Note: although the Ace is not technically a face card, we will represent it as one in order to make its identification by name easier.

The playing card deck must include 4 suits: Clubs, Diamonds, Hearts, and Spades.
The deck must be implemented as a public class with either a single **Java array** or **Java ArrayList** which stores playing card objects. The array/ArrayList must be declared as a private instance variable of the class.

The playing cards in the deck must be instantiated from a separate PlayingCard class with the following characteristics:

- A private instance variable representing the numeric point value of the card using the "BlackJack" numbering system as follows:
  - Ace (1 point)
  - 2, 3, 4, ... , 10 (2 points, 3 points, 4 points, ... , 10 points)
  - Jack (10 points)
  - Queen (10 points)
  - King (10 points)
- A private instance variable representing the name for "face cards" (Ace, Jack, Queen, King). **Your program must use an enumerated type to represent this value.** Non-face cards should be represented by the enum value None.
- A private boolean instance variable indicating if the card has been dealt in a hand
- A private instance variable representing the suit as described above (Clubs, Diamonds, Hearts, Spades). **Your program must use an enumerated type to represent this value.**
- Public accessors for all of the above.
- Only one mutator is required: the dealt indicator, since it can be changed after the card is created; the other member variables should remain read-only.
- An overridden toString method which returns a String that represents the card's state (all instance variable values). Use this method whenever you display a card. Do not display any output in this method.

Your playing card deck class must include the following methods:

1. a method to deal a random selection of cards of any size between 1 and 52 with no duplicates. Once a card has been used in a hand it can no longer be used (this is indicated by setting the boolean dealt indicator to true). Do not display any output in this method, simply return a hand containing the dealt cards.

   Hint: Since our deck is small I use a brute-force algorithm for this feature:
   - start with an empty hand (e.g. an empty array list of cards)
   - loop for the size of the hand (should be a parameter passed to your method)
   - select a random card from the deck (use the Random API with the nextInt method, there is an example of this in Ch. 1 of the textbook and in the solution for this practice exercise)
   - if the card has not been used in a hand, add it to the hand and increment the loop index (do not increment the index if the card has already been used)
   - once the loop exits, you will have the full hand which is the return value of the method

2. a method to display the dealt hand (there should not be any output in the deal method described above).
3. an overridden toString method which returns a String representing the contents of the entire deck. This method must call the toString method in the PlayingCard class as necessary. Do not display any output in this method.

Your program must provide a welcome message, display the full deck, and then display the dealt hand as shown in the sample output below.

Additional non-functional requirements:

- Use the var keyword at least once in your code to demonstrate your understanding of LVTI (Local Variable Type Inference).
- Use a text block for the welcome message to demonstrate your understanding of text blocks

All code should be saved in a single .java file; only the PlayingCardDeck class should be declared public.

Submit your .java file by committing and pushing it to your GitHub Classroom repository as described in the document attached to this assignment. Do not subnit any files tio Canvas.

Sample output is shown here (full deck should be displayed, truncated here for brevity):

```
Welcome to the Playing Card Simulator!
I will now create your deck of 52 cards and deal a random hand of size 5.

Here is your Playing Card Deck:
        Ace of Clubs (value = 1 dealt: false)
        2 of Clubs (value = 2 dealt: false)
        3 of Clubs (value = 3 dealt: false)
        ...
        Jack of Spades (value = 10 dealt: false)
        Queen of Spades (value = 10 dealt: false)
        King of Spades (value = 10 dealt: false)
```

```
size of hand = 5
        Queen of Clubs (value = 10 dealt: true)
        3 of Hearts (value = 3 dealt: true)
        6 of Hearts (value = 6 dealt: true)
        Ace of Diamonds (value = 1 dealt: true)
        Ace of Hearts (value = 1 dealt: true)
```

# cop2805cMod1GPA

```
Welcome to the playing card simulator!
I will now create your deck of 52 cards and deal a random hand of size 5.

card deck created.
size of hand = 5
8 of Diamonds (dealt: true)
King of Spades (dealt: true)
Jack of Spades (dealt: true)
2 of Clubs (dealt: true)
4 of Hearts (dealt: true)
```