B01902044 Steven YuChih Lin
FOOP 2015 fall HW3
Instructor: Huan-Tien Lin.
2015/11/16

Class Structures: Design and Reasons

All of my class definitions can be classified into two categories: Entity and Conceptual Entity.

#1 Entity - Real-world Objects
class `Card`: representing a card.
class `Deck`: representing a deck (of cards).
class `Player`: every instance of this class is a player that can participate in a game.
class `HumanPlayer`: a sub-class of `Player`, interacting with users and asking for input.

#2 Conceptual Entity - concepts that has precise definitions and explicit properties
class `OldMaidGame`: a base class for variants, filled with default implementations of HW1.
class `Var1`: with 51 cards (a Queen is removed) instead of having 52 regular cards plus 2 jokers.
class `Var2`: with 103 cards (2 regular deck and remove a Queen) and 6 players.
(`Var1` and `Var2` are both children of `OldMaidGame`)

It is normal to have a class definition of real-world objects. And the reason that I created a class for `OldMaidGame` is for variants to inherit, and therefore those variants can be created with less efforts and less time with more readability in codes.

How a human player should play

The two variants that I implemented in this homework have similar rules to the game described in homework 1. After `$make run`, you can choose one of the three games to play.

```
Which game to play?
[0] Classic OldMaidGame
[1] Variant #1
[2] Variant #2
Your choice >
```

“Choose which game to play.”

- program entry

After choosing the game, you can set the number of players and determine whether there would be a human player or not.

```
Set the number of players: (enter 0 for default)
> 0
Do you wanna create a human player?
[Y/N] > Y
```

“Set up the game”

- number of players, human player

If there's a human player and it's his/her turn, the program pauses and asks for user to "draw" a card, i.e., to choose an index for another player's hand.

```
Game start
There are 7 cards you can choose from.
Which one do you like[0-6]:
```

"Human player drawing a card"

- program pauses when it's human player's turn

It there isn't any human player in the game, the program will instantly finish the game.
(For sample output of no human player, please see to the Sample Output section.)

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.

Sample Output

Sample output for Var1

```
Which game to play?
[0] Classic OldMaidGame
[1] Variant #1
[2] Variant #2
Your choice > 1
Set the number of players: (enter 0 for default)
> 0
Do you wanna create a human player?
[Y/N] > N
Deal cards
Player0: S5 H6 C7 D7 C8 D8 H8 C9 H9 SJ DK HA
Player1: C2 D2 H2 H3 D4 C5 C6 D6 S6 CJ HJ HK CA
Player2: S2 C3 D3 S3 C4 D5 D9 C10 S10 DJ DQ HQ CK
Player3: H4 S4 H5 H7 S7 S8 S9 D10 H10 SQ SK DA SA
Drop cards
Player0: S5 H6 H8 SJ DK HA
Player1: H2 H3 D4 C5 S6 HK CA
Player2: S2 S3 C4 D5 D9 DJ CK
Player3: H5 S8 S9 SQ SK
Game start
Player0 draws a card from Player1 H3
Player0: H3 S5 H6 H8 SJ DK HA
Player1: H2 D4 C5 S6 HK CA
Player1 draws a card from Player2 CK
Player1: H2 D4 C5 S6 CA
Player2: S2 S3 C4 D5 D9 DJ
Player2 draws a card from Player3 S9
Player2: S2 S3 C4 D5 DJ
Player3: H5 S8 SQ SK
Player3 draws a card from Player0 SJ
Player3: H5 S8 SJ SQ SK
Player0: H3 S5 H6 H8 DK HA
Player0 draws a card from Player1 D4
Player0: H3 D4 S5 H6 H8 DK HA
Player1: H2 C5 S6 CA
Player1 draws a card from Player2 DJ
```

Left: 1/5

Right: 2/5

```
Player1: H2 C5 S6 DJ CA
Player2: S2 S3 C4 D5
Player2 draws a card from Player3 SJ
Player2: S2 S3 C4 D5 SJ
Player3: H5 S8 SQ SK
Player3 draws a card from Player0 HA
Player3: H5 S8 SQ SK HA
Player0: H3 D4 S5 H6 H8 DK
Player0 draws a card from Player1 CA
Player0: H3 D4 S5 H6 H8 DK CA
Player1: H2 C5 S6 DJ
Player1 draws a card from Player2 S3
Player1: H2 S3 C5 S6 DJ
Player2: S2 C4 D5 SJ
Player2 draws a card from Player3 SK
Player2: S2 C4 D5 SJ SK
Player3: H5 S8 SQ HA
Player3 draws a card from Player0 D4
Player3: D4 H5 S8 SQ HA
Player0: H3 S5 H6 H8 DK CA
Player0 draws a card from Player1 DJ
Player0: H3 S5 H6 H8 DJ DK CA
Player1: H2 S3 C5 S6
Player1 draws a card from Player2 SJ
Player1: H2 S3 C5 S6 SJ
Player2: S2 C4 D5 SK
Player2 draws a card from Player3 SQ
Player2: S2 C4 D5 SQ SK
Player3: D4 H5 S8 HA
Player3 draws a card from Player0 S5
Player3: D4 S8 HA
Player0: H3 H6 H8 DJ DK CA
Player0 draws a card from Player1 C5
Player0: H3 C5 H6 H8 DJ DK CA
Player1: H2 S3 S6 SJ
Player1 draws a card from Player2 SQ
```

```
Player1: H2 S3 S6 SJ SQ
Player2: S2 C4 D5 SK
Player2 draws a card from Player3 D4
Player2: S2 D5 SK
Player3: S8 HA
Player3 draws a card from Player0 H6
Player3: H6 S8 HA
Player0: H3 C5 H8 DJ DK CA
Player0 draws a card from Player1 SJ
Player0: H3 C5 H8 DK CA
Player1: H2 S3 S6 SQ
Player1 draws a card from Player2 D5
Player1: H2 S3 D5 S6 SQ
Player2: S2 SK
Player2 draws a card from Player3 S8
Player2: S2 S8 SK
Player3: H6 HA
Player3 draws a card from Player0 CA
Player3: H6
Player0: H3 C5 H8 DK
Player0 draws a card from Player1 H2
Player0: H2 H3 C5 H8 DK
Player1: S3 D5 S6 SQ
Player1 draws a card from Player2 S2
Player1: S2 S3 D5 S6 SQ
Player2: S8 SK
Player2 draws a card from Player3 H6
Player2: H6 S8 SK
Player3:
Player3 wins
Player0 draws a card from Player1 S2
Player0: H3 C5 H8 DK
Player1: S3 D5 S6 SQ
Player1 draws a card from Player2 S8
Player1: S3 D5 S6 S8 SQ
```

```
Player2: H6 SK
Player2 draws a card from Player0 H3
Player2: H3 H6 SK
Player0: C5 H8 DK
Player0 draws a card from Player1 SQ
Player0: C5 H8 SQ DK
Player1: S3 D5 S6 S8
Player1 draws a card from Player2 SK
Player1: S3 D5 S6 S8 SK
Player2: H3 H6
Player2 draws a card from Player0 H8
Player2: H3 H6 H8
Player0: C5 SQ DK
Player0 draws a card from Player1 S3
Player0: S3 C5 SQ DK
Player1: D5 S6 S8 SK
Player1 draws a card from Player2 H6
Player1: D5 S8 SK
Player2: H3 H8
Player2 draws a card from Player0 C5
Player2: H3 C5 H8
Player0: S3 SQ DK
Player0 draws a card from Player1 D5
Player0: S3 D5 SQ DK
Player1: S8 SK
Player1 draws a card from Player2 H3
Player1: H3 S8 SK
Player2: C5 H8
Player2 draws a card from Player0 SQ
Player2: C5 H8 SQ
Player0: S3 D5 DK
Player0 draws a card from Player1 H3
Player0: D5 DK
Player1: S8 SK
Player1 draws a card from Player2 SQ
Player1: S8 SQ SK
```

```
Player2: C5 H8
Player2 draws a card from Player0 D5
Player2: H8
Player0: DK
Player0 draws a card from Player1 S8
Player0: S8 DK
Player1: SQ SK
Player1 draws a card from Player2 H8
Player1: H8 SQ SK
Player2:
Player2 wins
Player0 draws a card from Player1 SK
Player0: S8
Player1: H8 SQ
Player1 draws a card from Player0 S8
Player1: SQ
Player0:
Player0 wins
Game Over
```

Left: 3/5

Middle: 4/5

Right: 5/5

Sample output for Var2

```
Which game to play?
[0] Classic OldMaidGame
[1] Variant #1
[2] Variant #2
Your choice > 2
Set the number of players: (enter 0 for default)
> 3
Do you wanna create a human player?
[Y/N] > N
Deal cards
Player0: D2 D2 H2 H2 S2 H3 C4 D4 S4 S4 C5 D5 H5 D6 C7 D8 S8 C10 D10 D10 H10 H10 DJ DJ SJ SQ SQ DK HK CA DA HA SA SA
Player1: C2 S2 C3 C3 D3 H3 S3 H4 D5 H5 S5 C6 C6 D6 D7 D7 C8 C8 H8 C9 D9 S10 CJ HJ SJ CQ DQ HQ HQ CK DK HK SK DA
Player2: C2 D3 S3 C4 D4 H4 C5 S5 H6 H6 S6 S6 C7 H7 H7 S7 S7 D8 H8 S8 C9 D9 H9 H9 S9 S9 C10 S10 CJ HJ DQ CK SK CA HA
```

```
Game start
Player0 draws a card from Player1 S5
Player0: S2 H3 D6 C7 H10 SJ SA
Player1: S3 H4 D6 H8 S10 SJ DA
Player1 draws a card from Player2 H4
Player1: S3 D6 H8 S10 SJ DA
Player2: C2 S7 S8 DQ
Player2 draws a card from Player0 SA
Player2: C2 S7 S8 DQ SA
Player0: S2 H3 D6 C7 H10 SJ
Player0 draws a card from Player1 D6
Player0: S2 H3 C7 H10 SJ
Player1: S3 H8 S10 SJ DA
Player1 draws a card from Player2 S7
Player1: S3 S7 H8 S10 SJ DA
Player2: C2 S8 DQ SA
Player2 draws a card from Player0 H3
Player2: C2 H3 S8 DQ SA
Player0: S2 C7 H10 SJ
Player0 draws a card from Player1 S10
Player0: S2 C7 SJ
Player1: S3 S7 H8 SJ DA
Player1 draws a card from Player2 SA
Player1: S3 S7 H8 SJ
Player2: C2 H3 S8 DQ
Player2 draws a card from Player0 C7
Player2: C2 H3 C7 S8 DQ
Player0: S2 SJ
Player0 draws a card from Player1 H8
Player0: S2 H8 SJ
Player1: S3 S7 SJ
Player1 draws a card from Player2 S8
Player1: S3 S7 S8 SJ
```

Up: 1/3 (setup the game)

Left: 2/3

Right: 3/3

```
Player2: C2 H3 C7 DQ
Player2 draws a card from Player0 H8
Player2: C2 H3 C7 H8 DQ
Player0: S2 SJ
Player0 draws a card from Player1 SJ
Player0: S2
Player1: S3 S7 S8
Player1 draws a card from Player2 C2
Player1: C2 S3 S7 S8
Player2: H3 C7 H8 DQ
Player2 draws a card from Player0 S2
Player2: S2 H3 C7 H8 DQ
Player0:
Player0 wins
Player1 draws a card from Player2 S2
Player1: S3 S7 S8
Player2: H3 C7 H8 DQ
Player2 draws a card from Player1 S8
Player2: H3 C7 DQ
Player1: S3 S7
Player1 draws a card from Player2 DQ
Player1: S3 S7 DQ
Player2: H3 C7
Player2 draws a card from Player1 DQ
Player2: H3 C7 DQ
Player1: S3 S7
Player1 draws a card from Player2 H3
Player1: S7
Player2: C7 DQ
Player2 draws a card from Player1 S7
Player2: DQ
Player1:
Player1 wins
Game Over
```

Bonus Part

Human Player Supported

The class `HumanPlayer` is the sub-class of `Player`. It allows users to create human players for any of the games. In addition, one can implement his/her own AI for drawing cards by simply overriding the function `drawingStrategy(int)`.

```
/**
 * human player
 */
class HumanPlayer extends Player {
    /**
     * ask for human player's input instead of pure random strategy
     * @param totalCount the total number of cards to choose from
     * @return integer indicating the index of the card we want
     */
    @Override
    public int drawingStrategy(int totalCount) {...
    }
}
```

The HumanPlayer class

- supported user interaction

Javadoc and Comments

I wrote proper comments for each method and instance variable of my classes according to the spec of Javadoc, providing sufficient explanation for all parameters as well as return values. Also the purposes of all methods are stated briefly. (Documentation can be accessed in `doc/` after `$make`)

```
/**
 * remove cards from the deck
 * @param s the suit of the card we'd like to remove
 * @param r the rank of the card we'd like to remove
 * @return true if removed successfully; false, otherwise.
 */
public boolean removeCard(String s, String r) {...
}

/**
 * add all the cards in the given ArrayList of Card class
 * @param cards an ArrayList of cards to add
 * @return true if succeeded; false if failed.
 */
public boolean addAll(ArrayList<Card> cards) {
    return this.cards.addAll(cards);
}
```

Comments & Descriptions

- according to javadoc spec