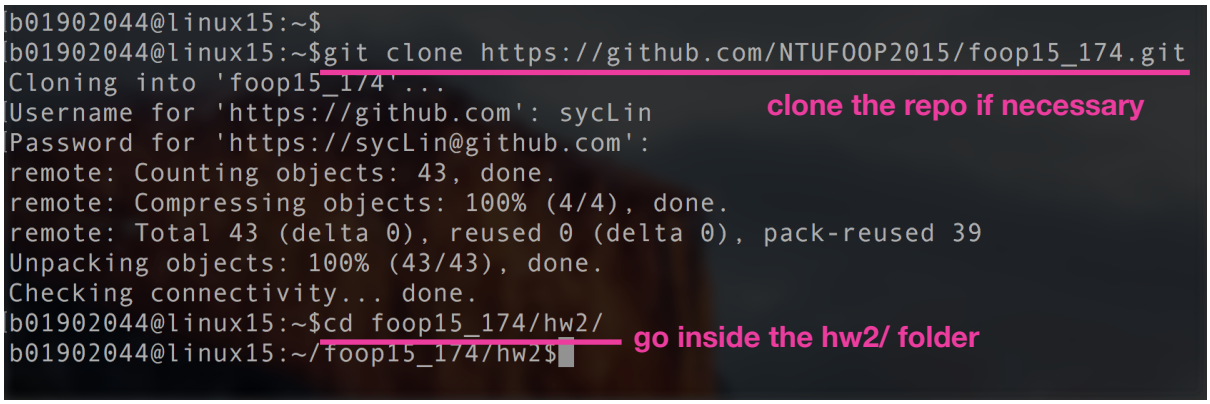
B01902044 Steven YuChih Lin
FOOP 2015 fall HW2
Instructor: Hsuan-Tien Lin.
2015/10/21

How a human player should play

Step #1: Go to my hw2/ folder under my GitHub Repo

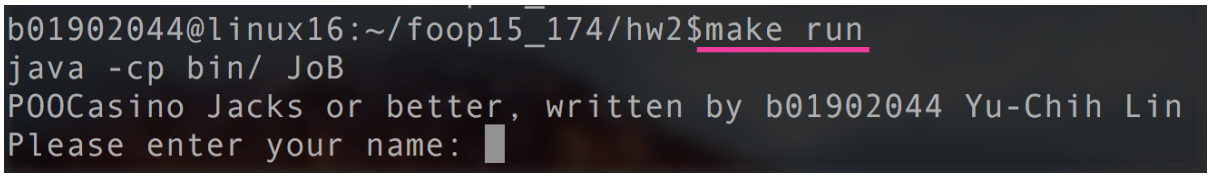


Step #2: Compile my source files properly

Simply type **`$make`** into your prompt will do.

Step #3: Start the game

Use **`$make run`** to start the game. As below:



Step #4: The game works exactly as stated in hw2.pdf

a) Input your bet, an integer from 1 to 5; input 0 if quitting the game.
b) Choose the card you'd like to keep by typing in the options (a) through (e). Input "none" if you'd like to keep all those 5 cards.
c) The program automatically replace the cards you don't want with new ones and evaluate your hand.
d) The program then pay you off according to the payoff table in hw2.pdf.

How I tested the correctness

Similar to the idea of unit testing, during the developing process, I wrote small pieces of codes to test the reliability of the methods (a.k.a., actions) of every class.

For example:

The way I tested my “Computer.determineBestHand()”.

```
/* test the method: Computer.determineBestHand */
System.out.println("Now let's test the functionality of the method: Computer.determineBestHand()");
Computer testComputer = new Computer();
// first let's give him nothing
int tmp = testComputer.determineBestHand(null);
System.out.println("it returns " + tmp + " if no arguments passed to the method.");
// let's give him less than 5 cards
Card[] lessCards = new Card[2];
lessCards[0] = new Card("S", "2");
lessCards[1] = new Card("D", "3");
tmp = testComputer.determineBestHand(lessCards);
System.out.println("it returns " + tmp + " if passing less than 5 cards.");
// let's give him more than 5 cards
Card[] moreCards = new Card[6];
moreCards[0] = new Card("S", "A"); moreCards[1] = new Card("S", "2"); moreCards[2] = new Card("S", "3");
moreCards[3] = new Card("D", "A"); moreCards[4] = new Card("D", "2"); moreCards[5] = new Card("D", "3");
tmp = testComputer.determineBestHand(moreCards);
System.out.println("it returns " + tmp + " if passing more than 5 cards.");
```

The result:

```
01:38:27[stevenmbpr@StevenMBPRetina:~/Google 雲端硬碟/2015fall/OOP/foop15_174/hw2]
└──> make run
java -cp bin/ JoB
Now let's test the functionality of the method: Computer.determineBestHand()
it returns -1 if no arguments passed to the method.
it returns -1 if passing less than 5 cards.
it returns -1 if passing more than 5 cards.
```

The output from three rounds

The blue bubbles indicate the input from human player.

```
b01902044@linux16:~/foop15_174/hw2$make run
java -cp bin/ JoB
POOCasino Jacks or better, written by b01902044 Yu-Chih Lin
Please enter your name: CharlieLaLaLa
Welcome, CharlieLaLaLa.
You have 1000 P-dollars now.
Please enter your P-dollar bet for round 1 (1-5 or 0 for quitting the game): 3
Your cards are (a) H6 (b) S7 (c) H8 (d) CK (e) HK
Which cards do you want to keep? de
Okay. I will discard (a) H6 (b) S7 (c) H8.
Your new cards are (a) D3 (b) H4 (c) HQ (d) CK (e) HK
You get a Jacks or better hand. The payoff is 3.
Your have 1000 P-dollars now.
Please enter your P-dollar bet for round 2 (1-5 or 0 for quitting the game): 4
Your cards are (a) H3 (b) C8 (c) C10 (d) DJ (e) HQ
Which cards do you want to keep? de
Okay. I will discard (a) H3 (b) C8 (c) C10.
Your new cards are (a) D3 (b) H4 (c) D6 (d) DJ (e) HQ
You get a others hand. The payoff is 0.
Your have 996 P-dollars now.
Please enter your P-dollar bet for round 3 (1-5 or 0 for quitting the game): 5
Your cards are (a) C3 (b) C8 (c) D9 (d) H10 (e) HK
Which cards do you want to keep? e
Okay. I will discard (a) C3 (b) C8 (c) D9 (d) H10.
Your new cards are (a) D3 (b) S10 (c) DK (d) HK (e) DA
You get a Jacks or better hand. The payoff is 5.
Your have 996 P-dollars now.
Please enter your P-dollar bet for round 4 (1-5 or 0 for quitting the game): 0
Good bye, CharlieLaLaLa. You played for 3 round and have 996 P-dollars now.
b01902044@linux16:~/foop15_174/hw2$
```

Any part worth getting “bonus” points

Javadoc and Comments

I wrote proper comments for each method and instance variable of my classes according to the specification of Javadoc, providing explanation for all parameters and return values as well as stating the purpose of the method briefly. (you can refer to `hw2/doc/` after `$make`)

For example:

```
/**
 * check if player wants to discard any cards; replace them.
 * display the new cards (if any) to player (to screen).
 * @param player the player to re-distribute cards.
 * @param keepOrNot an array of 5 booleans to indicate if keeping the card.
 */
public void redistributeCard(Player player, boolean[] keepOrNot) {
```

↑ Example 1. Comments for the method: `Computer.redistributeCard();`

↓ Example 2. Comments for the method: `Card.smallerThan();`

```
/**
 * Determine if smaller than a given card.
 * @param c another card.
 * @return  true if smaller than c; otherwise, false is return.
 */
public boolean smallerThan(Card c) {
```

Option order

The player can choose the cards he/she would like to keep by inputting options where the order and the count both do not matter. In addition, illegal characters will be ignored. For example:

```
Your cards are (a) C5 (b) S6 (c) H9 (d) CK (e) HK
Which cards do you want to keep? dbbbcxyz
Okay. I will discard (a) C5 (e) HK
```

Equivalent to inputting “bcd”.

(because ‘x’, ‘y’, and ‘z’ will be discarded, and it doesn’t matter how many ‘b’ you typed.)

Encapsulation

Use encapsulation for something best “kept private”.

For example:

```
private int[] payoffRatio = new int[]{250, 50, 25, 9, 6, 4, 3, 2, 1, 0};

/**
 * store the types of hand
 */
private String[] handType = new String[] {
```

↑ Instance variables such as `payoffRatio;` and `handType;` are better “kept private”.

↓ Likewise, it’s better to keep `Computer.determineBestHand();` private.

```
private int determineBestHand(Card[] hand) {
```