Programming Assignment: Nothing Like Blackjack

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 two cards to four players (including the dealer), 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 ...

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

Download the appropriate zip file for your OS below and unzip the file somewhere on your computer.

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.

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 ProgrammingAssignment2 solution. Those are the classes you'll use to you implement your solution.

Implementation Steps

Implement your solution by providing the required code under each comment in the `Main` method.

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.

Taking 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.

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.

Common Problems

Historically, a number of people post about grading errors on this assignment (because their code is wrong). Here are a couple things you should check:

1. You have to start with the Visual Studio project I gave you in the assignment materials and add your code in the section indicated. The code I gave you includes the appropriate structure for the automated grader to work. If you don't do this, you'll almost definitely fail all the test cases in the automated grader.
2. Be sure your code generates the correct output in the correct format exactly as shown above. If it's not, your code isn't working properly

Submitting Your Solution

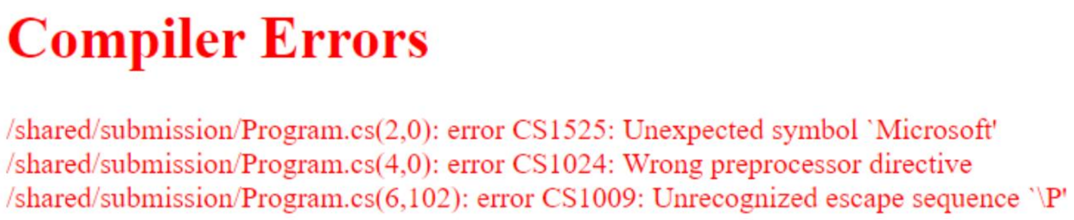
Go to the programming assignment on Coursera and click My submission near the top (next to Instructions). Click the + Create submission button, click the blue button on the left, upload your Program.cs file, and click the Submit button. If the Honor Code popup appears, click the check box, fill in your name if necessary, and click the Continue button. Then wait patiently while the automated grader grades your assignment!

What Is The Automated Grader Telling Me?

If you post about getting errors from the automated grader, be sure to include a screenshot of the grader feedback. You can get to that feedback by clicking the Show grader output link in the Passed? column, then clicking the View grader feedback link that appears when you do that. You'll know you're in the right place when all the characters are either green or red.

What If I Upload the Wrong File?

If you upload the wrong file, you'll almost definitely get compilation errors. The most common mistake people make is uploading the .sln file; if you do that, you'll get something like the following:



Compiler Errors

```
/shared/submission/Program.cs(2,0): error CS1525: Unexpected symbol `Microsoft'
/shared/submission/Program.cs(4,0): error CS1024: Wrong preprocessor directive
/shared/submission/Program.cs(6,102): error CS1009: Unrecognized escape sequence `\P'
```

If that happens, go back and upload the correct file.

Which Posts Will Dr. T Respond To?

I'm happy to try to help people solve problems they're having, but I'll only do that if you provide the appropriate information in your initial post. If you're getting a grader error, you have to provide both the screenshot from the previous section and a screenshot of your code running multiple times (as shown above) on your local computer, including the Press any key message at the end. That Press any key message is important, so be sure to include it