Activity 5

Activity Overview

At this point, you will apply what you have learned about inheritance to a hands-on coding activity. We will take a class that is pre-built, extend it, and give it more specific features.

Activity Instructions

1. In this activity you are going to extend the existing ***PlayerProfile*** class to create the ***CardPlayer*** class under the namespace ***CSharp.Activity.CardGame***. The interface called ***ICard*** is already provided. You will learn more about interfaces in the next activities, so for now, focus on the ***CardPlayer*** class.

2. The ***CardPlayer*** class has similar characteristics and behavior as the basic ***PlayerProfile*** but with a distinct difference: it is specifically designed to have a hand of ***ICard*** objects and behavior specific to manipulating that hand.

3. Create the ***CardPlayer*** class and declare that it is a subclass of the ***PlayerProfile*** class. This automatically grants it access to any members of the parent class that are either public or protected.

4. Create a class named ***SimpleCard*** and inherit it from the ***ICard*** interface. You would be required to add the objects of this ***SimpleCard*** class to the ***CardPlayer*** (to be developed in Step 5 below). The following behaviors would help you complete this class:

   a. ***public int CardAttribute { get; set; }*** – Property to hold the attribute of the card. For testing we can assign numbers (like 1, 2, etc.,) to differentiate the cards.
   b. ***public string GetCardAttribute()*** – Function to return the card attribute concatenated with a string. This would suffice for our activity.

5. It will be up to you to determine how to implement the hand of cards in the class (you'll need some kind of data structure to store the cards as one of ***CardPlayer's*** attributes). The only restriction is that the implementation must be indexed; that is, the first card added is index 0, the next is 1 and so forth.

HINT: To represent the hand you can use an instance of the ***ArrayStore<T>*** class that you created in the previous activity.

6. Use ***Activity5_Tests.cs*** to check if the class you made complies with the specification.

7. The following behaviors must be followed by the ***CardPlayer*** class:
   a. ***public CardPlayer(string, char, DateTime, int)*** – constructor that creates a player with a specified name, gender, date of birth and the maximum number of cards that the player can have in his/her hand.

   b. ***public bool AddCard(ICard) –*** Adds the specified ***Card*** object into the player's hand. Returns ***true*** if the add is successful, and returns ***false*** otherwise. Throws a new ***ArgumentNullException*** if the argument is ***null***.

   c. ***public bool RemoveCard(ICard) –*** Removes the specified Card object from the player's hand that is ***Equal()*** to the ***Card*** specified in the argument. Returns ***true*** if the remove is successful and ***false*** otherwise. If the argument is ***null***, throw an ***ArgumentNullException***.

   d. ***public bool IsFull() –*** Returns ***true*** if the hand is full, ***false*** otherwise.

   e. ***public ICard GetCard(int) –*** Returns the card at the specified index but does not remove it. No validation on the index is expected from this method.

   f. ***public int CardCount –*** Property to return the number of cards on hand.

   g. ***public int MaxSize –*** Property to return the maximum number of cards the hand can hold.

8. By committing to the TFS with comment “Finished Activity 5” inform about completion of this activity.