

Name: **Session:**
Programming II
Lab Exercise 5/2/2024
Making a Card Game

In this exercise we will create the components for a card game. In order to create a card game, we will require a `PlayingCard` class, a `DeckOfCards` class, and a `PlayerHand` class. In this exercise, we will make a console application to test our classes. Create a new Project but instead of creating a Windows Application, create a Console Application. This will create a module with a function named `Main`. At the end of this handout, you will find source code for all of the classes.

1. Let's start by creating a `PlayingCard` class. The class `PlayingCard` will require the following:
 - a. two private fields of type `String`, `myFace` and `mySuit`
 - b. a constructor to initialize the `PlayingCard` object
 - c. a `toString` method
2. Now let us create the `DeckOfCards` class. The `DeckOfCards` class will require the following:
 - a. a private array of 52 `PlayingCard` objects called `deck`
 - b. a private integer `currentCard` to serve as the index of the card to be dealt
 - c. a constructor function to initialize the `DeckOfCards` object
 - d. Create a `shuffle` method (we will create this later)
 - e. Create a `deal` method (we will create this later)
3. Before we build the `PlayerHand` class, let's test the two classes we have created by creating a console to create and print our deck of playing cards.
4. If our class works correctly so far, let us create a `shuffle` method.
5. Now let's add a `deal` method
6. Now create a `PlayerHand` class which contains the following:
 - a. a private array of 5 `PlayingCard` objects
 - b. an `addCard` method which adds a card to the `PlayerHand`
 - c. a `showHand` method that displays the `PlayerHand` object.
7. Now write a `Main` method that will play a two-player game.
8. When you have your two-player game working, modify it to be a four-player game.
9. Submit the source code and screen shot of your final game.

Program.cs

```
static void Main(string[] args)
{
    //Create a deck of cards and two players
    DeckOfCards theDeck = new DeckOfCards();
    PlayerHand player1 = new PlayerHand();
    PlayerHand player2 = new PlayerHand();

    //Shuffle the deck
    theDeck.shuffle();

    //Deal the cards
    for (int count = 0; count <= 4; count++)
    {
        player1.addCard(theDeck.deal());
        player2.addCard(theDeck.deal());
    }

    //Show player 1 hand
    Console.WriteLine("Player 1 Hand");
    player1.showHand();
    Console.WriteLine("Player 1 value = " + player1.getValue());

    //Show player 2 hand
    Console.WriteLine(Environment.NewLine + Environment.NewLine +
        "Player 2 Hand");
    player2.showHand();
    Console.WriteLine("Player 2 value = " + player2.getValue());

    Console.WriteLine();

    //Determine which player won
    if (player1.getValue() == player2.getValue())
        Console.WriteLine("It's a draw");
    else if (player1.getValue() > player2.getValue())
        Console.WriteLine("Player 1 wins");
    else
        Console.WriteLine("Player 2 wins");
}
```

PlayingCard Class

```
class PlayingCard
{
    private string myFace;
    private string mySuit;
    private int myValue;

    public PlayingCard(string f, string s, int v)
    {
        myFace = f;
        mySuit = s;
        myValue = v;
    }

    public override string ToString()
    {
        return myFace + " of " + mySuit;
    }

    public int getVal()
    {
        return myValue;
    }
}
```

DeckOfCards Class

```
class DeckOfCards
{
    private PlayingCard[] myDeck = new PlayingCard[52];
    private int currentCard;
    Random r = new Random();

    public DeckOfCards()
    {
        int count;
        string[] faces = new string[] { "Ace", "Deuce", "Three", "Four", "Five", "Six",
            "Seven", "Eight", "Nine", "Ten", "Jack", "Queen", "King" };
        string[] suits = new string[] { "Hearts", "Diamonds", "Clubs", "Spades" };
        int[] values = new int[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 10, 10 };
        currentCard = 0;
        for (count = 0; count <= 51; count++)
        {
            myDeck[count] = new PlayingCard(faces[count % 13], suits[count / 13],
values[count % 13]);
        }
    }

    public void shuffle()
    {
        int first, second;
        PlayingCard temp;
        currentCard = 0;
        for (first = 0; first <= 51; first++)
        {
            second = r.Next(0, 51);
            temp = myDeck[first];
            myDeck[first] = myDeck[second];
            myDeck[second] = temp;
        }
    }

    public PlayingCard deal()
    {
        {
            if (currentCard < 52)
            {
                currentCard++;
                return myDeck[currentCard - 1];
            }
            else
            {
                return new PlayingCard("Joker", "Joker", 0);
            }
        }
    }
}
```

PlayerHand Class

```
class PlayerHand
{
    private PlayingCard[] myHand = new PlayingCard[5];
    private int currentCard;

    public PlayerHand()
    {
        currentCard = 0;
    }

    public void addCard(PlayingCard pc)
    {
        if (currentCard < 5)
        {
            myHand[currentCard] = pc;
            currentCard++;
        }
        else
        {
            showHand();
        }
    }

    public void showHand()
    {
        int count;
        string str = "";
        for (count = 0; count <= 4; count++)
            str += myHand[count].ToString() + Environment.NewLine;

        Console.WriteLine(str);
    }

    public int getValue()
    {
        int count, sum = 0;
        for (count = 0; count <= 4; count++)
            sum += myHand[count].getVal();
        return sum;
    }
}
```