Group Project Deliverable2 Report

War Game

1. User Case Diagram:

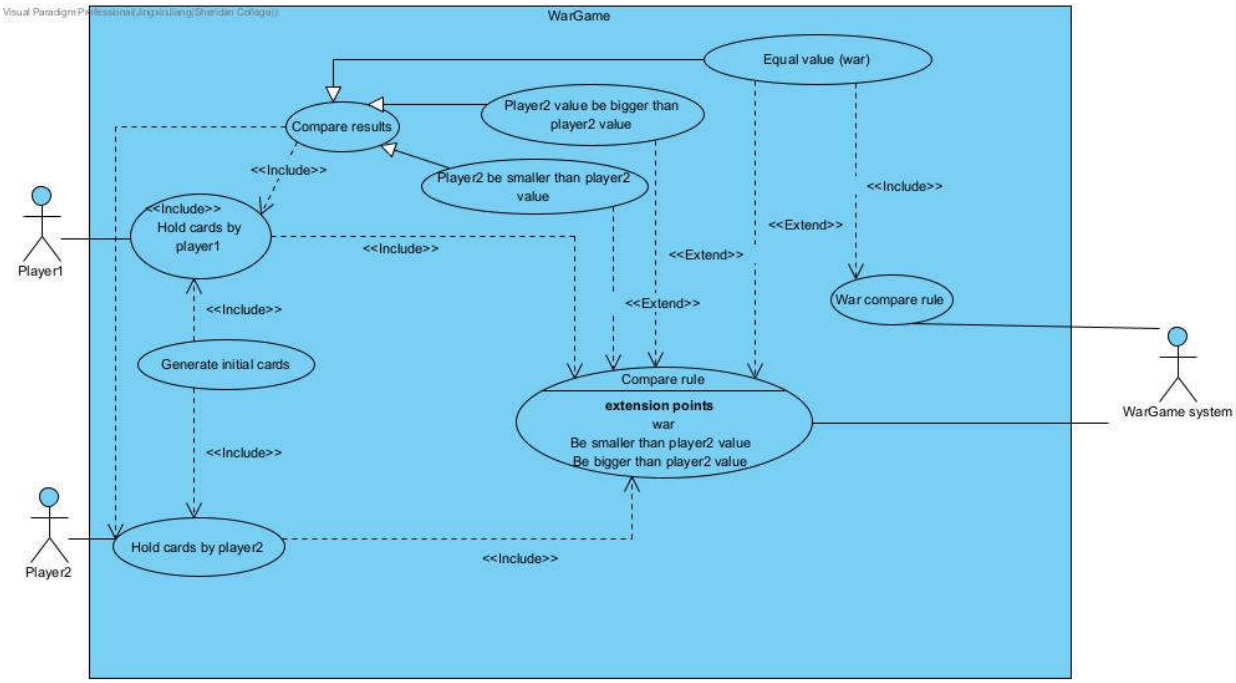

Figure 1 User case diagram

2. Class diagram:

1) Total class diagram

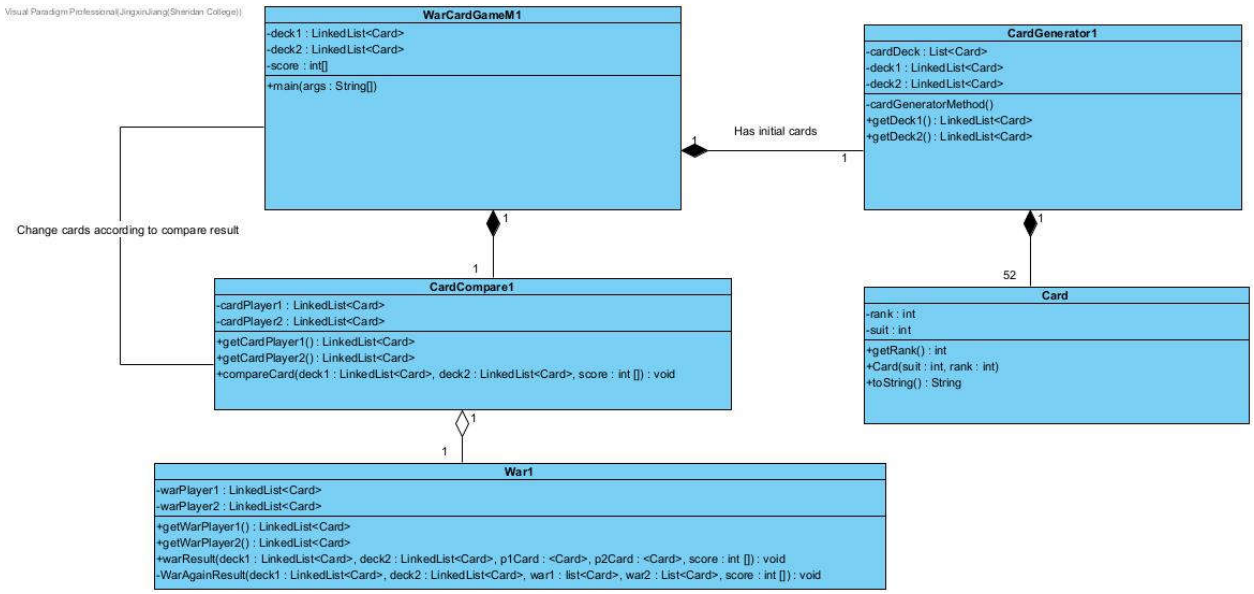

Figure 2 Total class diagram

2) Individual class diagram

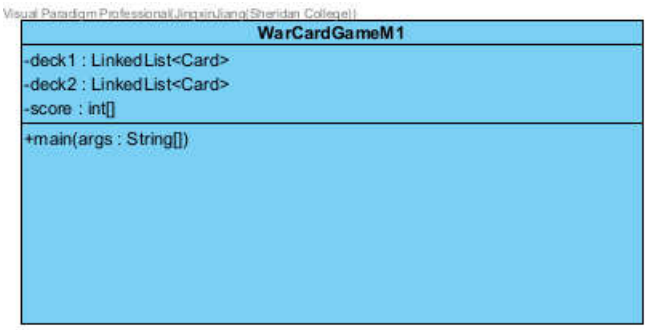

Figure 3 WarCardGameM1 class diagram

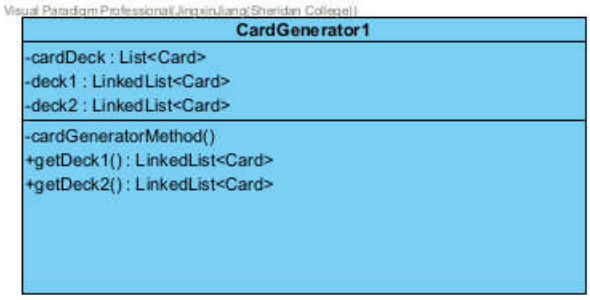

Figure 4 CardGenerator1 class diagram

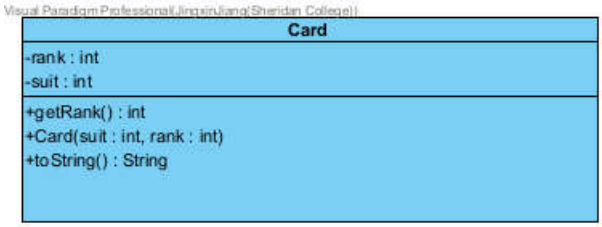

Figure 5 Card class diagram

Figure 6 CardCompare1 class diagram

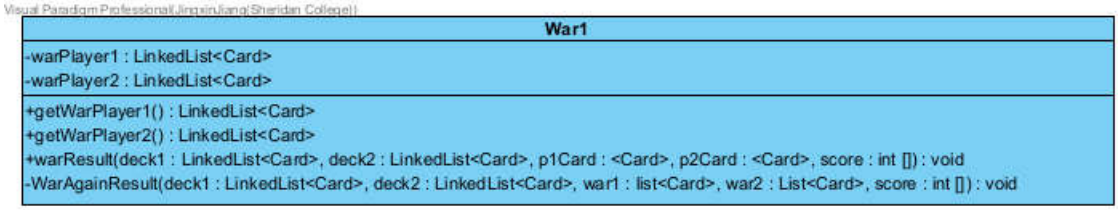

Figure 7 War1 class diagram

3 Design consideration:

Figure 2 shows that our WarCarGameM1 class composes with two parts, CardGenerator and card CardCompare1. It will record the score and show the result to players. We will use CardGenerator class create 52 initial cards and using CardCompare1 class compare the value and update the cards in LinkedList<Card> deck1 and LinkedList<Card> deck2. We will pass the reference when we invoke the public static compareCard method in CardCompare1. Therefore it can automatically update the cards in LinkedList<Card> deck1 and LinkedList<Card> deck2.
In CardGenerator class, 52 cards are created using private cardGeneratorMethod without return, then it will be random their order and update them to LinkedList<Card> deck1 and deck2, as shown in Figure 4. In class CardGenerator1, LinkedList<Card> deck1 and deck2 are private, and we will get them through their public getting method. In addition, as shown in Figure 5, Card class includes int type rank and suit. Suit represents 4 types of card including hearts, clubs,

diamonds, and spades. Rank places the order that ace is the highest rank followed by king all the way down to two. Method of toString will return a String including the information of card type and rank. Card creates using its constructor with the integer value of suit and rank. We will use public getRank method to get the rank of card to compare.

As shown in Figure 6, CardCompare1 includes public static compareCard method in order to decide the winner with bigger value of player 1's card and player 2's card, and it can be invoked by main method in WarCarGameM1 class. Inside it, there are three conditions including bigger, smaller, equal. For equal case, we will invoke public static warResult method in War1 class as shown in Figure 7. If two values are equal again, the private warAgainResult method will be invoked without return. In order to debug the code, we public the getting method of LinkedList<Card> type cardPlayer1, cardPlayer2, warPlayer1 and warPlayer2.

Detail as following:

a) CardGenerator1
   - Return all the used cards to the deck and shuffle them in a random sequence.
   - LinkedList<Card> Constructs an empty list.
   - Has initial cards for WarCardGameM1 using the Composition.
   - Multiplicity (CardGenerator1 Role A = 1) & (WarCardGameM1 Role B = 1).
   - cardDeck : List<Card> is a sorted collection of objects that allows for the storage of duplicate values.

b) Card
   - Multiplicity (Card Role A = 52) & (CardGenerator1 Role B = 1).
   - Card has 52 cards for CardGenerator1 using the Composition.
   - (-rank : int) The rank of a number in a list of numbers is returned.
   - For rank is Jack, Queen, King and Ace.
   - For suit is Spades, Hearts, Clubs and Diamonds.

c) CardCompare1

- Multiplicity (CardCompare1 Role A = 1) & (WarCardGameM1 Role B = 1).
- (CardCompare1 and WarCardGameM1) Change cards according to compare results.
- Using composition for the cards to compare.
- For war result of decks.
- For war result of scores.

d) War1
- Multiplicity (War1 Role A = 1) & (CardCompare1 Role B = 1).
- Using aggregation for CardCompare1.
- This is for War again results.
- This is for player 1 for War Battle.
- This is for player 2 for War Battle.

e) WarCardGameM1
- It is for player 1 to win the game.
- It is for player 2 to win the game.
- Showing scores for the players.
- Card generator to compare and war.
- For pressing 0 to quit the game and says game over.