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

The *Pokémon™ Trading Card Game* is a strategy-based card game where two players assume the role of Pokémon trainers and use their Pokémon to battle each other. The Pokemon characters each have an associated color and hit point value (used for scoring). The following table summarizes several characters and their attributes (note that these hit point values are fabricated and probably not accurate):

Character Name	Color	Hit Points
Charmander	Red	50
Squirtle	Blue	50
Pikachu	Yellow	60
Bulbasaur	Green	60
Koffing	Purple	50
Charizard	Black	75

Design and implement a Java program which represents a deck of Pokemon cards using the above characters (if you are familiar with the game, feel free to substitute/add characters). There should be one card per character. Our program will implement a simple "attack" strategy where a pair of random cards are selected from the hand in a single game turn (this isn't actually how the game is played, but we are keeping it simple for now).

The deck must be implemented as a public class with either a single **Java array** or **Java ArrayList** which stores Pokemon Card objects. The array/ArrayList must be declared as a private instance variable of the class.

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

- Private instance variables which represent the character name, numeric hit point value, and color as described in the above table.
- A private boolean instance variable which indicates whether the card has been used in an attack.
- Public accessors for all of the above
- Only one mutator I required: the attack indicator, since it can be changed after the card is created; the other member variables should remain read-only.
- An overridden toString method in your PokemonCard class to display the card's state (all instance variable values). Use this method whenever you display a card.

Your Pokemon deck class must include the following methods:

1. a method to attack by dealing a random selection of cards of any size between 1 and the total number of characters with no duplicates. Once a card has been used in an attack it can no longer be used (this is indicated by setting the boolean attack indicator to true). Do not display any output in this method, simply return a hand containing the attack 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 an attack, 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 attack hand (there should not be any output in the attack method described above).

Your program must provide a welcome message, display the full deck, and then display the attack 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

Sample output is shown here:

Welcome to the Pokemon game!
I will now create a deck and deal a random two-card attack hand.

Here is your Pokemon Deck:
Charmander, Red, HP=50
Squirtle, Blue, HP=50
Pikachu, Yellow, HP=60
Bulbasaur, Green, HP=60
Koffing, Purple, HP=50
Charizard, Black, HP=75

Charmander, Attack! (Red, 50)
Koffing, Attack! (Purple, 50)