**Deck of Cards (50pts, +15 ec)**

**Project Description & Goal**

Now that we know how to create a stack class, let’s use what we have learned to create a class that represents a deck of cards. Decks of cards are useful in many scenarios, so once we have our data structure built it will be plug-and-play for other projects.

**Project Specifications**

* Create a Card Class that will be used as the elements in the Deck Class.
  + Suit = string (see ec)
  + Value = string (see ec)
* Create a Deck Class that behaves like a stack class.
  + It will actually contain 2 stacks! The Deck itself and the Discard Pile.
* Create a main menu that allows for simple testing of the Deck Class.
  + Have a List of cards that can be considered the Hand.

**Required Methods**

* Card
  + public void Print()
    - Prints a card in this format
      * -----------------
      * | Ace of Hearts |
      * -----------------
* Deck
  + public void Shuffle()
    - Randomly sorts the deck.
    - Puts the discard pile back into the deck.
      * (Make this easy by simply re-creating the deck, and emptying the discard pile!)
  + public Card Draw()
    - Same as Pop(). Draws a card from the deck.
  + public void Discard(Card c)
    - Discard one of the cards to the discard pile.
  + public void PrintDeck()
    - Prints the remaining elements of the deck.
  + public void PrintDiscard()
    - Prints the elements in the discard pile

**Project Learning Objectives**

* Understand how a stack can be used to implement a Deck class.
* Experience working with multiple semi-complex classes together in an interesting way.

**Project Demonstrated Competencies**

1. Card class implemented correctly.
2. Deck class implemented correctly and uses the Card class.
3. Main menu allows for thorough testing of the Deck class.
4. Optional: Instead of the Card having two string values for face and value, use enumerations instead.
   * [**https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/enum**](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/enum)
5. Optional: Create two hands (simulate two players) and let them take turns with the menu.

**Rubric**

|  |  |  |
| --- | --- | --- |
|  | **Description of perfect implementation** | **Score** |
| Competency #1 | Card class has the required method and works flawlessly with the Deck class. | \_\_\_  15 |
| Competency #2 | Deck class has all the required methods and works flawlessly with the Card class. | \_\_\_  20 |
| Competency #3 | Main menu allows for extensive testing of the Deck class. | \_\_\_  15 |
| Competency #4 | A Card has two enumerations: Face and Value | \_\_\_  +5 |
| Competency #5 | In main, create two hands and allow the players to alternate their options. | \_\_\_  +10 |