Java Programming: Assignment 1

April 30, 2020

1 포커 카드 게임 만들기

포커 게임의 카드는 다이아몬드, 클로버, 하트, 스페이스 4가지의 문양과 각 문양 (shape) [S(스페이드), H(하트), D(다이아몬드, C(클로버)] 마다 13가지의 숫자 [1(A), 2, 3, 4, 5, 6 ,7, 8, 9, 10, 11(J), 12(Q), 13(K)] 로 이루어져 총 52가지의 카드가 있다 (조커 카드는 제외한다). 포커 게임 프로그램은 사용자와 사용자가 아닌 플레이어(경쟁자) 간의 대결 구도로 진행된다. 게임을 시작하면 프로그램은 사용자와 경쟁자에게 각각 2장, 2장, 1장 순으로 총 3번에 걸쳐 5개의 카드를 랜덤하게 분배한다. 본 문제에서는 주어진 카드에 대해 랭크 비교를 통해 승자와 패자를 결정하는 프로그램을 작성해야 한다.

1.1 포커 게임의 족보 (Ranks of hand)

프로그램에서 사용하는 포커 게임의 족보 (Ranks of hand)은 다음과 같다.

랭크	이름	조합
5	포카드 (Four of a Kind)	같은 숫자 4장이 나온 경우
4	플러쉬 (Flush)	같은 문양의 카드가 5장 나오면 플러쉬
3	N 트리플 (Three of a kind)	같은 숫자의 카드 N이 3장 나오면 트리플
2	N 원페어 (One Pairs)	같은 숫자의 카드 N이 2장 나오면 원페어
1	N 탑 (Top)	위에 해당되지 않는, 즉 아무것도 아닌 카드

또한, 랭크 1, 2, 3 에 한하여 플레이어의 카드패에서 동일 랭크를 갖는 조합이 2개 이상 가능한 경우에는 다음 우선순위에 따라 카드 조합을 정한다.

$$1(\mathrm{A}) > \mathrm{K}(13) > 12(\mathrm{Q}) > 11(\mathrm{J}) > 10 > 9 > 8 > 7 > 6 > 5 > 4 > 3 > 2$$

위의 우선순위에 따라 플레이어의 카드 중 가장 순위가 높은 카드 조합을 제시하고, 해당 조합으로 N을 결정하면 된다 (각 문양에 대한 우선 순위는 없다)

또한, 플레이어 카드패에서 2개 이상 족보가 가능한 경우에는 가장 랭크가 높은 족보를 최종 족보로 결정한다.

다음은 사용자 및 컴퓨터 플레이어간 족보 비교후 승패를 결정하는 예시이다.

사용자	경쟁자(컴퓨터)	결과
A 트리플	9 트리플	사용자 승
포카드	포카드	무승부
포카드	플러쉬	사용자 승
2 탑	포카드	경쟁자 승
10 원페어	10 원페어	무승부
4 원페어	K 원페어	경쟁자 승

1.2 프로그램의 구동 방식

매 카드 지급 시 마다 사용자는 '**GO**' 와 '**DIE**' 두 가지 옵션중 하나의 옵션을 입력해야 한다.

- 옵션 1 (**GO**) : 두 플레이어 모두 다음에 지급될 카드를 받는다. 마지막 카드를 지급 받은 후 GO 옵션을 입력한 경우 각 플레이어의 카드 조합을 비교해 승자와 패자를 결정한다.
- 옵션 2 (**DIE**) : 각 플레이어에게 현재까지 지급된 카드를 초기화 하고, 해당 게임을 다시 시작한다.

매 카드 지급 시 마다 사용자가 지금까지 지급받은 카드를 출력한다 (경쟁자의 카드는 출력하지 않는다). 사용자가 마지막 카드 지급 시에 **GO** 옵션을 입력한 경우 (총 3번의 GO 옵션을 입력한 경우) 두 플레이어에 대해 <**플레이어, 카드 5장, 최종 족보**>를 출력한다. 이후, 게임의 승자를 출력하고 게임을 계속하려면 **y**를 입력하고 그렇지 않으면 n 을 입력 하도록 한다.

1.3 프로그램 요구사항

1. `Card.java` 파일은 수정하지 않고 그대로 사용

```java
public class Card {
    private int rank;
    private String suit;

    public void setCard(String suit, int rank) {
        this.suit = suit;
        this.rank = rank;
    }

    public String getSuit() { return suit; }
    public int getRank() { return rank; }
}
```

Code 1: Card.java

2. 아래의 `Player` 클래스를 상속받아 '사용자'와 '경쟁자'를 위한 클래스를 만들어 사용 (Player 클래스는 수정하여 사용)

```java
public class Player {
    private String name;
    private Card[] cards;

    public Player() {
        // 생성자 코드
    }
}
```

Code 2: Player.java

3. 아래의 `Dealer` 클래스를 수정하여 각 플레이어에게 카드를 나누어 주는 `deal` 메소드와 각 플레이어의 점수를 계산하는 `score`메소드를 작성하여 `Dealer`클래스를 완성해서 사용

```java
public class Dealer {
    private String[] suits= {"S", "H", "D", "C"};
    private int numcard;

    public Dealer() { numcard = 13;}

    //
}
```

Code 3: Dealer.java

2 차량 예약 관리 프로그램 만들기

본 문제에서는 키보드로부터 사용자 정보를 입력 받고 난 후 예약 정보를 입력 받아서, 차량과 좌석을 관리하는 프로그램을 작성한다. 사용자 계정을 담당하는 `AccountManager` 클래스, `Vehicle`, `Bus`, `Taxi` 클래스와 실행 결과를 보고, `ReserveManager` 클래스의 메소드를 작성하시오 [제출파일 : `AccountManager.java`, `VihicleManager.java`].

2.1 프로그램 실행 예

- **[회원 가입 실행화면]**

```
--------------------차량 예약 관리 프로그램 입니다--------------------
1. 회원가입 | 2. 로그인 | 3. 차량예약 관리| 4. 로그아웃 | 5. 종료
-----------------------------------------------------------
선택>> 1
--------------------회원가입--------------------
ID :201955057
이름 :홍길동
Password :1234567m
```

- **[로그인 수행화면]**

```
--------------------차량 예약 관리 프로그램 입니다--------------------
1. 회원가입 | 2. 로그인 | 3. 차량예약 관리| 4. 로그아웃 | 5. 종료
-----------------------------------------------------------
선택>> 2
--------------------로그인--------------------
ID: 201955057
비밀번호: 1234567m
```

- **[차량 추가 실행화면]**

```
--------------------차량 예약 관리 프로그램 입니다--------------------
1. 회원가입 | 2. 로그인 | 3. 차량예약 관리| 4. 로그아웃 | 5. 종료
-----------------------------------------------------------
선택>> 3
```

```
-------------------------------------------------------
1. 차량 추가 | 2. 차량 예약 | 3. 차량 목록 | 4. 예약 현황 | 5. 종료
-------------------------------------------------------
<선택> 1
--------------------
1. 택시 | 2. 버스
--------------------
<선택> 1
차량 번호 : 12가1234
-------------------------------------------------------
1. 차량 추가 | 2. 차량 예약 | 3. 차량 목록 | 4. 예약 현황 | 5. 종료
-------------------------------------------------------
<선택> 1
--------------------
1. 택시 | 2. 버스
--------------------
<선택> 2
차량 번호 : 56나5678
```

- **[차량 예약 실행화면]**

```
-------------------------------------------------------
1. 차량 추가 | 2. 차량 예약 | 3. 차량 목록 | 4. 예약 현황 | 5. 종료
-------------------------------------------------------
<선택> 2
차량 번호 : 12가1234
예약자 이름 : Kim
예약이 완료 되었습니다.
-------------------------------------------------------
1. 차량 추가 | 2. 차량 예약 | 3. 차량 목록 | 4. 예약 현황 | 5. 종료
-------------------------------------------------------
<선택> 2
차량 번호 : 56나5678
예약자 이름 : Lee
좌석 (x, y) : 1 2
예약이 완료 되었습니다.
```

- **[차량 목록 실행화면]**

```
-------------------------------------------------------
1. 차량 추가 | 2. 차량 예약 | 3. 차량 목록 | 4. 예약 현황 | 5. 종료
-------------------------------------------------------
<선택> 3
12가1234 택시
56나5678 버스
```

- **[예약 현황 실행화면]**

```
------------------------------------------------------------
1. 차량 추가 | 2. 차량 예약 | 3. 차량 목록 | 4. 예약 현황 | 5. 종료
------------------------------------------------------------
<선택> 4
차량 번호 : 12가1234
Kim
------------------------------------------------------------
1. 차량 추가 | 2. 차량 예약 | 3. 차량 목록 | 4. 예약 현황 | 5. 종료
------------------------------------------------------------
<선택> 4
차량 번호 : 56나5678
- -      - -
- -      Lee -
- -      - -
- -      - -
- -      - -
```

2.2 상세 내용 및 요구사항

- 로그인에 대한 `Account.java`, 차량에 대한 `Vehicle.java`, `Bus.java`, `Taxi.java` 그리고 전체 동작을 수행하는 main함수를 포함하는 `VehicleReservationProgram.java` 는 다음과 같이 주어진 코드를 그대로 활용하면 된다 (수정 x).

```java
public class Account {
    private String accountNumber;
    private String password;
    private String name;
    public Account(String accountNumber, String name, String
        password) {
        this.accountNumber = accountNumber;
        this.name = name;
        this.password = password;
    }
    public String getName() { return name; }
    public String getPassword() { return password; }
}
```

Code 4: Account.java

```java
public class Vehicle {
      private String number;
      private String type;
      public Vehicle(String number, String type) {
            this.number = number;
            this.type = type;
      }
      public void setNumber(String number) { this.number = number
          ;}
      public String getNumber() { return number; }
      public void setType(String type) { this.type = type; }
      public String getType() { return type; }
}
```

Code 5: Vehicle.java

```java
public class Bus extends Vehicle{
	private String[][] seat;
	public Bus(String number, String type) {
		super(number, type);
		this.seat = new String[5][4];
	}
	public void setSeat(int x, int y, String pname) {
		this.seat[x][y] = pname;
	}
	public String getSeat(int x, int y) {
		return seat[x][y];
	}
}
```

Code 6: Bus.java

```java
public class Taxi extends Vehicle{
	private String seat;
	public Taxi(String number, String type) {
		super(number, type);
		this.seat = new String();
	}
	public String getSeat() {
		return seat;
	}
	public void setSeat(String seat) {
		this.seat = seat;
	}
}
```

Code 7: Taxi.java

```java
public class VehicleReservationProgram {
    private static Scanner scanner = new Scanner(System.in);

    public static void printMenu() {
        System.out.println("차량------------
            예약관리프로그램입니다   -------------");
        System.out.println("회원가입1. | 로그인2. | 차량예약3. 관리|
            로그아웃4. | 종료5.");
        System.out.println("
            ----------------------------------------------------
            ");
    }

    public static void main(String[] args) {
        VehicleManager vihiclemanager = new VehicleManager();
        AccountManager accountmanager = new AccountManager();
        HashMap<String, Account> accountHash;

        Account current_account = null;
        while(true) {
            printMemu();
            System.out.print("선택>> ");
            int selectNo = scanner.nextInt();
```

```
20
21              if(selectNo == 1) { // 회원가입
22                  accountmanager.signup();
23              }else if(selectNo == 2) {  // 로그인
24                  current_account = accountmanager.Login();
25              }else if(selectNo == 3) {  // 차량 예약 관리
26                  if (current_account == null) {
27                      System.out.println("로그인 하십시오.");
28                  }else {
29                      vihiclemanager.run(current_account);
30                  }
31              }else if(selectNo == 4) {  // 로그아웃
32                  current_account = null;
33                  System.out.println("로그아웃 되었습니다.");
34              }else if(selectNo == 5) {  // 종료
35                  System.out.println("프로그램을 종료합니다.");
36                  break;
37              }
38          }
39      }
40  }
```

Code 8: VehicleReservationProgram.java

- `VehicleManager.java` 코드의 일부분 : 요구사항에 맞추어 (A). `addVehicle()` (B). `reserveVehicle()`, (C). `vehicleList()`, (D). `printSeat()` 함수를 구현하시오.

```
1   public class VehicleManager {
2       private Scanner scanner;
3       private HashMap<String, Vehicle> vehicleHash;
4
5       public VehicleManager() {
6           this.scanner = new Scanner(System.in);
7           this.vehicleHash = new HashMap<String, Vehicle>();
8       }
9
10      public void run(Account account) {
11          boolean flag = true;
12          while(flag) {
13              System.out.println("
                    ---------------------------------------------");
14              System.out.println("차량추가1. | 차량예약2. |
                    차량목록3. | 예약현황4. | 5. 종료");
15              System.out.println("
                    ---------------------------------------------");
16              System.out.print("선택<> ");
17              int selectNo = scanner.nextInt();
18              if(selectNo == 1) {
19                  System.out.println("--------------------");
20                  System.out.println("1. 택시 | 2. 버스");
21                  System.out.println("--------------------");
22                  System.out.print("선택<> ");
23                  int vehicleNo = scanner.nextInt();
24                  addVehicle(vehicleNo);
```

```
            }else if(selectNo == 2) {
                reserveVehicle(account);
            }else if(selectNo == 3) {
                vehicleList();
            }else if(selectNo == 4) {
                printSeat();
            }else if(selectNo == 5) {
                flag = false;
            }
        }
        System.out.print("프로그램을 종료합니다.");
    }

    private void addVehicle(int vehicleNo) { // 차량 추가
          // A. 다음 메소드를 작성하시오.
          // 입력 매개변수 vehicleNo가 1이면 택시, 2이면 버스
          // 차량 번호를 입력받아 매개변수 종류의 객체를 생성하고 HashMap에 추가
    }
    private void reserveVehicle(Account account) { // 차량 예약
          // B. 다음 메소드를 작성하시오.
          // 차량 번호를 입력 받아 차량의 종류에 따라 아래와 같이 구현한다.
          // 차량의 종류가 1. 택시인 경우, 승객 이름을 입력 받아 좌석에 승객 이름으로 예약
          // 차량의 종류가 2. 버스인 경우, 승객 이름과 좌석(x, y)을 입력 받아 해당 좌석에 승객 이름으로 예약

          // 단, 승객이름을 입력을 입력하지 않고 Enter를 누를 경우, account객체의 이름을 승객이름으로 하여 예약
    }
    private void vehicleList() { // 차량 목록
          // C. 다음 메소드를 작성하시오.
          // HashMap에 추가된 모든 차량의 번호와 종류를 출력
}
    private void printSeat() { // 예약 현황
          // D. 다음 메소드를 작성하시오.
          // 차량 번호를 입력받아 차량의 종류에 따라 아래와 같이 구현한다.
          // 차량의 종류가 1. 택시인 경우, 승객의 이름을 출력 (빈 좌석일 경우 '-'로 표시)
          // 차량의 종류가 2. 버스인 경우, 좌석을 실행 결과와 같이 출력 (빈 좌석은 '-'로 표시)
    }
}
```

Code 9: VehicleManager.java

- `AccountManager.java` 코드의 일부분 : 요구사항에 맞추어 (E). `signup()`, (F). `Login()` 함수를 구현하시오.

```
public class AccountManager {
    private Scanner scanner;
    private HashMap<String, Account> accountHash;

```

```
5       public AccountManager() {
6           this.scanner = new Scanner(System.in);
7           this.accountHash = new HashMap<String, Account>();
8       }
9
10      public void signup() { //회원가입
11          // E. 다음 메소드를 작성하시오.
12          // 회원가입에 필요한 정보인 ID, 이름, 비밀번호를 입력 받아
                HashMap에 추가
13          // 만약 아이디가 HashMap에 존재한다면 재입력
14      }
15
16      public Account Login() { //로그인
17          // F. 다음 메소드를 작성하시오.
18          // 로그인에 성공하면 HashMap에서 해당 계정을 반환
19          // 만약 아이디가 존재하지 않으면 재입력. 비밀번호가 틀렸을시
                재입력 처리
20                  return account;
21      }
22  }
```

Code 10: AccountManager.java

Table 1: 차량 예약 관리 프로그램의 `VehicleManager`와 `AccountManager`의 구현 함수에 대한 세부 요구사항 요약

문항	함수 이름	구현 사항
A	`addVehicle`	· 차량 번호를 입력 받아 vehicleNo에 따라서 해당 객체를 생성하여 `vehicleHash`에 추가 (type 1 : 택시, type 2 : 버스)
B	`reserveVehicle`	· 차량의 번호를 입력받아 차량의 type에 따라 다음과 같이 구현 1. 차량의 type이 택시인 경우, 승객의 이름을 입력받아 seat를 승객의 이름으로 설정 2. 차량의 type이 버스인 경우, 승객의 이름과 x, y좌표를 입력 받아 x, y 좌표의 seat를 승객의 이름으로 설정 · 단, 승객이름을 입력을 입력하지 않고 Enter를 누를 경우, account객체의 이름을 승객이름으로 하여 예약 · seat가 정상 설정된 경우 "예약이 완료 되었습니다." 메시지 출력 (seat가 설정 되어 있을 경우, 새로운 값으로 덮어씌움)
C	`vehicleList`	`vehicleHash`에 추가 되어있는 모든 차량 number와 type를 출력
D	`printSeat`	· 차량의 번호를 입력받아 차량의 type에 따라 다음과 같이 구현 1. 차량의 type이 택시인 경우, 승객의 이름을 출력 (빈 좌석일 경우 '-'로 표시) 2. 차량의 type이 버스인 경우. 예약된 자리에는 예약한 승객의 이름을 출력하고, 빈 좌석은 '-'로 표시 (실행결과와 출력 예제 참고)
E	`signup`	· 회원 가입에 필요한 아이디, 이름, 비밀번호 등을 입력 받아 `accountHash`에 저장 · 만약 아이디가 `HashMap`에 존재한다면 재입력
F	`Login`	· 아이디와 비밀번호를 입력하여 로그인. 로그인에 성공하면 `accountHash`에서 해당 계정을 반환 · 만약 아이디가 존재하지 않으면 재입력. 비밀번호가 틀렸을 시 재입력 처리

3 제출 내용 및 평가 방식

본 과제 결과물로 필수적으로 제출해야 내용들은 다음과 같다.

- **코드 전체**
- **보고서**: 구현 방법 및 실행 결과를 요약한 보고서

과제 평가항목 및 배점은 다음과 같다.

- **코드의 정확성 및 완결성 (80점)**
- **코드의 Readability 및 객체지향적 접근성 등 (10점)**

- **보고서의 완결성 (10점)**