

객체지향프로그래밍 - 4 주차 실습 답안지

성명: 조우연

학과: 수학과

학번: 201921195

※ 본 실습활동지에 대한 보고서나 코드를 작성함에 있어 교재나 강의노트를 제외한 다른 자료로부터 일부 또는 전체를 복사하였습니까? 예() 아니오(O)

1번

가) CardTest1.java 실행

(CardTest1.java는 실습활동지의 코드와 동일하게 따른다.)

- CardTest1.java실행결과

```
Console
<terminated> CardTest1 [Java Application] C:\WProgr
Suit : Spade, Rank : 1
Suit : Heart, Rank : 2
```

Card 클래스로 정의된 c1,c2에 public Card(int r, String s)메소드를 호출해 각각의 rank와 suit을 (1,"Spade"), (2,"Heart")로 주었다. 따라서 c1,c2의 rank와 suit은 각각 위와 같이 설정이 되고, c1과 c2의 suit과 rank를 반환하는 메소드인 getRank()과 getSuit()을 이용해 각각의 suit과 rank를 불러와 출력을 하면 각각의 rank와 suit정보를 왼쪽 이미지와 같이 출력된다.

나) Card의 getRank()와 getSuit()를 실습활동지에 따라 수정 후, CardTest1.java를 실행

- 수정된 Card로 CardTest1.java실행결과

```
Console
<terminated> CardTest1 [Java Applicati
Suit : Spade, Rank : 1
Suit : Heart, Rank : 2
```

(가)와 같은 결과를 출력한다.

this.은 인스턴스의 자기 자신을 의미한다. this.은 생략을 해도 되지만, 생략을 하더라도 모든 instance변수에는 this.이 붙은 형태로 프로그램이 작동한다. instance변수의 이름을 유일하게 설정해서 프로그램을 실행할 때 instance변수가 명확하다면 큰 문제가 없지만, 전역변수와 메소드, 혹은 생성자의 매개변수의 이름이 인스턴스변수와 동일할 때에는 컴퓨터가 프로그램을 실행할 때 잘못된 결과를 도출할 수 있으므로 local변

수와 구분하기 위해 this.를 사용한다. 따라서 수정전의 Card클래스와 수정후의 Card클래스의 rank와 suit은 모두 instance변수이므로, this.를 사용하지 않더라도 this.이 붙은 형태로 프로그램이 작동하기 때문에 동일한 결과를 갖는다.

다) Card class에 toString() method를 추가 후, CardTest2.java를 실행
(CardTest2.java는 실습활동지의 코드와 동일하게 따른다.)

- toString() 메소드	- CardTest2.java 실행결과
<div>Card.java CardTest1.java CardTest2.java</div> <pre>18 } 19 public String toString() 20 { 21 String Rank=""; 22 switch (this.rank) { 23 case 1 : 24 Rank="Ace"; 25 break; 26 case 2 : 27 Rank="2"; 28 break; 29 case 3 : 30 Rank="3"; 31 break; 32 case 4 : 33 Rank="4"; 34 break; 35 case 5 : 36 Rank="5"; 37 break; 38 case 6 : 39 Rank="6"; 40 break; 41 case 7 : 42 Rank="7"; 43 break; 44 case 8 : 45 Rank="8"; 46 break; 47 case 9 : 48 Rank="9"; 49 break; 50 case 10 : 51 Rank="10"; 52 break; 53 case 11 : 54 Rank="Jack"; 55 break; 56 case 12 : 57 Rank="Queen"; 58 break; 59 case 13 : 60 Rank="King"; 61 break; 62 } 63 return (Rank+" of "+suit); 64 }</pre>	<div>Console</div> <div><terminated> CardTest2 [Java Applicati</div> <div>Ace of Spade 2 of Heart</div>

라) Card class에 compareTo() method를 추가 후, CardTest3.java를 실행
(CardTest3.java는 실습활동지의 코드와 동일하게 따른다.)

- compareTo() method	- CardTest3.java 실행결과
<pre> 64 } 65 public int compareTo(Card other) 66 { 67 if (this.rank > other.rank) 68 { 69 if (other.rank == 1) 70 return -1; 71 else 72 return 1; 73 } 74 else if (this.rank < other.rank) 75 { 76 if(this.rank == 1) 77 return 1; 78 else 79 return -1; 80 } 81 } 82 else 83 { 84 if(this.suit==other.suit) 85 return 0; 86 else 87 { 88 if(this.suit=="Spade") 89 return 1; 90 else if(other.suit=="Spade") 91 return -1; 92 else { 93 if(this.suit=="Diamond") 94 return 1; 95 else if(other.suit=="Diamond") 96 return -1; 97 } 98 else { 99 if(this.suit=="Heart") 100 return 1; 101 else if(other.suit=="Heart") 102 return -1; 103 } 104 } 105 } 106 } 107 } 108 return 0; 109 } 110 } </pre>	<p>일반적으로 rank가 더 크면 이긴다.(ACE제외)</p>
	<pre> Console <terminated> CardTest3 [Java Application] 2 of Diamond 8 of Heart 8 of Heart wins 2 of Diamond Console <terminated> CardTest3 [Java Application] 3 of Spade 6 of Heart 6 of Heart wins 3 of Spade </pre>
	<p>ACE는 다른 rank보다 값이 가장 크다.</p>
	<pre> Console <terminated> CardTest3 [Java Application] Ace of Diamond 8 of Club Ace of Diamond wins 8 of Club </pre>
	<p>rank가 같으면 suit에 따라 승패가 결정됨</p>
	<pre> Console <terminated> CardTest3 [Java Application] 2 of Diamond 2 of Club 2 of Diamond wins 2 of Club </pre>
	<p>같은 패(비겼을 때)</p>
	<pre> Console <terminated> CardTest3 [Java Application] 2 of Club 2 of Club 2 of Club ties to 2 of Club </pre>

2번

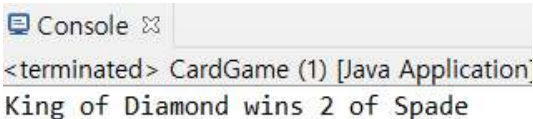
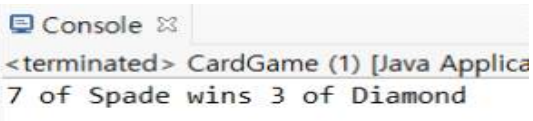

가) CardDeck class에서 Card 52장을 생성해 초기화하는 constructor와 shuffle(), dealCard() method들을 완성하시오.

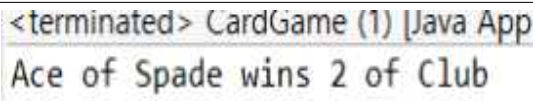
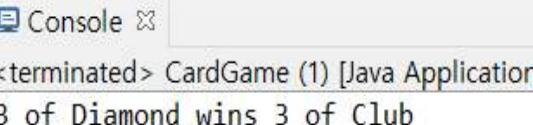
```
//CardDeck.java
public class CardDeck {
    private static final int N = 52;
    private Card[] deck;
    private int top = 0;
    static final String[] cardSuits = {"Spade", "Diamond", "Heart", "Club"};

    public CardDeck()
    {
        int k = 0;
        deck = new Card[N];
        for(int i=0; i<4; i++)
            for(int j=1; j<14; j++)
                deck[k++] = new Card(j,cardSuits[i]);
    }
    public void shuffle()
    {
        int r=0, i=0;
        Card temp;

        for(i=0; i<N ; i++) {
            r = (int)(Math.random()*52);
            temp = deck[i];
            deck[i] = deck[r];
            deck[r] = temp;
        }
    }
    public Card dealCard()
    {
        return deck[top++];
    }
}
```

나) Card class와 CardDeck class를 이용하는 CardGame.java 실행 및 결과설명
(CardGame.java는 실습활동지의 코드와 동일하게 따른다.)

- CardTest3.java 실행결과
일반적으로 rank가 더 크면 이긴다.(ACE제외)




ACE는 다른 rank보다 값이 가장 크다.

rank가 같으면 suit에 따라 승패가 결정됨


결과설명 :

CardGame.java는 두명의 player에게 임의로 섞인 카드를 한 장씩 나누어주고, 카드의 값이 큰 player를 승자로 결정하고, 두 player가 뽑은 카드의 정보와 게임결과(승패)를 출력하는 프로그램이다. 이때, 카드의 값은 rank가 클수록(단 Ace>King), rank가 같다면 Suit이 클수록(Spade > Diamond > Heart > Club) 카드의 값이 크다고 본다.

따라서 위의 표와 실행결과를 보면 볼 수 있듯이, 두 player의 rank가 더 높은 사람이 이기고, rank가 같다면 Suit이 큰 사람이 이기는 것을 볼 수 있다. 단, CardDeck class안의 dealCard() method가 카드를 중복되지 않게 두 player에게 카드를 나누어주므로, 비기는 경우는 없다.