**Homework #4 POOCasino Blackjack**

B01902080 資工四 王于青

1. **Implementations of my player’s strategy**
2. Making bet:

For each round, my initial bookie is one fifteenth of the current chips I hold.

Then, I use a random multiplier to slightly change the bookie based on human actions.

If player wins in the last round, the multiplier falls between [1, 2].

If player pushes in the last round, the multiplier falls between [0.75, 1.5].

If player loses in the last round, the multiplier falls between [0.5, 1].

1. Buying insurance:

I make this decision simply based on the probability of dealer getting a 10.

The probability for dealer to get a 10 roughly falls between [0.186, 0.372].

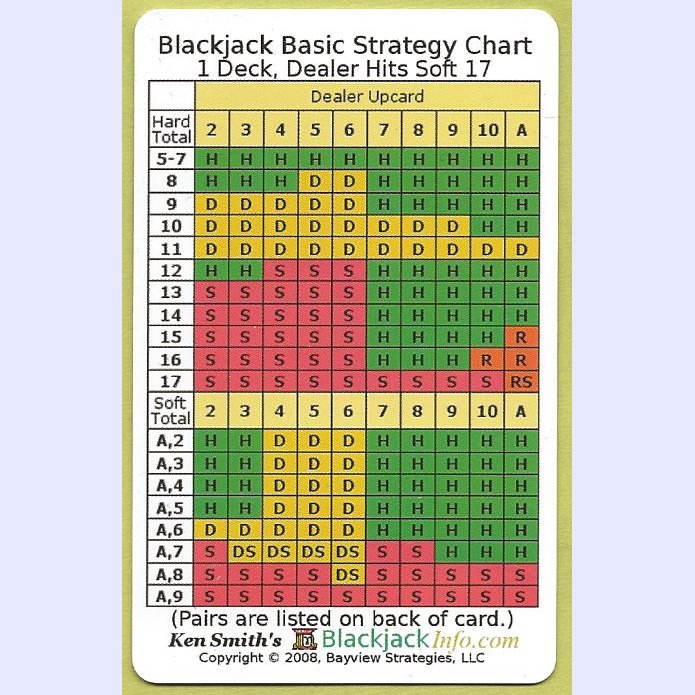
Therefore, player will buy insurance if the probability this round is higher than 0.28.

1. Doing split, double, hit, and stand:

I use the general one-deck strategy below to implement player’s decision.

I create several lookup tables to store the decisions under different circumstances.

Then, I can simply calculate the index of [my\_open, dealer\_open] pairs and return the corresponding action in the lookup tables.



1. Doing surrender:

As for surrendering, since the player cannot know the value of the face-down card, it’s not possible for player to follow the actions in the lookup table.

Therefore, I assume the face-down card is a 10-value card (probability of 10 is usually the highest).

Then, player can follow the lookup table above to see if surrendering is necessary.

1. **Designs and reasons of all the classes related to the casino**
2. Shuffler.java

I implement a Shuffler with methods to shuffle one deck of cards and assign single card.

Then I can simply shuffle cards and assign cards at the beginning of each round, and get new cards when players do splitting and hitting with this class.

1. POOCasino.java
2. I make all dealer instructions specified in spec into functions (as the screenshot shown below), and then I do further functionizing and implementations inside each function.

By doing so, it’s easier for me to check if the existing bugs and unexpected outcomes are caused by logical mistakes or programming errors.

1. I use an ArrayList<Integer> indexMatcher to specify the relation between hands and players.

Therefore, I can still get the owner of each hand

1. when some players are broke and out of the game.
2. after some players do splitting and have two hands at the same time.
3. Since the given Player.class requires specific type of inputs (e.g., Card, ArrayList<Card>, Hand…), I implement some methods to convert objects between these types efficiently.
4. PlayerB01902XXX.java toString()

The toString() method of my classmates’ and mine will print the remaining chips the player holds.

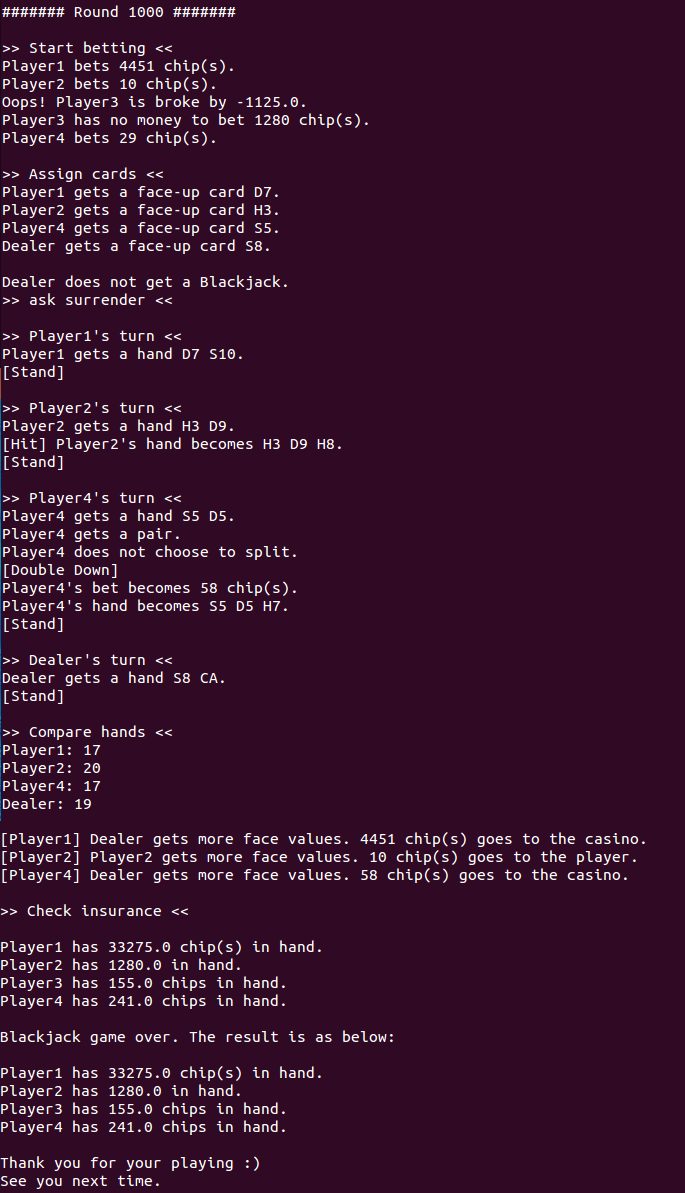
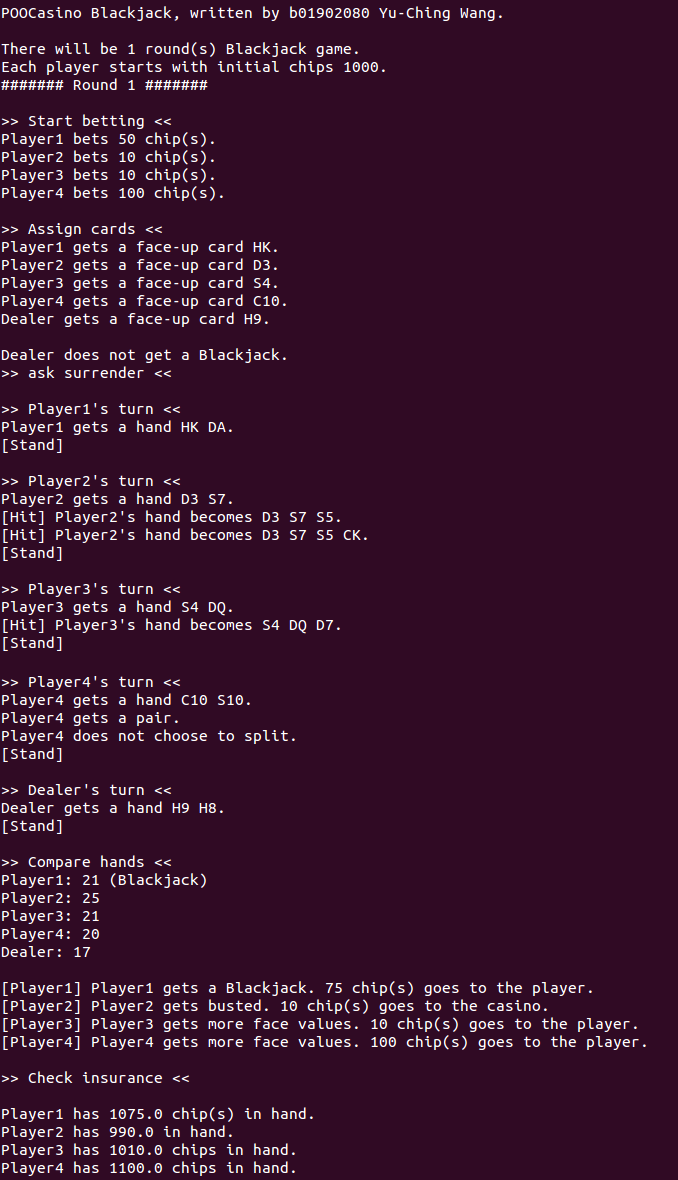
Then, it would be easier for us to keep track of players’ accounts during the game and therefore check the correctness of our program.

1. **The results of the duel between my classmates and me**

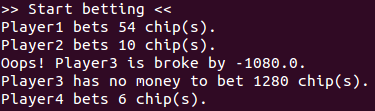
Player1: PlayerB01902080 (my player) , Player2: PlayerB01902041

Player3: PlayerB01902102 , Player4: PlayerB01902058

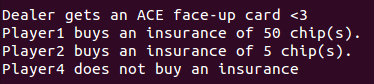
1. Result of the whole game (partial screenshot is shown as below):

The result of 1st round The result of last (1000st) round

1. Make bet (handling player.brokeException):



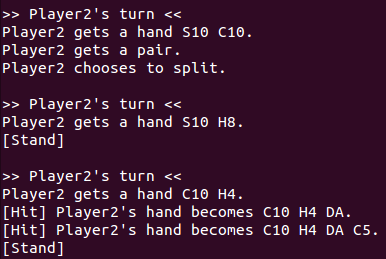
1. Buy insurance:



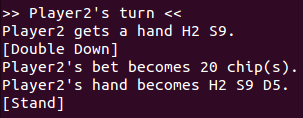
1. Do surrender:



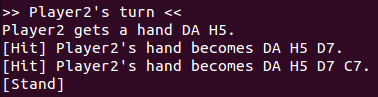
1. Do split:



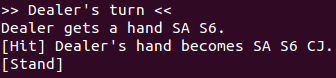
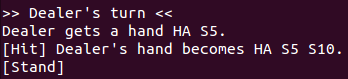
1. Do double down:



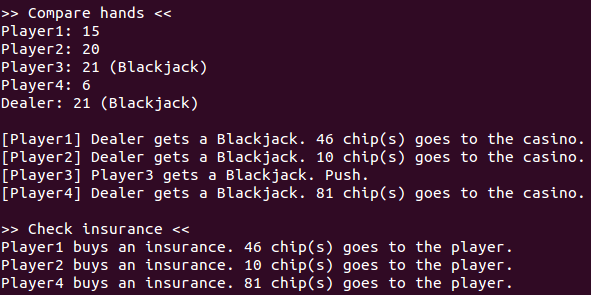
1. Do hit & stand:



1. Dealer’s actions

1. Compare the dealer’s and players’ hands



1. What we’ve learned during the duel:

My classmates say that I’m a cautious player compared to other players.

My bet is never bigger than 2/15 of the current chips I hold, so I won’t be broke anyway.

On the other hand, some of my classmates will make bigger bets intending to win back their loss regardless of how many chips they have. (isn’t that too crazy 😐 ?)

Therefore, my classmates sometimes become billionaires while sometimes they get broke.

I usually earn less money than they do, but I won’t be broke and I can leave the casino safely <3

1. **BONUS TIME <3**
2. Exceptions checking:

Whenever player is doing actions related to money (i.e., increase\_chips(diff), decrease\_chips(diff)), I will check if the action is valid and do the following instructions:

1. brokeException when making bet: player is out of the current round.

(Side effect: player who is broke will be out of the rest of the game.)

1. brokeException when buying insurance: player is not allowed to buy insurance.
2. brokeException when doing split: player is not allowed to split hand.

Also, if player make a 0 bet, I assume the player is quitting the game.

1. Assigning target cards commands:

By modifying makefile make run commands, user can assign target cards to players and dealer.

Commands: -$(playerPosition)$(openCardSuit)$(openCardValue)$(holeCardSuit)$(holeCardValue)

Ex: java POOCasino … playerB01902XXX -1SAHT -2D2D5 -3CQS8 -4H7D7 –dCADK

Player1 will get a hand SA H10.

Player2 will get a hand D2 D5.

Player3 will get a hand CQ S8.

Player4 will get a hand H7 D7.

Dealer will get a hand CA DK.

With these commands, it’s more convenient and efficient to test the correctness of my program.

1. BLACKLIST mechanism:

I’ve heard from other casinos that there exists a difficult customer B01902135 who usually makes negative bets, does split and double when having not enough money, and throws exceptions whenever she likes without considering the dealer and other players.

In order for the fairness and safety of my lovely casino, I’ve listed the customer B01902135 on my blacklist and forbidden the customer to come into my casino FOREVER <3