January 12, 2016

POOCasino - BlackJack

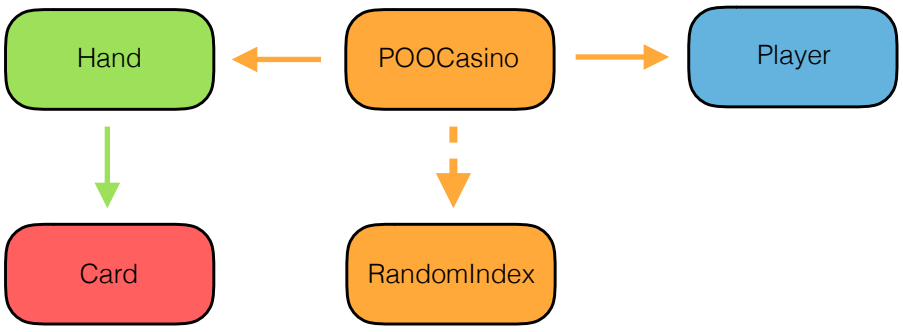
1. The player strategy I implemented

I follow and slightly modify the table on: http://wizardofodds.com/games/blackjack/basics/, since the casinos in our class have different rules.

Player's Hand	Dealer's Up Card	
Hard 4~8	Hit	Hit
Hard 9	Double /Hit	Hit
Hard 10/11	Double if player's hand > dealer's card /Hit	
Hard 12~16	Stand	Hit
Hard 17~21	Stand	Stand
Soft 13~15	Hit	Hit
Soft 16~18	Double /Hit	Hit
Soft 19~21	Stand	Stand
22/33/66/77/99	Split	Split
88/AA	Split	Split

My player is a true man (Chen-Nan-Jen), he never surrenders or buys insurance. In order to not lose too much money, he only bet 1 chip at a time.

2. My design of class structures



POOCasino class:

(a). **Main** function

```
public static void main(String argv[]){
    POOCasino casinoComputer = new POOCasino(argv);
    casinoComputer.startGame();
    while(casinoComputer.isGameOver()!=true){
        casinoComputer.playOneRound();
    }
}
```

(b). **POOCasino** constructor**:** Get each player by class name, give each player chips and set the round of game.

(c). **startGame** method: Say hello to player, set up player's name and position.

(d). **playOneRound** method**:** Clearly express how a game runs in a round

(e). **makeBets** method**:** Ask players to make bets. Exceptions happened when a player have chips < 0, I hold this exception by not letting this player to play(do any action) anymore, which means the game will continue.

```
void playOneRound(){
    round++;
    makeBets();
    firstTwoCards();
    buyInsurance();
    surrender();
    split();
    doubleDown();
    playerHit();
    dealerAction();
    moneyTime();
}
```

(f). **firstTwoCards** method**:** Give each player & dealer 2 cards, one face-up & one face-down.

(i). **buyInsurance** method: Ask each player if they want to buy insurance.

(j). **surrender** method: Ask each player if they want to surrender.

(k). **split** method: Ask each player if they want to split. The way I implement splitting is, I have an array of **PlayerXXX** (length of 4), and I maintain an array of **Hand** (length of 8) to store players' hand. The last 4 **Hand-**elements of the array are used to handle splitting situations. The indexing is simple, the class of player i is **Player[i]**, and he/she has all **Hand[i%4]**. This naive indexing save me a lot of

time when processing results. The methods below use the same indexing method.

(l). **doubleDown** method: Ask each **hand** of player if they want to double down. (I use the word hand here because a player might have 2 hands now). In my casino, a player can double down even after splitting.

(m). **playerHit** method: Ask each hand of player if they want to hit.

(n). **dealerAction** method: Time for dealer to take action(hit/stand).

(o). **moneyTime** method: Process the results due to the rules from HW4 spec.

RandomIndex Class: the class that TA provided in the last HW.

3. The result of the duel between me and B01902002:

nChips=1000,nRound=10000

Result(I lose a little bit):

```
PlayerB01902002(0) has 1203.0 chips now
PlayerB01902002(1) has 1175.0 chips now
PlayerB01902039(2) has 974.5 chips now
PlayerB01902039(3) has 1014.5 chips now
```

Procedure:

a. Split:

```
--------Now players' hands--------
PlayerB01902002(0-0)(bet: 1.0) has HEART_9 CLUB_9
PlayerB01902002(1-0)(bet: 1.0) has CLUB_13 SPADE_1
PlayerB01902039(2-0)(bet: 1.0) has CLUB_9 SPADE_4
PlayerB01902039(3-0)(bet: 1.0) has CLUB_6 DIAMOND_4
Dealer has SPADE_5 DIAMOND_11
--------Buy insurance:--------
The Dealer's face-up card is SPADE_5
--------Surrender:--------
The Dealer's face-down card is DIAMOND_11
--------Split:--------
PlayerB01902002(0) splits
PlayerB01902002(0) get HEART_13
PlayerB01902002(4) get HEART_3
--------Now players' hands--------
PlayerB01902002(0-0)(bet: 1.0) has HEART_9 HEART_13
PlayerB01902002(1-0)(bet: 1.0) has CLUB_13 SPADE_1
PlayerB01902039(2-0)(bet: 1.0) has CLUB_9 SPADE_4
PlayerB01902039(3-0)(bet: 1.0) has CLUB_6 DIAMOND_4
PlayerB01902002(0-1)(bet: 1.0) has CLUB_9 HEART_3
Dealer has SPADE_5 DIAMOND_11
```

b. Double down:

```
--------Double down:--------
PlayerB01902039(3) double down
--------Now players' hands--------
PlayerB01902002(0-0)(bet: 1.0) has HEART_9 HEART_13
PlayerB01902002(1-0)(bet: 1.0) has CLUB_13 SPADE_1
PlayerB01902039(2-0)(bet: 1.0) has CLUB_9 SPADE_4
PlayerB01902039(3-0)(bet: 2.0) has CLUB_6 DIAMOND_4
PlayerB01902002(0-1)(bet: 1.0) has CLUB_9 HEART_3
Dealer has SPADE_5 DIAMOND_11
--------Player hit:--------
PlayerB01902039(3) get CLUB_5
--------Now players' hands--------
```

c. Dealer action & comparing hand value

```
--------Now players' hands--------
PlayerB01902002(0-0)(bet: 1.0) has HEART_9 HEART_13
PlayerB01902002(1-0)(bet: 1.0) has CLUB_13 SPADE_1
PlayerB01902039(2-0)(bet: 1.0) has CLUB_9 SPADE_4
PlayerB01902039(3-0)(bet: 2.0) has CLUB_6 DIAMOND_4 CLUB_5
PlayerB01902002(0-1)(bet: 1.0) has CLUB_9 HEART_3
Dealer has SPADE_5 DIAMOND_11
--------Dealer action:--------
Dealer hit
Dealer get HEART_5
--------Now players' hands--------
PlayerB01902002(0-0)(bet: 1.0) has HEART_9 HEART_13
PlayerB01902002(1-0)(bet: 1.0) has CLUB_13 SPADE_1
PlayerB01902039(2-0)(bet: 1.0) has CLUB_9 SPADE_4
PlayerB01902039(3-0)(bet: 2.0) has CLUB_6 DIAMOND_4 CLUB_5
PlayerB01902002(0-1)(bet: 1.0) has CLUB_9 HEART_3
Dealer has SPADE_5 DIAMOND_11 HEART_5
PlayerB01902002(0-0)(bet:1.0)'s hand value is 19
PlayerB01902002(1-0)(bet:1.0)'s hand value is 21
PlayerB01902039(2-0)(bet:1.0)'s hand value is 13
PlayerB01902039(3-0)(bet:2.0)'s hand value is 15
PlayerB01902002(0-1)(bet:1.0)'s hand value is 12
Dealer's hand value is 20
```

5. bonus:

(a). I think I use a good indexing method to hold the splitting situations.

(b). I think the class provided by TA this time is not very convenient to use. There are some parts that made me confused.

(i). In **Player** class, **toString()** should be pre-implemented (for example: return get_chip() in String format) and return a consistent return value in all kinds of Players.

(ii). In **Hand** class, every time I need to remove or insert an Card, I have to use my array list of Card to re-construct a hand class and replace the original one. It's too tedious.

(iii). In **Card** class, if it's me I will give it a toString() to return the suit and value in String format (ex: SPADE_13)