Deck of Cards

We will be working on building a program together that allows us to model and simulate the behaviors for a deck of cards.

Part I - The Card Class

The standard cards in a 52-card deck have the same properties and behaviors. Develop a class that models a card using the below properties and behaviors.

Properties

Name	Type	Get	Set	Description
Suit	string	X		Represents which suit that the card is for.
Value	int	X		Represents what value the card represents.
IsFaceUp	bool	X		Represents if the card is currently facing up.
Color	string	X		**Derived Property**, returns "red" if Suit is hearts or diamonds, "black" for clubs or spades.
FaceValue	string	X		Returns the face value (e.g. "2 of Diamonds, 9 of Spades, King of Hearts, etc.")

Constructors

The card class can have a single constructor that requires a known suit and value when created. Every card that is created is face down by default.

Constructor	Description
Card(string suit, int value)	Initializes a new card using suit and value. All new cards are face down by default.

Behaviors

Method Name	Returns	Description
TurnOver()	bool	Flips the card over by changing IsFaceUp. Returns if the card is currently facing up or down.

Part II - The Deck Class

A card is pretty boring by itself. But put it in a deck of cards and you can do a lot of pretty cool things with it! Develop a class that models the standard deck using the below properties and behaviors and you'll have the ability to begin building a program that uses a deck of cards.

Some questions to consider before reading further

- What logic should we use to populate the cards into the deck?
- The deck needs to store the cards, but we don't want to make it accessible to other parts of the program right? After all, you don't give someone the cards in the deck, you let them draw the top X cards.

Private Variables

Name	Type	Description
cards	List<Card>	Holds all of the cards in the deck as a private variable to *protect* the data from outside the class.

Properties

Name	Type	get	set	Description
NumberOfCardsRemaining	int	X		Returns the number of cards remaining in the deck (i.e. cards.Count).

Constructors

Constructor	Description
Deck()	Initializes a brand new deck and populates it with all 52 standard playing cards.

Behaviors

Method Name	Returns	Description
Draw(int numberOfCards)	Card[]	Removes the specified number of cards from the deck and returns them to the caller. Returns an array because, we don't want the user to draw 1 and add their own!

Part III - Print Out the Deck

Let's make sure the deck is working. Go over to your Program.cs file and instantiate a new instance of the Deck class.

1. Once you have an instantiated deck, try using the Draw(int) method that you wrote to draw a single card and print its FaceValue to the screen to make sure it works!
2. If it does work, try writing some code that draws each of the cards from the deck and prints their FaceValue to the screen. If you see all 52 cards in your deck, nice work!

We're not going to build any kind of user interface around our Deck of Cards yet, but it lets us confirm that we are on the right path!

Part IV - Shuffling

Our deck needs a Shuffle behavior. Lets add this new method so that when invoked, it shuffles the order of the cards.

Method Name	Returns	Description
Shuffle()	void	Shuffles all of the remaining cards inside the deck so that they are not in the same order. Returns void because it just changes the order of the cards in the deck.