Programming with C#

# Creating Arrays and Classes Practice

## Create Classes

1. Open Visual Studio if it is not already open.
2. Locate the folder where you extracted the companion content.
3. In the Chapter 7 folder, locate either the Windows 7 or Windows 8 folder.
4. Open the Classes project.
5. Open the program.cs file.
6. Locate the Main method and notice that it contains an exception handling block that calls a method named doWork(), from within the try block.
7. Directly above that is the doWork method with a // TODO: comment.
8. Open the Point.cs file and notice that it also contains a // TODO: comment.
9. Go back to the Program.cs file and delete the // TODO: comment in doWork().
10. Replace the code with:   
      
    **Point origin = new Point();**
11. Because the Point class has no content, it’s not currently usable so switch to the Point.cs file.
12. Replace the // TODO: comment with the following constructors:  
      
    **// Default Constructor  
    public Point()  
    {  
     Console.WriteLine(“Default Constructor Called”);  
    }**  
      
    **// Non-default constructor**  
    **public Point(int x, int y)  
    {  
     Console.WriteLine(“The point coordinates are x:{0}, y:{1}”, x, y);  
    }**
13. Go back to the Program.cs file and add the following code:  
      
    **Point bottomRight = new Point(1366, 768);**
14. Modify the Point class and add two private integer variables to represent x and y values.
15. In the non-default constructor, remove the Console.WriteLine code and assign the private member variables using the **this** keyword.
16. Create a public method in the Point class called **PrintCoords().**
17. In this new method, use a Console.WriteLine() method to output the values of the instance member variables using the **this** keyword for accessing them.
18. Run the application again and verify it works correctly. Fix any errors you encounter.
19. Modify your Point class to use a static variable called **objectCount**.  
      
    **private static int objectCount = 0;**
20. In the constructor, ensure that you increment the objectCount variable by one.
21. Create a new public method to return the object count:  
      
    **public static int ObjectCount()  
    {  
     return objectCount;  
    }**
22. In Program.cs, create two or three Point instances.
23. In the doWork() method, use a Console.WriteLine method to out the number of objects created by calling the static method on the Point class.
24. Choose Debug, Start Without Debugging or press CTRL + F5 to run the application.
25. Note the two different outputs.
26. Press Enter to quit the application.

## Creating Arrays of Classes

1. Open Visual Studio if it is not already open.
2. Navigate to where you installed the companion content.
3. In the Chapter 10 folder, locate either the Windows 7 or Windows 8 folder.
4. Open the Cards project.
5. Review the **Value.cs** file and note the enumeration for card values.
6. Review the **Suit.cs** file and note the enumeration for the card suits.
7. Review the **PlayingCard.cs** file and note the **PlayingCard** class file.
8. Open the **Pack.cs** file. This file contains a class to represent a pack of cards.
9. Note there are constants to represent the number of suits and the number of cards per suit in a pack of cards.
10. Note that it also has a 2-dimensional array of **PlayingCard** objects call **cardPack** and a random card selector.
11. Locate the //TODO comment in the Pack constructor **public Pack().**
12. Delete the comment and add your own code to instantiate a new instance of **cardPack** as a 2-dimension array.  
      
    **this.cardPack = new PlayingCard[NumSuits, CardsPerSuit];**
13. Create a nested for loop with the outer loop setting the suits and the inner loop setting the values. This code goes in the same location as step 12:  
      
    **for (Suit suit = Suit.Clubs; suit <= Suit.Spades; suit++)   
    {   
     for (Value value = Value.Two; value <= Value.Ace; value++)   
     {   
     this.cardPack[(int)suit, (int)value] = new PlayingCard(suit, value);   
     }  
     }**
14. Locate the **DealCardFromPack** method in the **Pack** class.
15. Delete the comment and NotImplementedException lines and replace them with the following line of code to select a random suit:  
      
    **Suit suit = (Suit)randomCardSelector.Next(NumSuits);**
16. Following this line of code, add this while loop:  
      
    **while (this.IsSuitEmpty(suit))   
    {   
     suit = (Suit)randomCardSelector.Next(NumSuits);   
    }**
17. Follow the while loop with this code to select a random card. Enter the following code after the while loop:  
      
    **Value value = (Value)randomCardSelector.Next(CardsPerSuit);   
    while (this.IsCardAlreadyDealt(suit, value))   
    {   
     value = (Value)randomCardSelector.Next(CardsPerSuit);   
    }**
18. Follow up in the same method with this code that returns a card:  
      
    **PlayingCard card = this.cardPack[(int)suit, (int)value];   
    this.cardPack[(int)suit, (int)value] = null;   
    return card;**
19. Locate the IsSuitEmpty method and enter the following code, which checks to see if there are any more cards of the provide suit left in the pack:  
      
    **bool result = true;   
    for (Value value = Value.Two; value <= Value.Ace; value++)   
    {   
     if (!IsCardAlreadyDealt(suit, value))   
     {   
     result = false;   
     break;   
     }   
     }   
    return result;**
20. Now we need to check if a card has already been dealt or not. Enter the following code in the IsCardAlreadyDealt method:  
      
    **return (this.cardPack[(int)suit, (int)value] == null);**
21. Once we have the dealt cards, we need to add them to the current hand. Open the review the code in the Hand.cs file. This file describes the Hand class.
22. Locate the AddCardToHand method and enter the following code:  
      
    **if (this.playingCardCount >= HandSize)   
    {   
     throw new ArgumentException("Too many cards");   
    }  
     this.cards[this.playingCardCount] = cardDealt;   
    this.playingCardCount++;**
23. Locate the dealClick method in the MainWindow.xaml.cs file and in the try block, add the following line of code:  
      
    **pack = new Pack();  
    for (int handNum = 0; handNum < numHands; handNum++)  
    {  
     hands[handNum] = new Hand();  
     for (int numCards = 0; numCards < Hand.HandSize; numCards++)   
     {  
     PlayingCard cardDealt = pack.DealCardFromPack(); hands[handNum].AddCardToHand(cardDealt);   
     }  
    }**

**north.Text = hands[0].ToString();   
 south.Text = hands[1].ToString();   
 east.Text = hands[2].ToString();   
 west.Text = hands[3].ToString();  
}**

1. Click Debug, Start Debugging or press F5 to run the application.
2. Fix any errors that may occur.
3. Test the application by clicking the Deal button. If you are using the Windows 8 version, you may have to right-click the app to bring up the Deal button.
4. If all goes as planned, close the application.
5. If you have any issues with the code, open the Cards Complete solution to see the finished product.
6. Select the File menu in Visual Studio.
7. Select Close Solution.
8. Close Visual Studio for now as this practice is complete.