FOOP—HW3 Report
Name: 江東峻
Student ID: B01902032

1. Class structures

The main idea of design is to make “implement a variant” easy. The most common variant is “change the cards used”, or “can not drop some cards” or “how to win”…. My goal is keep definition part of these rules in a simple function so that I can change these rules easily. In my class structures, the key part is “loadRules”. By revise (override) this function, we make a new variant.

(1) class Card
  (a) constructor: set the suit and number of a card.
  (b) toString: override “toString” function to print in [suit][number] format.
  (c) compare: used in sort function to compare number first, and then suit.
  (d) ispairOf: check whether two cards have the same number.
  (e) notIn: check whether a card is not in a set of cards(ArrayList) or not.

(2) class Player
  (a) constructor: set player id, name and intialize the state of players (Quit = 0 means this player are still playing the game).
  (b) getHands: add a card to the hand of the player.
  (c) Handsort: sort hand cards in the order defined by “compare” in class Card.
  (d) Showhands: print out the cards in hand.
  (e) DropPairCard: drop card pairs with the same number and not special cards.
  (f) DrawCardFrom: randomly draw a card from another player and add the card in the hand.
  (g) noHands: check whether the player has no hand cards.

(3) class OldMaid
  (a) loadRules: to define some additional or variant rules.
  (b) card and player initialize: to initialize the deck and player hand cards.
  (c) someoneQuit: to check whether there is a player playing (Quit = 0) and without hand cards (noHands = 1).
  (d) Play: define the main procedure of the OldMaid game. User can use “Play()” to start a OldMaid game.

(4) class VarianceOne and class VarianceTwo
  (a) all have to do is to override the loadRules function and we can easily make a lot of variants.

(5) class PlayGame
  (a) choose which variant to play.

2. How to play the two variants

The Two variants are variants of special cards and the definition of win.

(To play Original OldMaid, use “make run”)

(1) VarianceOne (“make run1”)

The difference between VarianceOne and Original is special cards. In Original game, players cannot drop a pair of jokers which we called “special cards”. But in

VarianceOne, the game system randomly throw out two cards so there are two cards that would not find the pair, and the two jokers would not be added in this game. Until the end of game, players finally know which cards are special cards so the tip of this variant to win is just luck. This game seems in favor of low-skilled players.

(2) VarianceTwo (“make run2”)

The difference between VarianceTwo and Original is the definition of win. In Original game, players with no hand cards win, but in VarianceTwo, players with no hand cards lose. Players have to keep the two jokers to the end, and everyone want to draw jokers from others. The tip to win is to hide the jokers and mislead others. This game needs schemes.

3. How to test the correctness

I write a unit test program (test.java) to test each function in each class. To see the result of test, use “make test”, and it will show the boolean value of whether all functions in this class work or not. Sample output:

```
TestCard: true
...
TestPlayer: true
...
TestOldMaid: true
```

4. The sample output for each variant

(Use … to skip some details to save space of pages)
Original OldMaid (Classical):

```
GameType: Classical
load rules
Deal cards
...
Drop cards
Anonymous (player 0): S2 S3 C5 C9 H10 CA
Anonymous (player 1): H5 S6 SJ HQ
Anonymous (player 2): B0 C2 C3 C8 D10 HJ DQ HK DA
Anonymous (player 3): R0 D6 D8 S9 DK
Game start
Anonymous (player 0) draws a card from Anonymous (player 1) H5
Anonymous (player 0): S2 S3 C9 H10 CA
Anonymous (player 1): S6 SJ HQ
...
Anonymous (player 0): S3 S6 SJ
Anonymous (player 1): B0 C3 HQ
Anonymous (player 1) draws a card from Anonymous (player 2) DQ
Anonymous (player 1): B0 C3
Anonymous (player 2): R0 HJ
Anonymous (player 2) draws a card from Anonymous (player 3) D6
Anonymous (player 2): R0 D6 HJ
Anonymous (player 3):
Anonymous (player 3) wins
Basic game over
Continue
Anonymous (player 0) draws a card from Anonymous (player 1) C3
Anonymous (player 0): S6 SJ
Anonymous (player 1): B0
...
Anonymous (player 2):
Anonymous (player 2) wins
Anonymous (player 0) draws a card from Anonymous (player 1) R0
Anonymous (player 0): R0 B0 S6
Anonymous (player 1): D6
Anonymous (player 1) draws a card from Anonymous (player 0) S6
Anonymous (player 1):
Anonymous (player 0): R0 B0
Anonymous (player 1) wins
Bonus game over
```

VarianceOne:

```
GameType: VarianceOne
load rules
Deal cards
...
Drop cards
...
Game start
Anonymous (player 0) draws a card from Anonymous (player 1) C2
Anonymous (player 0): C2 S3 H4 D5 C6 C8 CJ DQ CK DA
Anonymous (player 1): H3 S6 H8 D10 DJ HK
...
Anonymous (player 2) draws a card from Anonymous (player 3) R0
Anonymous (player 2): D6 S10 DQ
Anonymous (player 3):
Anonymous (player 3) wins
Basic game over
Continue
…

Anonymous (player 1) draws a card from Anonymous (player 2) CJ
Anonymous (player 1): CJ DQ
Anonymous (player 2):
Anonymous (player 2) wins
Bonus game over
```

In the end of VarianceOne, we can see the final remaining cards are not R0 and B0.

VarianceTwo:

```
GameType: VarianceTwo
load rules
Deal cards
...
Drop cards
...
Game start
...
Anonymous (player 1) draws a card from Anonymous (player 2) B0
Anonymous (player 1): B0 D4
Anonymous (player 2): R0 D9
Anonymous (player 2) draws a card from Anonymous (player 3) C9
Anonymous (player 2): R0
Anonymous (player 3):
Anonymous (player 3) loses
Anonymous (player 0) draws a card from Anonymous (player 1) D4
Anonymous (player 0):
Anonymous (player 1): B0
Anonymous (player 0) loses
Anonymous (player 1) draws a card from Anonymous (player 2) R0
Anonymous (player 1): R0 B0
Anonymous (player 2):
Anonymous (player 2) loses
Anonymous (player 1) win
Bonus game over
```

Players without hand cards lose, only the player keep the cards until the end wins.

5. How to execute

(1) compile: “make”
(2) to start Original OldMaid game: “make run”
(3) to start VarianceOne game: “make run1”
(4) to start VarianceTwo game: “make run2”
(5) to get the result of testing the function in each class: “make test”

6. To get “bonus” points

(1) I implement another variant: VarianceThree. This variant can play with 5 players, and I only add a code to change the PlayerNumber. Other versions of more players OldMaid seem to be easily implemented.
(2) I redo the bonus game in HW1, to continue the game. And in other variants, if they can, I implement the similar bonus games, too.