Blackjack Game using Object Oriented Programming Methodology and Python

Harshad Naik
Computer and IT Department
VJTI
Mumbai-19, India.

ABSTRACT
In this paper, it has been described how to implement the Blackjack game using Object Oriented Programming in the programming language Python. Object oriented programming is a programming technique which involves dividing the program into classes which have attributes (data) and methods (functions). Objects are basically instances of classes. By using Object Oriented Programming one can get several benefits such as ease of maintainability and code reuse.

General Terms
Blackjack, Python, Object Oriented Principles, Classes and Objects.

Keywords
Blackjack, Python, Object Oriented Principles.

1. INTRODUCTION
Blackjack is a popular card game that is played in casinos. There are two main players. The two players are as follows, the Dealer who represents the casino and the player who is playing. Initially both the dealer and the player are dealt two cards each. Both cards of the player are kept face up. One card of the dealer is kept face up and the other is kept face down. The objective of this game is to get a score equal to 21 or as close to 21 as possible without exceeding 21.King, Queen and jack have a value of 10, Ace can have a value of 11 or 1. Other cards have value as per their rank. The Player has two options. He can hit and take another card or he can stand and stop taking cards. If the player crosses 21, he is busted and he loses. If the player stands, dealer starts drawing cards till he reaches 17 or crosses 17.Whoever has higher score will win. Getting a score of 21 will automatically invoke stand. The Player cannot lose if he reaches 21 (21 is called Blackjack). He can only tie or win once his score is 21. In general, the player can win in two scenarios. First, if his score is greater than that of the Dealer. Second if the Dealer busts. There are two types of scores. For example if one is dealt an ace and a four, then the score can be a 'soft' 5 or it can be 15 depending on how we count an ace (Ace can have a value of 1 or 11). This is the most challenging part of implementing blackjack, since ace can be 1 or 11.

2. DESIGNING CLASSES
The code has been organized into 5 classes:

- Card
- Hand
- Deck
- Play
- CardLabel.

The classes have been described below.

2.1 Card
This class is used to represent individual cards.

It has attributes as follows:

- Rank: Tells value of the card. Example: 2, 3, A, K etc.
- Suit: Tells value of Suit. Example: Clubs, Spade, Diamond or Hearts
- Id: Useful for displaying the images of cards.

It has following methods:

- Get_rank(self): Tells us the rank of the card.
- Get_suit(self): Tells us the suit of the card.

2.2 Hand
This is used to represent the hand of the dealer or player. That is, it represents the cards that the player or dealer currently owns.

It has attributes as follows:

- Cards[]: An array to store cards

It has following methods:

- add_card(self,card): Used to add a card to the hand.
- get_value(self): Tells us the score of the hand
- soft_value(self): Tells us the soft value of the hand.

2.3 Deck
Deck represents the pack of 52 cards. It is used to deal cards to the dealer and player.

It has the following attributes:

- Cards[]:
- An array which stores cards of all suits and ranks at the start.

It has following methods:

- Shuffle(self): A method which shuffles the cards in the deck using random method of python.
- Deal_card(self): A method which pops card from deck and returns the card.

2.4 Play
This class is responsible for handling the flow of the game.

It has the following attributes:

- myDeck: An instance of class deck.
- hPlayer: An instance of Hand. Represents the player's hand.
- hDealer: An instance of Dealer. Represents the dealer's hand.
- in_play: Variable used to find if game is on or if it's finished. Set to false when player/dealer busts or once game is over.

It has the following methods:

- Hit(self): Used when the player wants to Hit. It adds a card to the player's deck. It also checks if player/dealer busted while invoking hit. Also if player reaches 21, it displays message of blackjack
- and invokes stand. If player busts, it increments losses by 1.
- Stand(self): It keeps adding cards to the dealers hand until it reaches 17. Depending on scores of the player and dealer, it displays appropriate message and also updates the statistics regarding win, losses and ties.
- check_black_jack(self): checks if the player has a blackjack. Useful when the game has just begun, it is called to check if player has a blackjack with the two cards he has received.
- Restart(self): Used to start a new game.

2.5 CardLabel

This class is responsible for fetching the images and displaying them.

It has the following attributes:

- images[]: stores all images of cards that have been imported using Photo Image Library.

It has the following methods:

- load_images(): static method which uses PhotoImage library to load the images from given directory to the array called images[].
- Display(self,side,id): Used to display the image. Side can be front or back. Id is used to access a specific index from the array images[].

2.6 Other Functions

Other functions include display() which invokes display of CardLabel to display the cards of the dealer and the player. Instructions() displays the instructions of the game to the user.

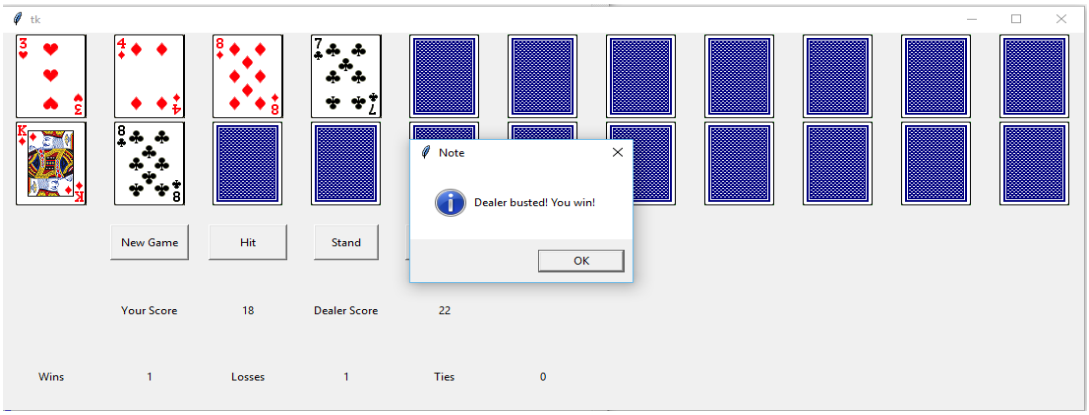

Fig 1: The Graphical interface of blackjack game designed by me in tkinter.

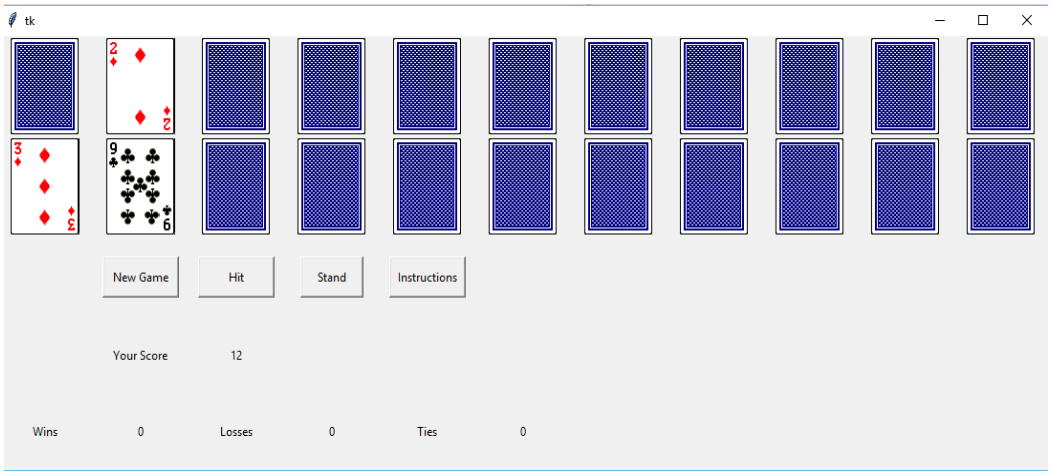

Fig2: Figure shows the game when it's just started.

Fig3. Figure shows hard and soft scores of the player's hand.

3. DESIGNING THE GUI

The GUI is designed using tkinter and PhotoImage library of python. In this case the grid functionality of tkinter is used to divide the graphical interface into rows and columns as shown in figure 2, 4 buttons have been provided which allow the user to start a new game, Hit, Stand and Instructions. New game, Hit and Stand invoke methods restart() ,hit() and stand() of an instance of class play. Instructions button invokes the function instructions(). As can be seen above, there is a provision to display up to 11 cards. The reason for this being the maximum number of cards that are required to reach 21 or beyond is 11 cards. This has been proved below.

3.1 Reason for providing provision to display only 11 cards for Player/Dealer:

Assume a scenario in which the first 4 cards that the player/dealer receives are all aces. This would give him a soft score of 4 and a hard score of 14. Now let the next 4 cards he receive be 2's (Since 2 has the least score after an ace, if ace is counted as 1). This gives the player/dealer a score of 12. Now all aces and 2's in the deck are exhausted. Now lets us assume that the player/dealer gets two 3's as his next cards. This takes his score to 18. Now if he gets 3, he will reach blackjack and it will lead to stand automatically. If he gets any other card then he will bust. Thus in the worst case the total number of cards needed to reach blackjack or to bust is 4+4+2+1=11. Hence a player can draw 11 cards to the maximum possible extent.

4. ALGORITHM

The algorithm for this program is pretty simple. The main function creates an instance of play. That instance contains myDeck, hPlayer, hDealer (instances of class Deck, Hand and Hand respectively).

1) In the init() function of play, myDeck.shuffle() is called which shuffles the cards using the random function of python.

2) Using myDeck.dealCard(), deal 2 cards to the player and Dealer.

3) Display function can be called to display cards. We check if Player has reached blackjack. If yes, then invoke the method stand(). If no then wait for the next call.

4) If the player clicks hit we invoke the method hit(). Another card is added to the player's hand. We check if Player has busted (that is his score is more than 21. We call get_value() method to find the score.) If score is 21, player has blackjack and we invoke the method stand() is invoked.

5) If the Player invokes the method stand() then cards are added to the hand of the dealer until the dealers score reaches 17 or above. Then compare the scores of the Player and the Dealer and declare the winner.

6) The display() function is called each time hit() or stand() is invoked.

4.1 Counting ace as one or eleven

This is taken care of in the get_value() function which returns the hard score of the player. The Algorithm for this is as follows:

1) Initially all aces are counted as 11. Also count the number of aces and store it in a variable called no_aces.

2) While the score exceed 21 and no_aces is greater than 0 then subtract 10 from the score and decrement the value of no_aces by 1.

3) This ensures that the dealer and the player are not disadvantaged when they get an ace.

In the function soft_value(), the value of ace is always counted as 1. soft_value() gives the player an idea as to how much squeezing room he has.

5. CONCLUSIONS AND FUTURE SCOPE

5.1 Conclusion

It has been explained how the game of blackjack can be implemented in python using Object Oriented Programming and tkinter for making the user interface. Object Oriented Programming makes it simpler to write functions as the logic needed to write it is simplified. It also helps in code reuse and

maintainability. For example, these classes of card, deck and hand can be reused to implement a card game like Rummy.

5.2 Future Scope

The function of split and double in the blackjack game have not been implemented in this game. Those functions can be implemented and added to this game. Also if actual money is being used, then one can add the functionality of placing and increasing bets. One can also use these classes and change only the display() function and port the game to pygame. Pygame can be used to provide an even more attractive GUI. Perhaps sound effects can be added to make the game for immersive.

6. REFERENCES

[1] Article from Business Insider India dated June 25, 2014:http://www.businessinsider.in/You-Better-Know-These-Basic-Rules-Before-You-Even-Think-Of-Playing-Blackjack/articleshow/37197646.cms.

[2] Paul Gries, Jennifer Campbell and Jason Montojo. "Practical Programming Second Edition An Introduction to Computer Science Using Python 3". (2014).

[3] Timothy C Lethbridge and Robert Laganiere "Object-Oriented Software Engineering". (2004).

[4] Webpage on rules of Blackjack by website wizardsofodds.com:http://wizardofodds.com/games/blackjack/basics/

[5] Webpage how to play blackjack by the website http://entertainment.howstuffworks.com/how-to-play-blackjack.htm

[6] K. Lieberherr, I. IIolland, A. Riel, 1988, Object-Oriented Programming: An Objective Sense of Style.

[7] Bjarne Stroustrup, What is ''Object-Oriented Programming''? (1991 revised version).