CS 0008
University of Pittsburgh
Project 3

Overview

For your final project, you will create a realistic game of Blackjack. The goal of the game is to build a hand of cards that values as close to 21 as possible without exceeding it.

First, here is the terminology used in the game.

- Hit: Take another card for your hand
- Stand: Stop taking cards
- Bust: Exceed a value of 21 in your hand
- Push: Tie with the dealer

The cards are valued as follows:

2-10 Face value (ex: 2 of hearts has a value of 2, 5 of clubs has a value of 5)

Jack, Queen, or King 10

Ace Either 11 or 1

Of particular note, an Ace can be considered either value 11 or 1. Given rational strategy, it will be considered 11 unless that will cause a bust, in which case it will be considered 1. In each hand, play proceeds as follows.

1. The dealer takes one card, which the player can see.
2. The player is given two cards.
3. The player has the option to hit (take one additional card). The player can hit as many times as desired, until they reach or exceed 21, or decide to stand (stop taking cards).
4. If the player did not bust (exceed 21), the dealer takes cards until she reaches 17 or higher (i.e., the dealer will hit on 16 or fewer points, and stand on 17 or more points).
5. If the player busts, the dealer immediately wins, and the dealer will not draw more cards. If the dealer busts, the player wins. If the player and dealer have the same total, it is a push. Otherwise, the winner is the one with higher point total.
6. If the player loses, the player also loses their bet. If the player wins, they receive the bet. If the hand is a push, the bet is returned (i.e., no gain or loss).

You will implement this game by first creating classes to represent a playing card and a deck of cards. Your program should be realistic in that it should consider playing each hand with a new deck of 52 cards (making it impossible to draw 5 Aces in a single hand, for example).

Once you have classes to represent cards and decks, you will move on to using these classes to create a Blackjack game.

Note that this project is quite underspecified compared to previous assignments. There are several questions on *how* to implement your game that you will need to decide how best to handle.

1 Card and Deck classes

For the first part of this project, you will create two classes to represent a playing card and a deck of playing cards. Minimally, you will need initializer methods for each of these classes, ways to access the attributes of the cards (i.e., number and suit), a method to draw a card from the deck, and a way to shuffle the deck. Other questions to consider:

- Could either class use a str() method. How could this be helpful?
- How should you represent the group of cards in the deck?

In addition to creating these two basic classes, you should also create a *driver* program to test that they are working. For example, you driver program could be called "deck tester.py, and could create a new instance of your deck class, shuffle the deck, draw three cards from the deck, and print the values of those cards to the screen. By keeping the driver program simple, you will be able to more easily debug your card and deck classes before writing the whole Blackjack game.

2 Blackjack Game

For the second part of the project, you will use your completed card and deck classes to create a realistic Blackjack game. Your game should start by allowing the user to enter their name, and then assign them $1,000 to bet with. After this, the game should proceed as described above until the user either decides to quit or has no money to bet.

Questions to consider in implementing this game:

- How should you represent the collection of cards stored in the user's/dealer's hand?
- Could you benefit from the use of classes to represent the user and the dealer? How would you design these classes? Or would it be easier to forgo object oriented design for this part of the project?
- Where/how should you handle the value of an Ace (given that it can be either a 1 or an 11)?

Whichever approach you take, your game will need to be carefully designed. Take some time to write/draw out the flow of your game and figure out what methods and classes you will need in order to implement your game.

As an example, a run of your game should proceed as follows:

```
Name? John
John has $1,000
Bet? (0 to quit, Enter to stay at $25) 100

Bet: $100
Dealer’s hand: 10 (hearts)
Value: 10
John’s hand: J (spades) 2 (clubs)
Value: 12

Move? (hit/stay) h

Bet: $100
Dealer’s hand: 10 (hearts)
Value: 10
John’s hand: J (spades) 2 (clubs) 3 (clubs)
Value: 15

Move? (hit/stay) h

Bet: $100
Dealer’s hand: 10 (hearts)
Value: 10
John’s hand: J (spades) 2 (clubs) 3 (clubs) 7 (spades)
Value: 22
John bust

John has $900

Bet? (0 to quit, Enter to stay at $100)

Bet: $100
Dealer’s hand: 10 (diamonds)
Value: 10
John’s hand: 4 (hearts) 2 (clubs)
Value: 6

Move? (hit/stay) h

Bet: $100
Dealer’s hand: 10 (diamonds)
Value: 10
John’s hand: 4 (hearts) 2 (clubs) 3 (clubs)
Value: 9

Move? (hit/stay) h

Bet: $100
Dealer’s hand: 10 (diamonds)
Value: 10
John’s hand: 4 (hearts) 2 (clubs) 3 (clubs) 9 (clubs)
Value: 18
```

```
Move? (hit/stay) h

Bet: $100
Dealer’s hand: 10 (diamonds)
Value: 10
John’s hand: 4 (hearts) 2 (clubs) 3 (clubs) 9 (clubs) A (hearts)
Value: 19

Move? (hit/stay) s

Bet: $100
Dealer’s hand: 10 (diamonds) 4 (clubs)
Value: 14
John’s hand: 4 (hearts) 2 (clubs) 3 (clubs) 9 (clubs) A (hearts)
Value: 19

Bet: $100
Dealer’s hand: 10 (diamonds) 4 (clubs) K (spades)
Value: 24
John’s hand: 4 (hearts) 2 (clubs) 3 (clubs) 9 (clubs) A (hearts)
Value: 19

Dealer bust

John has $1100

Bet? (0 to quit, Enter to stay at $100)

Bet: $100
Dealer’s hand: A (clubs)
Value: 11
John’s hand: 10 (clubs) 7 (clubs)
Value: 17

Move? (hit/stay) h

Bet: $100
Dealer’s hand: A (clubs)
Value: 11
John’s hand: 10 (clubs) 7 (clubs) 7 (diamonds)
Value: 24

John bust

John has $1000

Bet? (0 to quit, Enter to stay at $100)

Bet: $100
Dealer’s hand: 8 (spades)
Value: 8
John’s hand: 6 (clubs) K (hearts)
```

```
Value: 16

Move? (hit/stay) h

Bet: $100
Dealer’s hand: 8 (spades)
Value: 8
John’s hand: 6 (clubs) K (hearts) A (spades)
Value: 17
Move? (hit/stay) s

Bet: $100
Dealer’s hand: 8 (spades) 5 (spades)
Value: 13
John’s hand: 6 (clubs) K (hearts) A (spades)
Value: 17

Bet: $100
Dealer’s hand: 8 (spades) 5 (spades) 8 (diamonds)
Value: 21
John’s hand: 6 (clubs) K (hearts) A (spades)
Value: 17

Dealer wins

John has $900

Bet? (0 to quit, Enter to stay at $100)

Bet: $100
Dealer’s hand: 4 (diamonds)
Value: 4
John’s hand: 2 (diamonds) J (hearts)
Value: 12

Move? (hit/stay) s

Bet: $100
Dealer’s hand: 4 (diamonds) A (hearts)
Value: 15
John’s hand: 2 (diamonds) J (hearts)
Value: 12

Bet: $100
Dealer’s hand: 4 (diamonds) A (hearts) A (diamonds)
Value: 16
John’s hand: 2 (diamonds) J (hearts)
Value: 12

Bet: $100
Dealer’s hand: 4 (diamonds) A (hearts) A (diamonds) 10 (hearts)
Value: 16
```

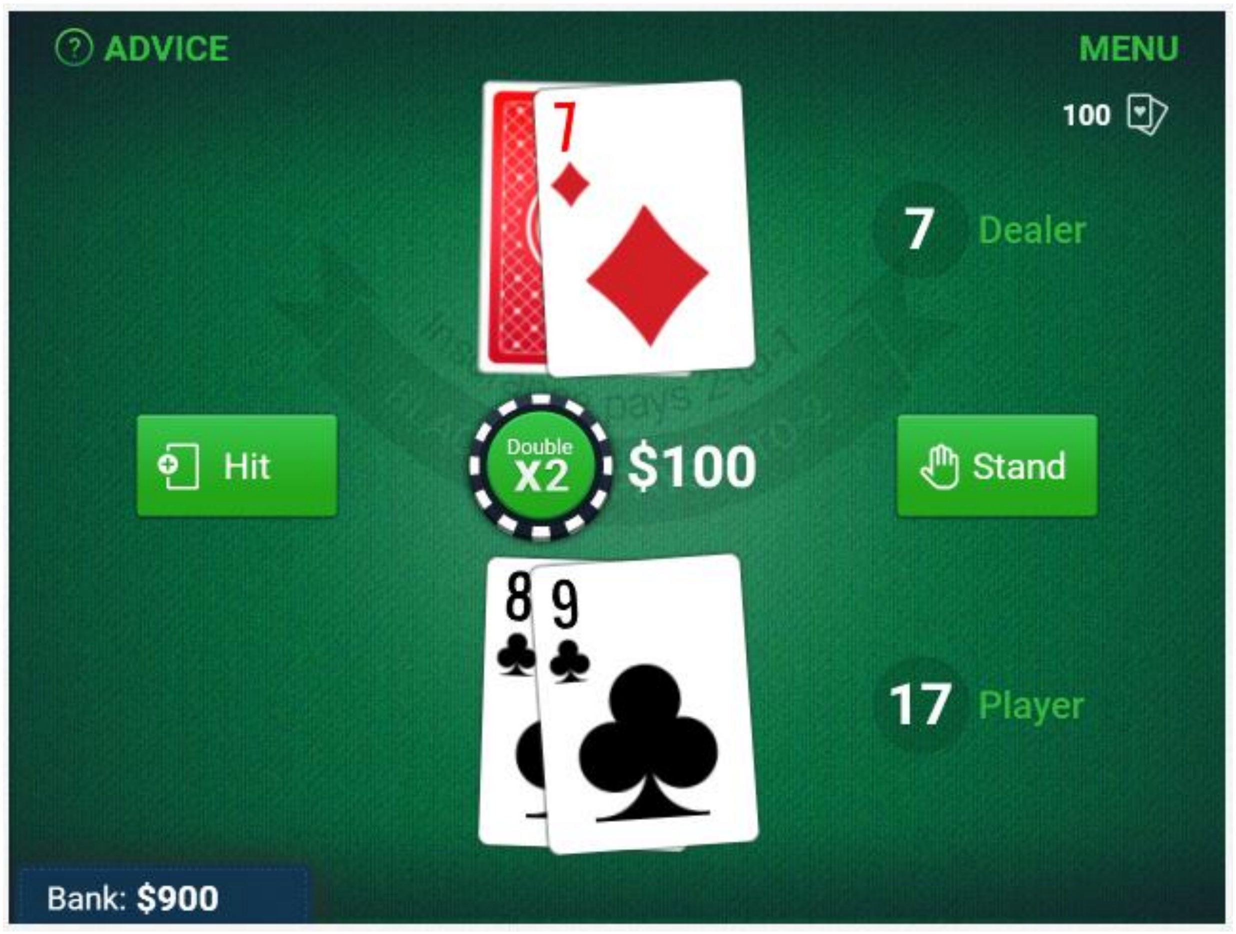
```
John's hand: 2 (diamonds) J (hearts)
Value: 12

Bet: $100
Dealer's hand: 4 (diamonds) A (hearts) A (diamonds) 10 (hearts) 8 (diamonds)
Value: 24
John's hand: 2 (diamonds) J (hearts)
Value: 12

Dealer bust

John has $1100

Bet? (0 to quit, Enter to stay at $100) 0
```

3 Bonus Points

You can earn up to 20 bonus points by implementing a graphical version of the assignment. This means that you must create a completely Graphical User Interface (GUI) which the player will use to play the game. However, the game must still have all the features discussed in Part 1 and Part 2. In the class so far (and in this project), we have been using a text based approach which required us to type in everything when running our programs but graphical interfaces typically have buttons, images, and even sound effects to convey what is happening to the user. I suggest you reference Python's graphical modules and libraries to create a graphical version of BlackJack. You can find information about Python's graphical modules on the Chapter 13 slides (which are posted on Courseweb) and from various sources online. Because this is optional and it is for up to 20 Bonus Points, I will not offer that much assistance to help you with implementing the graphical interface.

As an example, there is a screenshot below of a blackjack game I found online which you can use to get an idea of what a graphical user interface looks like (although you have seen graphical user interfaces before but probably didn't know what they were called). Do not worry about the Double X2 in the middle of the screenshot, that is not something you need to implement. However, notice the user's bet is shown in the middle and there is one button on the left to Hit and one on the right to Stand. The value of each hand is also shown. The total money that the user has left is shown in the bottom left corner. I want you to be creative with how you approach this so your program does not have to look exactly like the image below. You can make it simple or you can put in more effort and make it more colorful and interactive. The number of bonus points you receive will be at my discretion and will be related to the amount of effort you put into this section. Less effort will result in less bonus points and more effort will result in more bonus points. Therefore, feel free to add in animations, music, and sound effects if you would like. However, remember that your game should still have all the features and act as discussed in Part 1 and Part 2.

For those interested, I posted the link below of where I got the screenshot from. By going to the website, it might help you understand the rules of Blackjack if you are not familiar but if there are differences between Part 1 and Part 2 and what is done in the online game, follow what is written in Part 1 and Part 2.



Source of the screenshot: http://www.arkadium.com/games/blackjack/.