CS 1110, LAB 11: CLASSES: BLACKJACK

http://www.cs.cornell.edu/courses/cs1110/2018sp/labs/lab11/lab11.pdf

First Name: _____ **Last Name**: _____ **NetID**: _____

Getting Credit: **As always, strive to finish during the lab session** — it's the best way to stay on track in this course.[1]

1. Setup

Create a new directory on your hard drive for this lab's files. Then, download into it the files for lab 11 from http://www.cs.cornell.edu/courses/cs1110/2018sp/labs .

2. The (CS1110 version of the) Card Game Blackjack

In this lab, you will complete the skeleton for the definition of a class Blackjack that a casino could use to run multiple blackjack games simultaneously.

A player wins at blackjack by ending with a hand that has more points than the dealer's, but not more than 21 — if a player exceeds 21 points, they have "gone bust" and lose immediately. Points come from the ranks of the cards in a hand: 10 points for each face card (Jack, Queen, or King), 11 points for an ace, and the rank of the card for anything else (e.g., a 4 of anything is 4 points).[2]

Play begins with two cards being dealt to the player and one to the dealer. All cards in each hand are visible to all participants. The player can chose to "hit" (get an additional card from the deck) or "stay" (turn over play to the dealer). Once a player stays, if they haven't gone bust, then the dealer draws cards until they go bust or decide to themselves stay.

Our dealer is following a common protocol: they continue to hit while their hand is under 17, but once their hand reaches 17 or more, they stay (if they haven't already gone bust). See if you can use this to your advantage.

The next page shows a sample transcript from our solution code.

Lab authors: D. Gries, L. Lee, S. Marschner, W. White

[1] But if you don't manage finish during lab, here are the alternate checkoff opportunities: (a) at ACCEL Green room consulting hours, listed at http://www.cs.cornell.edu/courses/cs1110/2018sp/about/staff.php , **from today until Tue Apr 17 inclusive**, (b) at non-professorial TA office hours **from today to Wed Apr 18 3:45pm inclusive**, although at TA office hours, questions about course material or assignments take precedence over lab check-offs; or (c) during the **first 10 minutes of your next scheduled lab (Tue Apr 17 or Wed Apr 18)**. Beyond that time, the staff have been instructed not to give you credit.
Labs are graded on effort, not correctness. We just want to see that you tried all the exercises, and to clarify any misunderstandings or questions you have.

[2] In some versions of blackjack, an ace can be worth either 1 or 11, whichever is better; we do *not* allow this in our implementation.

```
[llee: lab11] python blackjack.py
Welcome to CS 1110 Blackjack.
Rules: Face cards are 10 points. Aces are 11 points.
       All other cards have face value.

Your hand: 4 of Diamonds, 5 of Clubs
Dealer's hand: 2 of Spades

Type h for new card, s to stop: h
You drew the 8 of Clubs

Type h for new card, s to stop: h
You drew the 2 of Diamonds

Type h for new card, s to stop: s

Dealer drew the 3 of Spades
Dealer drew the 8 of Spades
Dealer drew the 10 of Hearts

Dealer went bust, you win!
The final scores were player: 19; dealer: 23
```

3. Completing the Blackjack class definition

For each implementation subsection below,

(1) Read the directions in this handout and the specification in the blackjack.py file.
(2) Look at the appropriate test cases in blackjack_checks.py, to better understand the usage and goals of the code you will write.
(3) Replace the "pass" lines with your implementation. If you need a model, you can take a look at the methods in class Card in card_lab11.py.
(4) Check your code by entering either python blackjack_checks.py or python blackjack_checks.py quiet, depending on how much output you want and how far along you are in the lab.

Make sure you've checked your work for a given subsection *before* moving on to the next one. This is important because many of the methods here build on earlier ones.

3.1. **Alteration of the Card class to make the test cases more readable.** You actually don't need to look at the Card class to complete this lab, but you may notice in the test cases in blackjack_checks.py that Card objects are sometimes being created with a new kind of expression. An example is (assuming you've imported the module stored in card_lab11.py as s11)

```
c = Card(alt='AH')
```

What we've done is altered the __init__ function for class Card so that it can take a 2-character string as a value for an *optional* parameter alt; the idea is that the test cases will be more readable this way. The first character indicates the rank: 'A' for Ace, '2' for 2, . . . , '9' for 9, 'T' for 10, 'J' for Jack, 'Q' for Queen, and 'K' for King. The second character indicates the suit: 'C' for Clubs, 'D' for Diamonds, 'H' for Hearts, and 'S' for Spades.

3.2. **Implement and test `__init__`.** Implement `__init__` so that it initializes the three instance attributes of `Blackjack` in the manner described in the specification. You'll probably want to use some list methods, which are described in section 5.1 of the Python library. Our solution is three lines long. Write yours here:

3.3. **Read over but don't change helper function `_score`.** The leading underscore in this function's name indicates that it is meant to be a private helper function.

Note that it is a function that is *not* a method. Given the specification of `_score`, why does it make sense that this function is not a method of class Blackjack?

3.4. **Implement `dealerScore()` and `playerScore()`.** Make use of function `_score`. Write your code for `dealerScore` here (ours is a single line):

3.5. **Implement and test `playerBust()` and `dealerBust()`.** Your implementation should use `dealerScore()` and `playerScore()` as helper methods. Write your code for `playerBust()` here:

3.6. **Implement and test `__str__`.** . Note that this method is "higher up" in the file, just after `__init__`, as is conventional. Use `dealerScore()` and `playerScore()` as helper methods.

Write your code here:

3.7. **Play some Blackjack!** Run `blackjack.py`:

```
python blackjack.py
```

Follow the directions on the screen. The command 'h' is for 'hit', and 's' is for stay.

4. Facilitating checking-off

Here's a checklist to be ready to quickly demonstrate your work to a staff member.

- ☐ You have a copy of this handout with all the white boxes filled in.
- ☐ You are ready to show that your code passes the given test cases by running `python blackjack_checks.py quiet` in the command shell.
- ☐ You are ready to show that you can play the blackjack game by running `python blackjack.py` in the command shell.