

객체지향 설계 Term Project

#1

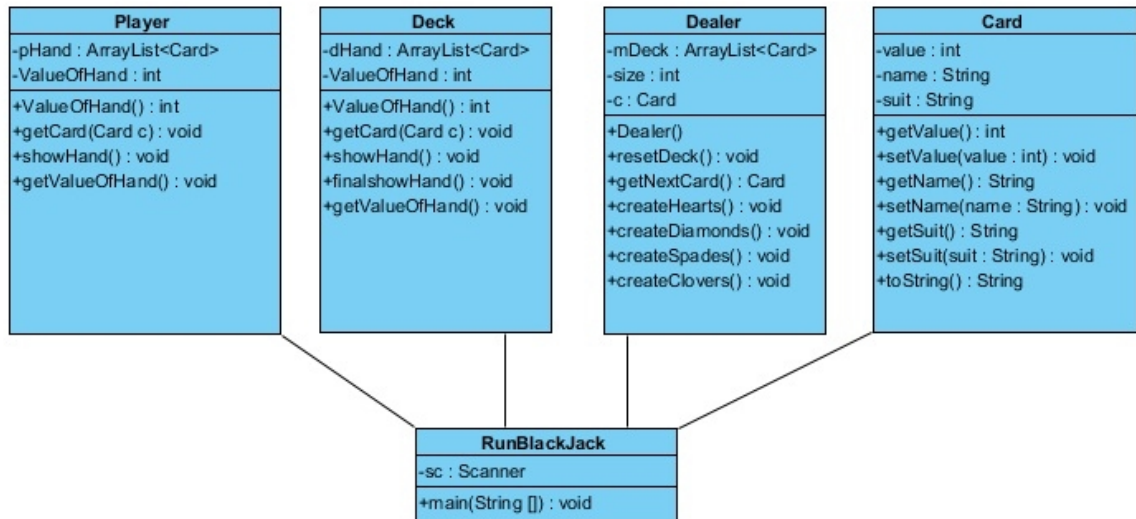
-BlackJack

201102435

02반

박민수

# 1) Class Diagram



## 2) 소스코드

### a) Dealer Class

```
import java.util.ArrayList;

public class Dealer {
    private ArrayList<Card> dHand = new ArrayList<Card>();
    private int ValueOfHand = 0;

    public int ValueOfHand(){
        return this.ValueOfHand;
    }

    // 카드를 드로우하는 메소드
    public void getCard(Card c){
        dHand.add(c);
    }

    // 딜러의 핸드에 있는 카드를 보여주는 메소드 (첫번째 카드는 숨겨짐)
    public void showHand(){
        System.out.print("딜러의 현재 핸드 (첫번째 카드는 숨겨짐) : ");
        for(int i =1 ; i<dHand.size(); i++){
            System.out.print(dHand.get(i)+" ");
        }
    }

    // 최종 딜러의 핸드에 있는 모든 카드를 보여주는 메소드
    public void finalshowHand(){
        System.out.print("딜러의 최종핸드 : ");
        for(int i =0 ; i<dHand.size(); i++){
            System.out.print(dHand.get(i)+" ");
        }
    }
}
```

```

//딜러의 핸드에있는 카드의 값을 모두 합하는 메소드
public void getValueOfHand(){
    int ValueOfHand = 0;
    int findAce = 0;
    for(int i = 0; i<dHand.size();i++){
        if (dHand.get(i).getName().equals("ACE")) {
            findAce++;
            continue;
        }
        ValueOfHand += dHand.get(i).getVlaue();
    }
    // 딜러의 핸드에서 ACE는 무조건 11로 계산한다.
    while (findAce > 0){
        ValueOfHand += 11;
    }
    this.ValueOfHand = ValueOfHand;
}

```

## b) Player Class

```
import java.util.ArrayList;

public class Player {
    private ArrayList<Card> pHand = new ArrayList<Card>();
    private int ValueOfHand = 0;

    public int ValueOfHand(){
        return this.ValueOfHand;
    }

    //카드를 핸드로 받아오는 메소드
    public void getCard(Card c){
        pHand.add(c);
    }

    // 플레이어의 핸드에 있는 카드를 보여주는 메소드
    public void showHand(){
        System.out.print("플레이어의 현재 핸드 : ");
        for(int i =0 ; i<pHand.size(); i++){
            System.out.print(pHand.get(i)+" ");
        }
    }

    //플레이어의 핸드에있는 카드의 값을 모두 합하는 메소드
    public void getValueOfHand(){
        int ValueOfHand =0;
        int findAce = 0;
        for(int i = 0; i<pHand.size();i++){
            if (pHand.get(i).getName().equals("ACE")) {
                findAce++;
                continue;
            }
            ValueOfHand += pHand.get(i).getVlaue();
        }
        // 플레이어의 핸드에서 ACE는 나머지 핸드의 카드의 합이 10이하일경우 11로, 아닐경우 1로 계산한다.
        while(findAce > 0){
            if(ValueOfHand <= 10 )
                ValueOfHand += 11;
            else
                ValueOfHand += 1;
            findAce--;
        }
        this.ValueOfHand = ValueOfHand;
    }
}
```

## c) Deck Class

```
⊕ import java.util.ArrayList;

public class Deck {
    private ArrayList<Card> mDeck = new ArrayList<Card>();
    private int size = 0;
    private Card c = new Card();

    //생성자
    Deck(){
        createHearts();
        createDiamonds();
        createSpades();
        createClovers();

        Collections.shuffle(mDeck);
    }

    // 덱을 초기화 시키는 메소드.
    public void resetDeck(){
        mDeck = null;
        size = 0;

        createHearts();
        createDiamonds();
        createSpades();
        createClovers();

        Collections.shuffle(mDeck);
    }

    //덱에 다음 번카드를 딜러와 플레이어에게 나누어주는 메소드
    public Card getNextCard(){
        Card rCard = mDeck.get(52-size);
        size--;
        return rCard;
    }
}
```

```

private void createHearts(){
    for(int i =1 ; i<14 ; i++){
        c = new Card();
        c.setSuit("Hearts");
        if(i == 1){
            c.setName("ACE");
            c.setVlaue(i);
        }
        else if(i == 11){
            c.setName("J");
            c.setVlaue(10);
        }
        else if(i == 12){
            c.setName("Q");
            c.setVlaue(10);
        }
        else if(i == 13){
            c.setName("K");
            c.setVlaue(10);
        }
        else{
            c.setName(""+i+"");
            c.setVlaue(i);
        }
        mDeck.add(c);
        size++;
    }
}

```

```

private void createDiamonds(){
    for(int i =1 ; i<14 ; i++){
        c = new Card();
        c.setSuit("Diamonds");
        if(i == 1){
            c.setName("ACE");
            c.setVlaue(i);
        }
        else if(i == 11){
            c.setName("J");
            c.setVlaue(10);
        }
        else if(i == 12){
            c.setName("Q");
            c.setVlaue(10);
        }
        else if(i == 13){
            c.setName("K");
            c.setVlaue(10);
        }
        else{
            c.setName(""+i+"");
            c.setVlaue(i);
        }
        mDeck.add(c);
        size++;
    }
}

```

```

private void createSpades(){
    for(int i =1 ; i<14 ; i++){
        c = new Card();
        c.setSuit("Spades");
        if(i == 1){
            c.setName("ACE");
            c.setVlaue(i);
        }
        else if(i == 11){
            c.setName("J");
            c.setVlaue(10);
        }
        else if(i == 12){
            c.setName("Q");
            c.setVlaue(10);
        }
        else if(i == 13){
            c.setName("K");
            c.setVlaue(10);
        }
        else{
            c.setName(""+i+"");
            c.setVlaue(i);
        }
        mDeck.add(c);
        size++;
    }
}

private void createClovers(){
    for(int i =1 ; i<14 ; i++){
        c = new Card();
        c.setSuit("Clovers");
        if(i == 1){
            c.setName("ACE");
            c.setVlaue(i);
        }
        else if(i == 11){
            c.setName("J");
            c.setVlaue(10);
        }
        else if(i == 12){
            c.setName("Q");
            c.setVlaue(10);
        }
        else if(i == 13){
            c.setName("K");
            c.setVlaue(10);
        }
        else{
            c.setName(""+i+"");
            c.setVlaue(i);
        }
        mDeck.add(c);
        size++;
    }
}

```



## d) Card Class

```
package BlackJack;

public class Card {
    private int vlaue;
    private String name;
    private String suit;

    //value getter & setter
    public int getVlaue() {
        return vlaue;
    }
    public void setVlaue(int vlaue) {
        this.vlaue = vlaue;
    }

    //name getter & setter
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    //Suit getter & setter
    public String getSuit() {
        return suit;
    }
    public void setSuit(String suit) {
        this.suit = suit;
    }

    public String toString(){
        return suit+"/"+name;
    }
}
```

## c) RunBlackJack Class

```
public class RunBlackJack {  
  
    private static Scanner sc;  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
  
        Deck deck = new Deck();  
        Dealer dealer = new Dealer();  
        Player player = new Player();  
        int select = 0;  
        sc = new Scanner(System.in);  
  
        System.out.println("***** 게임을 시작 합니다. *****");  
  
        System.out.println("카드를 두장씩 배분합니다.");  
        dealer.getCard(deck.getNextCard());  
        player.getCard(deck.getNextCard());  
        dealer.getCard(deck.getNextCard());  
        player.getCard(deck.getNextCard());  
        dealer.getValueOfHand();  
        player.getValueOfHand();  
  
        dealer.showHand();  
        System.out.println();  
        player.showHand();  
        System.out.println();  
        System.out.println("플레이어의 핸드의 합 : "+ player.ValueOfHand());  
    }  
}
```

- Deck, Player, Dealer 객체를 생성하고 플레이어와 딜러에게 각각 2장씩 배분한 다음 딜러와 플레이어의 핸드를 공개한다. 이때 딜러의 첫 번째 카드는 감춰져있다.

```

if(dealer.ValueOfHand() == 21 && player.ValueOfHand() == 21){
    dealer.finalshowHand();
    System.out.println();
    System.out.println("딜러의 핸드의 합 :"+ dealer.ValueOfHand());
    System.out.println("Dealer & Player are BlakJack!");
    System.out.println("무승부입니다!");
    return;
}
else if(dealer.ValueOfHand() == 21){
    dealer.finalshowHand();
    System.out.println();
    System.out.println("딜러의 핸드의 합 :"+ dealer.ValueOfHand());
    System.out.println("Dealer is BlakJack!");
    System.out.println("딜러가 승리하였습니다!");
    return;
}
else if(player.ValueOfHand() == 21){
    dealer.finalshowHand();
    System.out.println();
    System.out.println("딜러의 핸드의 합 :"+ dealer.ValueOfHand());
    System.out.println("Player is BlakJack!");
    System.out.println("플레이어가 승리하였습니다!");
    return;
}

```

- 딜러와 플레이어의 블랙잭 유무를 판별한다. 둘 다 블랙잭일 경우 무승부이고, 나머지는 블랙잭인 쪽이 승리하게 된다.

```

while(true){
    if(player.ValueOfHand()<21){
        System.out.println("다음 진행할 행동을 선택합니다. (1. Stand 2.Hit)");
        select = sc.nextInt();
    }else{
        select = 1;
    }
    if (select == 1){
        System.out.println("[Stand]");
        System.out.println("딜러의 카드를 제공합니다.");
        dealer.finalshowHand();
        System.out.println();
        while(dealer.ValueOfHand() < 17){
            System.out.println("딜러의 핸드의 합이 16이하여서 한장을 Hit합니다.");
            dealer.getCard(deck.getNextCard());
            dealer.getValueOfHand();
            dealer.finalshowHand();
            System.out.println();
        }
        player.showHand();
        System.out.println();

        System.out.println("플레이어의 핸드의 합 :"+ player.ValueOfHand());
        System.out.println("딜러의 핸드의 합 :"+ dealer.ValueOfHand());
    }
}

```

```

        if(dealer.ValueOfHand()>21 && player.ValueOfHand()>21){
            System.out.println("Dealer & Player are Bust!");
            System.out.println("딜러가 승리했습니다!");
        }
        else if(dealer.ValueOfHand()>21 && player.ValueOfHand()<21){
            System.out.println("Dealer is Bust!");
            System.out.println("플레이어가 승리했습니다!");
        }
        else if(dealer.ValueOfHand()<21 && player.ValueOfHand()>21){
            System.out.println("Player is Bust!");
            System.out.println("딜러가 승리했습니다!");
        }
        else if(dealer.ValueOfHand() > player.ValueOfHand()){
            System.out.println("딜러가 승리했습니다!");
        }
        else if(dealer.ValueOfHand() < player.ValueOfHand()){
            System.out.println("플레이어가 승리했습니다!");
        }
        else{
            System.out.println("무승부입니다!");
        }
        return;
    }
    if (select == 2){
        System.out.println("[HIT]");

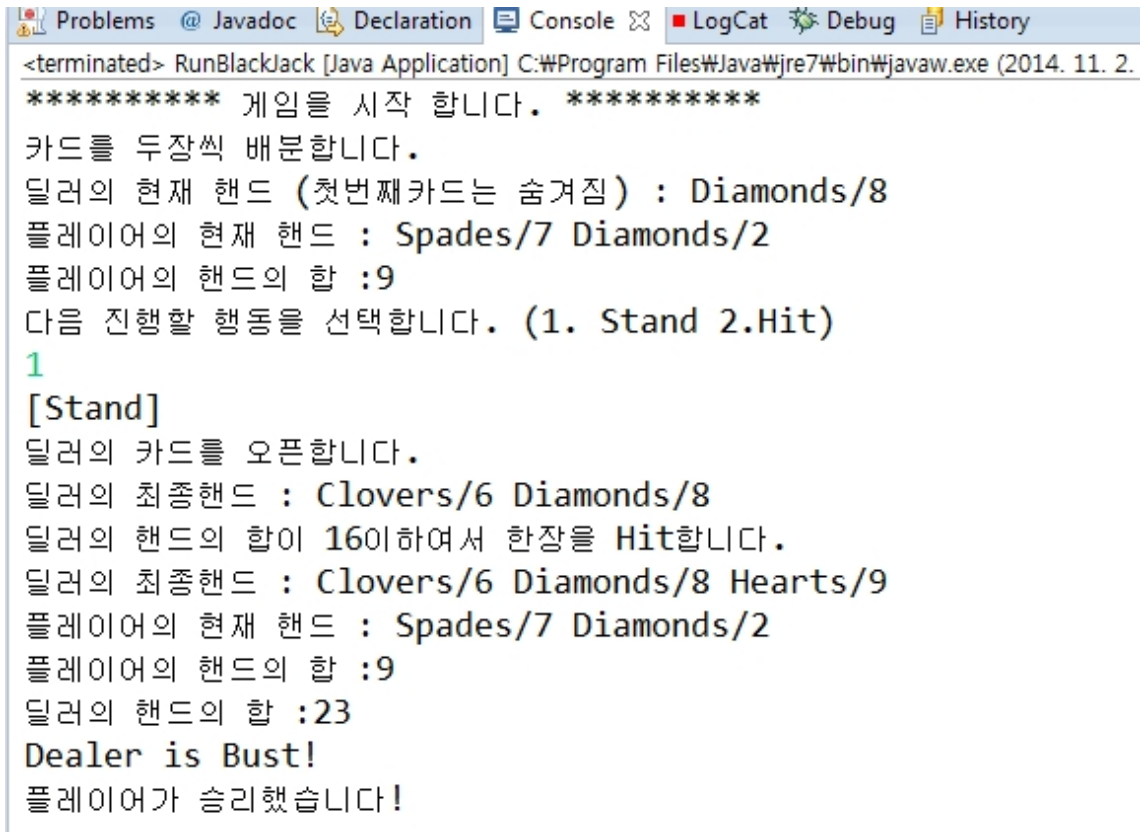
        player.getCard(deck.getNextCard());
        player.getValueOfHand();

        dealer.showHand();
        System.out.println();
        player.showHand();
        System.out.println();
        System.out.println("플레이어의 핸드의 합 : "+ player.ValueOfHand());
    }
}

```

- 블랙잭이 아니라면 플레이어는 Stand 할 것인지 Hit할 것인지 선택한다. Stand를 선택할 경우, 플레이어는 카드를 더 이상 받지 않고 딜러의 경우 핸드의 합이 16이하이면 17이상이 될 때까지 카드를 한 장씩 받는다. Hit를 선택할 경우 플레이어는 카드를 한 장 받게 되고 만약 더 받아서 버스트가 된다면 즉시 Stand 단계로 넘어가고 딜러의 턴으로 넘어간다. 그 이후 딜러와 플레이어의 핸드의 합을 비교하게 되는데 이때 양쪽 모두 버스트가 된 경우는 딜러가 승리하게 되며, 나머지의 경우는 버스트를 하지 않았거나 핸드의 합이 더 큰 쪽이 승리하게 된다.

### 3) 실행화면



```
<terminated> RunBlackJack [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (2014. 11. 2.  
***** 게임을 시작 합니다. *****  
카드를 두장씩 배분합니다.  
딜러의 현재 핸드 (첫번째카드는 숨겨짐) : Diamonds/8  
플레이어의 현재 핸드 : Spades/7 Diamonds/2  
플레이어의 핸드의 합 :9  
다음 진행할 행동을 선택합니다. (1. Stand 2.Hit)  
1  
[Stand]  
딜러의 카드를 엽니다.  
딜러의 최종핸드 : Clovers/6 Diamonds/8  
딜러의 핸드의 합이 16이하여서 한장을 Hit합니다.  
딜러의 최종핸드 : Clovers/6 Diamonds/8 Hearts/9  
플레이어의 현재 핸드 : Spades/7 Diamonds/2  
플레이어의 핸드의 합 :9  
딜러의 핸드의 합 :23  
Dealer is Bust!  
플레이어가 승리했습니다!
```

- 실행결과화면이다. 처음 플레이어는 스페이드7과 다이아몬드 2를 가지고 있어서 합이 9이다. 이때 Stand를 선택하면 딜러의 모든 카드를 엽니다, 이때 딜러의 핸드의 합이 16이하이면 17이상일 때까지 카드를 받는다. 최종적으로 딜러는 클로버6, 다이아몬드8, 하트9를 가지게 되고 이때 핸드의 합은 23이 되어 버스트가 되기 때문에 플레이어가 승리하게 된다.