FOOP—HW4 Report
Name: 江東峻
Student ID: B01902032

1. Player Strategy

To decide the player strategy, I refer to some webpages:
refer A: https://www.blackjackinfo.com/blackjack-basic-strategy-engine/?numdecks=1&soft17=h17&dbl=all&das=yes&surr=ns&peek=no
refer B: https://www.blackjackinfo.com/blackjack-rules/insurance-and-surrender/

(1) Hit, Double, Split
I use the tables in refer A, and the reference also show the estimated casino edge for the strategy(1 deck, H17, DAS, No Surrender, No Peek) and the edge is 0.15%.

(2) Buy Insurance
In refer B: It says: "Unless you are card counter and know the deck is skewed sufficiently, just ignore the insurance bet. It doesn't matter whether you have a good hand or a bad hand." So I choose to never buy insurances.

(3) Surrender
Considering the class structure, I have to decide whether to surrender before I check the face-down card. It doesn't make sense to surrender with having information of only one card. So I choose to never surrender.

(4) Bet
I don't consider to win, so I always bet 1 dollar.

2. Class Design

(1) Class POOCasino
The entrance of this program. It handles arguments from shell and the initialization of players and Blackjack game.

(2) Class Blackjack
The main part of this program. It runs Blackjack games and catches the two exceptions thrown from Player.class. It also maintain the table of player information(PlayerInfo).
I split the process of Blackjack into several stages: bet, insurance, surrender and hit & double down. My expectation is to make function of each stage like "layers" and less parameters, so that I can call a function, and do something to all players. "doBet" and "askInsurance" are the example. But "askHit" and "askDoubleDown" cannot be layer-like functions, I just add a "position" argument.

For handling BrokeException, I remove(kick out) the player which is broken and only when the player's chips is negative. So a player can make a bet larger than his/her chips, but after the player hand over the bet(call decrease_chips), the player's chips is negative, BrokeException throws and Blackjack will remove this player(not give back the bet he hands over).
For handling NegativeException, I think this is a system error(It cannot happen!), so the Blackjack print information and exit the program.
For passing "current_table", I think it doesn't matter to pass all hands of players and not to add a specific player information is not necessary. So I choose to pass all open hands of players.
For each round, the dealer open a new deck and shuffle it.
To split, the player has to advance 2*bet dollars for another position('2' for the probability of double down).

(3) Class Dealer
The class for dealer. It contains some dealer's action(e.g. open a new deck, deal cards to a player, …).

(4) Class PlayerInfo
The information of a player. It contains the name of player, player's id(for system check because player's name can be duplicate), player's status('OK' or 'Surrender' or 'Busted'), bet, insurance, hand(open cards and face-down cards). Blackjack will repeatedly refresh the table of PlayerInfo(contains all information of players).

(5) Class PlayerB01902032
The implementation of player strategy. toString() returns the name of the player.

3. Running Result

傅彥禎 B01902030
Parameters: 1000 rounds, 1000 initial chips
Players: player 0, 2: PlayerB01902032; player 1, 3: PlayerB01902030

```
[play]=============Round 1000===============
[doBet]Player b01902030 bets 1!
[doBet]Player B01902032 bets 1!
[doBet]Player b01902030 bets 1!
[doBet]Player B01902032 bets 1!
Dealer: [PlayerInfo]Status: OK, Bet: 0, Hand: H3 , Insurance: No
[play]-----player time-------
[play]Player B01902032[0] turn!
[play]Player B01902032[0] flip out!
[PlayerInfo]Status: OK, Bet: 1, Hand: DJ S10 , Insurance: No
------------------------------
```

```
[play]Player b01902030[1] turn!
[play]Player b01902030[1] flip out!
[PlayerInfo]Status: OK, Bet: 1, Hand: SJ D3 , Insurance: No
[play]Player b01902030[1] hit: C10
[PlayerInfo]Status: Busted, Bet: 1, Hand: SJ D3 C10 , Insurance: No
------------------------------
[play]Player B01902032[2] turn!
[play]Player B01902032[2] flip out!
[PlayerInfo]Status: OK, Bet: 1, Hand: S6 D5 , Insurance: No
[askDoubleDown]Player B01902032[2] Double Down! total bet: 2
[PlayerInfo]Status: OK, Bet: 2, Hand: S6 D5 S8 , Insurance: No
------------------------------
[play]Player b01902030[3] turn!
[play]Player b01902030[3] flip out!
[PlayerInfo]Status: OK, Bet: 1, Hand: S7 H8 , Insurance: No
------------------------------
[play]-----dealer time-------
[play]Dealer flip out!
[PlayerInfo]Status: OK, Bet: 0, Hand: H3 DK , Insurance: No
[play]Dealer hit: D7
*********************************
[doResulting]Resulting:
Dealer: [PlayerInfo]Status: OK, Bet: 0, Hand: H3 DK D7 , Insurance: No
---------------------------------
[PlayerInfo]Status: OK, Bet: 1, Hand: DJ S10 , Insurance: No
Player B01902032[0] bets 1.000000, reward 1.000000, now: 1043.500000
---------------------------------
[PlayerInfo]Status: Busted, Bet: 1, Hand: SJ D3 C10 , Insurance: No
Player b01902030[1] bets 1.000000, reward 0.000000, now: 977.000000
---------------------------------
[PlayerInfo]Status: OK, Bet: 2, Hand: S6 D5 S8 , Insurance: No
Player B01902032[2] bets 2.000000, reward 0.000000, now: 1028.500000
---------------------------------
[PlayerInfo]Status: OK, Bet: 1, Hand: S7 H8 , Insurance: No
Player b01902030[3] bets 1.000000, reward 0.000000, now: 1027.000000
---------------------------------
==========================================
```

4. How to execute

(1) to compile: “make”
(2) to start game: “make run”

6. To get “bonus” points

This part shows the amendment of the spec:

(1) current_table

Not the same in spec, I pass all hands rather than the hands of others hands. This approach is more efficient because we only have to maintain one table.

(2) toString

Player’s method “toString()”, in java document, shows the status of a player, but I only show the player name. I choose to store the information in PlayerInfo class, and it’s more consistent of information storage(not only chips but also other informations: bet, insurance, …, etc). I think store information in the same place is a better approach.