**Project Title:** Black Jack

**Professor Name:** Amandeep Sidhu

**Date of Submission:** 17-March-2020

**Group Members Name:** Anshuk, Ekta Rao, Khushpreet Singh Brar, Vanshdeep Singh Sandhu (Leader)

## 1. Project Background and Description

As discussed in deliverable 1, the objective of the game is to beats the dealer's set of cards' total value by having a set of cards whose total value is more than dealer's set of cards. The cards have their own value except for the face cards and the ace. The face cards are all ten each, and the ace is either one or eleven depending on what the player wants. All the players are playing against the dealer.

**How to Play:**

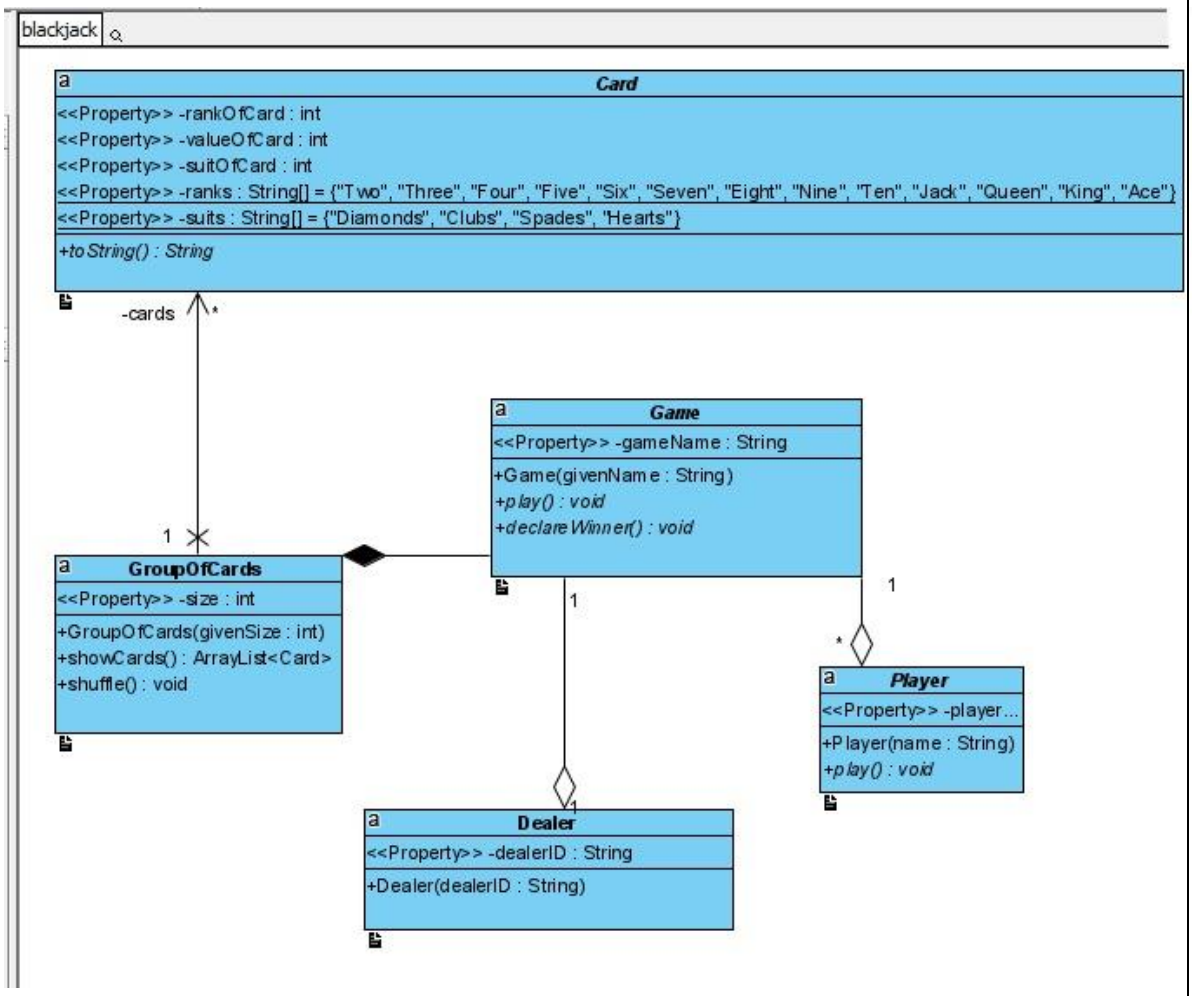
a. Select a person to be the dealer & 7 other people to be players. We need minimum one player to play the game with dealer.
b. Have the dealer shuffle the deck.
c. Now players have the option to bet according to the table minimum & maximum.
d. If the player thinks they have a good chance of beating the dealer, they should bet more.
e. Have the dealer deal two cards to each person.
f. Players' cards will be face up, dealer will have the first one laid face down and the second face up.
g. The dealer now gives each player starting to the left of him/herself the option of taking another card. They can keep taking cards until they are satisfied and decide to stand or until they go over 21, in which case they bust.
h. Once every player is done the dealer then hits until they reach at least 17 in which case he/she must stand or bust.

**GitHub Link:**

https://github.com/Sandhvan/BlackJack

## 2. Design Consideration

- **Class Diagram**




### Description

1. Card, Game, Player Class are Abstract Classes.
2. Card and GroupOfCards are associated with each other (GroupOfCards HAS-A Card).
3. GroupOfCards is a Part-Of Game class. This relationship is called Composition.

4. Player and Dealer are Made-Of Game class. This relationship is called Aggregation.
5. The attributes of Card class are suitOfCard, valueOfCard, and rankOfCard with all int data type. And String array ranks and suit. The abstract method toString ().
6. The attributes of GroupOfCards are size of data type int and cards an Array List of Card and these are kept as a private. The methods are GroupOfCards (), showCards, shuffle (). The multiplicity is 1 to many of this Class to Card class because total no of cards in one deck is 52.
7. The attributes of Player class are playerID with data type as a String and with constructor Player () and an abstract method play () of void data type.
8. The attributes of Dealer class are dealerID with data type as a String and with constructor Dealer ().
9. The attributes of Game class is game Name which is of String data type and the constructor Game () and two abstract methods play (), declareWinner (). The multiplicity of this class with Player class is 1 to many because 7 players can play a Black jack game at a time and have 1 to 1 multiplicity with Dealer as there can be only one dealer in a game.

- **Use Case**

## 1. Main Path

1.Player Buys Chips
2.Player Places a Bet
3. Dealer Deals Cards to Players
4. Each Player has 2 cards, both cards are face up
5. Dealer has one card up and one face down.
6. Dealer starts at the person on their left
7. Player play his hand
8.First two cards are acceptable, player stand and dealer will move to next player

## 2. Alternate Path(a)

If player like to have more cards
1. The dealer will deal more cards to player
2. One at a time
3. Player choose to bust
4. Player went over 21

## 3. Alternate Path(b)

If player have hand that is useful to them but they need additional cards
1. Player ask for additional card
2. Dealer deals only 1 additional card
3. Player continues to play

## 4. Alternate Path(c)

Insurance given by dealer (It is a side bit offered only when dealer has an Ace as an up-card)
1. Dealer Deals Cards
2. Dealer has Ace as an up-card
3. Dealer offers Insurance

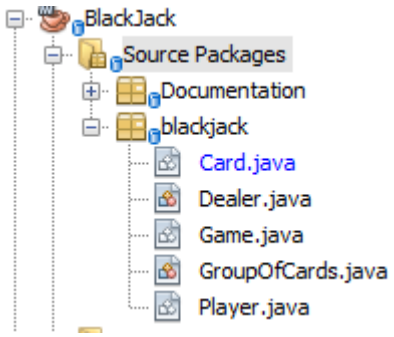
4. If player think, dealer has a blackjack, he starts betting on it.
5. Dealer don't have ten under up card
6. Dealer takes any insurance bet that were made
7. Game Continues

- **OPP Principles Brief**
  - Encapsulation: Hiding data from the outside sources to prevent misuse. (Achieved by adding access modifiers like private, public etc.)

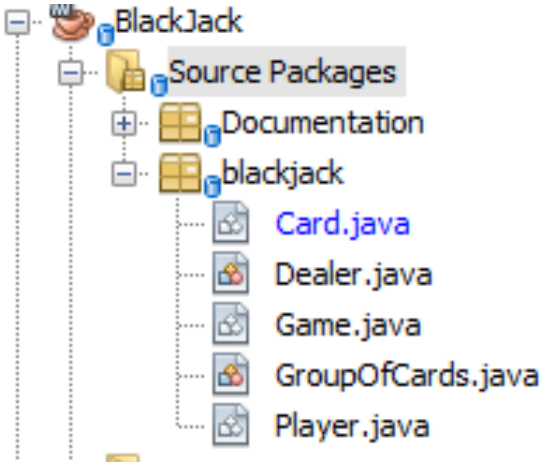
```
//The group of cards, stored in an ArrayList
private ArrayList <Card> cards;
private int size;//the size of the grouping
```

```
public abstract class Game
{
    private final String gameName;//the title of the game
    private ArrayList <Player> players;// the players of the game
```

  - Delegation: Separation of concern so that a class can do its responsivities properly (Achieved by making different number of classes)




  - Cohesion: refers all about how a single class is designed. (Making different classes for different function)



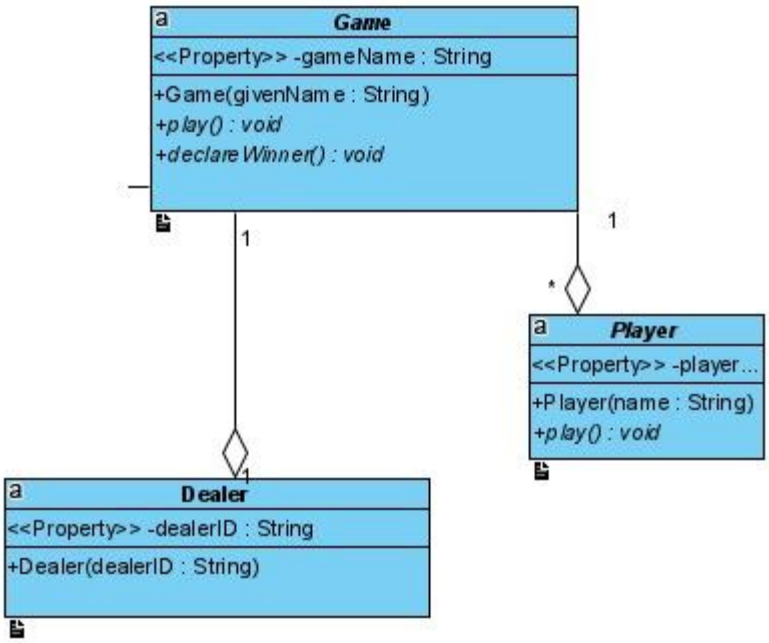

Card as a model, Player as view, Dealer, Game and GroupOfCards as controller.

- Coupling: Inter-Connection between the classes.

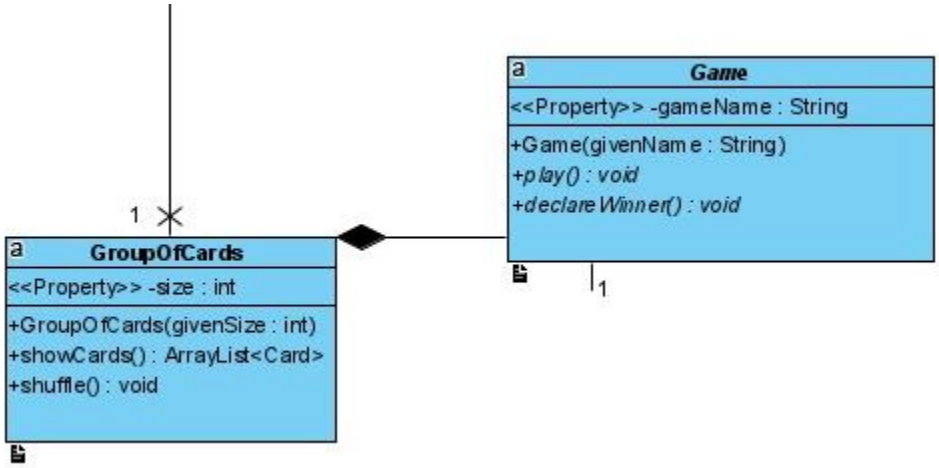
```
public abstract class Card {

    private int rankOfCard;//represents the rank of a c
    private int suitOfCard;//represents the suit of a c
    private int valueOfCard;//represents the value of a
public class GroupOfCards
{

    //The group of cards, stored in an ArrayList
    private ArrayList <Card> cards;
    private int size;//the size of the grouping
```

- Inheritance: is a mechanism in which one object acquires all the properties and behaviors of a parent object.

- Aggregation: implies a relationship where the child can exist independently of the parent




- Composition: implies a relationship where the child cannot exist independent of the parent.




- Flexibility/Maintainability: The ease or difficulty with which a software system can be modified is known as its maintainability/Flexibility.