Programming Assignment 3: Nothing Like Blackjack

Table of Contents

Assignment Description

Imagine you've decided to write a program to play a game that's nothing like Blackjack. You hope to get rich by selling your game to the casinos in Las Vegas, so you want to do a great job!

In this assignment, you'll deal one card to four players, deal another card to the same four players, then give one more card to a couple of the players. Finally, you'll flip over and print the cards for each player. You might as well start shopping for your new McLaren now ...

Hint: If you deal the cards properly, player 1 will end up with the King of Spades and the 9 of Spades.

Why do we care?

This assignment gives you valuable practice using classes and objects. Because we use object-oriented programming to write our Unity games (coming soon!), we need to understand how to use classes and objects.

Starting the Assignment – Clone Your Repository

1. Accept the assignment to create your repository for submitting your work:

   https://classroom.github.com/a/NCC_0mxP
2. In GitHub Desktop, clone the repository to your desktop.
3. Open the project in Visual Studio.
   a. **Important**: You MUST only add code as indicated by the comments in that file. If you don't, you're virtually guaranteed to fail all the test cases in the automated grader.

4. Double-click the index file in the Help folder and click the ConsoleCards link in the pane on the left; this is the documentation for the classes I included as a dll in the starter code. Those are the classes you'll use to you implement your solution.

Implementation Steps

5. Implement your solution by providing the required code under each comment in the **`Main`** method.
    a. Take Small Bites: The best way to approach this is to provide the required code under the first comment in the **`Main`** method, then test (and debug as necessary) that code to make sure it works. Next provide the required code under the second comment and test and debug it. Continue with this process until you're done.
    b. Doing it this way lets you focus on small pieces of the code at a time and also makes debugging a lot easier if something goes wrong.
    c. Commit your changes in GitHub Desktop after you have each step completed (and tested).

Submit Your Work

6. Make a final test of your code and copy the output from the terminal window.
7. If you need to make any additional changes to your code, make sure you commit them.
8. By committing and pushing your updates to GitHub you have submitted your assignment on GitHub Classroom.
9. Return to CodeHS. Paste your output into the code window to complete the assignment.

Helpful Hint:

One of the things your code has to do is deal a card to a player. It helps most people to think about how this actually happens if you have a physical deck of cards: in that case, you take the top card from the deck and you place it in a location in front of the player. Since the Deck class exposes a TakeTopCard method and you can have as many Card variables (named memory locations) as you want, you should be able to see how to implement that functionality in your solution.

Running Your Code

Because of the code I included to work with the automated grader on Coursera, when you run your program the command prompt window will open and it will sit there doing nothing. To make your code run, type 1 and press the <Enter> key; your code should then run so you can check your output.

You can actually run your code again if you want to by typing 1 and pressing the <Enter> key again. When you're ready to stop running your code, type q (for quit).

Required Output Format

Each line of your output is the information for a specific card. The comments at the end of the **`Main`** method have you print all the cards for player 1, then all the cards for player 2, and so on. The required format for printing out a card is the card rank, followed by a comma, followed by the card suit. For example, the King of Spades would be printed as:

```
King,Spades
```

Note that there are no spaces in the line above; it's just the card rank, a comma, and the card suit.

Test Case Inputs

The autograder just runs your code once, providing 1 then q as the inputs to your code. You should figure out what the expected results are for correct behavior and make sure your code is generating those expected results in the required format before submitting your code for grading.