Lab04: Classes Due: Oct 15, 2018 at 3:30pm

The goal of this lab assignment is for you to practice and become comfortable with creating and maintaining classes that support multiple modules. Once again, careful design and planning will save you lots of coding and debugging time down the road. You will write one program with two supporting class modules for this lab that should be 100% GUI-based. (Remember to adhere to the GUI guidelines specified in Lab03.)

You will write a program (called Lab04_Blackjack.py) that lets the user play blackjack. The rules of blackjack for this program are as follows:

1. Each round starts with the user/player being dealt two cards and the computer/dealer likewise getting two cards. One of the dealer cards is visible to the player - the other is unseen.

2. All cards are worth their face value (ie a 9 is worth 9) with the exception of face cards (kings, queens, and jacks, which are all worth 10) and aces (which can be worth 1 or 11, player's choice)

3. The goal is to get your cards as close to (or exactly equal to) 21 without going over.

4. After the deal, the user goes first. They may ask for an additional card ("hit") or stay where they are ("stand").

5. If the user hits, they are dealt another card and, if not over 21 ("bust"), has again the option of hitting or standing. This repeats until the user decides to stand or goes bust.

6. If the user busts, the round is over and the dealer wins. If the user stands, it is the dealer's turn but, by rule, he does not have any real options. He must keep hitting while he has 16 or less. He must stand once he gets to 17 (or more). He would obviously stop if he goes bust.

7. If the dealer busts, the player wins. If he does not bust, then the higher of the two point values wins. If a tie, no one wins (called a "push"). For an ace, assume its value is 11 unless that would bust a player. If the dealer has 17 with an ace counting as 11, he still must stand.

8. A message box should be always on screen with messages that persist, explaining who won and why.

To implement this program you need to create separate module called Button.py that will contain a Button class that you will use for this program. Be careful that your Button class is generic in its functionality so you can reuse it in later programs.

You will also create a third module called PlayingCard.py that will contain a PlayingCard class with all important info including how to draw itself on a window. All cards will be shown as an upright rectangle (with white background) with the rank (A-K) and suit (hearts, spades, etc.) shown in its middle. In addition, if it is a red suit (hearts/diamonds), then the rank and suit should be 100% red. Spades and Clubs should be all black. You must create your own suit drawings.

Other specs for your Blackjack game:

- You will start the program by opening a small window with an Entry object, asking for the user's name. Once entered, the user will click a "Start" button which will close this small dialog and open the main game window.
- The main game window should have a set of buttons for Hit, Stand, Play Again, and Quit. These should all be enabled/disabled as appropriate (with the exception of the Quit button which should always be active). The locations of these buttons is up to you but please place them in somewhat logical/practical locations.
- Your program will capture repeated rounds of this game, displaying the cards and keeping a displayed score of who wins. To do this, there should be a modest scoreboard somewhere on the window that tracks the overall score in terms of rounds won. Please use the user's name here (not just "user") and "Dealer."
- The user's cards should appear along the top of the screen, starting from left to right. Two cards would initially appear on the far left and then continue to be added at an equal spacing to the right as the user chooses to "hit." Leave room for six cards. If the user takes a 6th card and is not bust, disable the Hit button and only allow the user to stand at that point (or Quit). I know this is not the rule, just deal with it.
- The dealer's cards should appear along the bottom of the screen in the same manner. The initially hidden card should look like the other cards, but with just a large question mark on it.
- If the user busts, there should be a significant message on the screen that they have lost. If not, immediately "flip" the dealer's hidden card and play out his hand (this will seem instantaneous, don't worry).
- All results from this point (dealer bust, user wins on points, dealer wins on points, or push) should be indicated to the user and the overall score adjusted accordingly. In addition, if neither side busts then you should show the point comparison (ie "Dealer wins 20-19," "Richie wins 18-17," or "Richie and Dealer push 20-20").
- Randomly generating card ranks and suits: use the `randrange` function from the random library (import random.py) to get random values. Don't worry about getting duplicate cards. To get an integer value between 3 and 19 inclusive, you would have a command such as `x = randrange(3,20)` (just like the range function).
- Note that extra methods/functions are only allowed within the class definition in the PlayingCard class.

As always, I am expecting significant comments for all submitted files. The GUI should be so clear that I could pull a random high school student (who knows the rules of blackjack) from the cafeteria and have them go through the program without explanation (not a bad test, to be honest). This lab is worth 8 points for proper execution and adherence to the specs, 1 point for comments, and 1 point for aesthetics and design. I will deduct points if a program: does not follow proper encapsulation techniques, creates multiple object variables when not necessary, is not user-friendly as described, and does not work to the specifications (or at all).

TOTAL: 10 pts