

Java Programming: Assignment 1

April 26, 2021

1 미니 블랙잭 게임 만들기

블랙잭 게임은 플레이어 카드로 하는 게임으로, 플레이어와 딜러가 카드를 차례로 한 장씩 받아가며 21점에 가까운 수를 만드는 사람이 이기고, 21점을 초과하면 지는 게임이다. 표준형 플레이어 카드 1덱은 {Clubs(♣), Diamonds(♦), Hearts(♥), Spades(♠)} 4가지 문양과, {A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, K, Q} 13가지 랭크의 조합으로 52장의 카드로 구성되며, 블랙잭에는 1개 이상의 텍이 사용될 수 있다.

1.1 미니 블랙잭 게임의 규칙

구현해야 하는 미니 블랙잭 게임의 규칙은 다음과 같다.

1.1.1 미니 블랙잭 게임의 기본 규칙

- 미니 블랙잭 게임은 유저 1명과 컴퓨터 딜러와의 대결로 구성하며, N개 텍의 카드들로(N * 52) 게임을 진행한다.
- 각각의 카드의 점수는 문양과 상관없이 랭크로만 결정되며, 랭크에 대한 점수는 다음과 같다¹.

Rank	A	2	3	4	5	6	7	8	9	10	J	Q	K
Score	1 or 11	2	3	4	5	6	7	8	9	10	10	10	10

- 유저와 딜러는 시작 카드 2장을 받고, 딜러는 1장을 유저에게 공개한다. 시작 카드를 받고 나면 진행 순서는 유저 -> 딜러 순으로 단 한 번씩만 진행된다.
- Hit:** 카드를 한 장 더 받는다. 원하는 만큼 Hit를 하고 나면 Stand 해야 한다.
- Stand:** 더 이상 카드를 받지 않는다.
- Bust:** 가지고 있는 카드 점수의 총합이 21점을 초과할 시 패배한다.
- 유저와 딜러 모두 Stand 이거나, 한쪽이 Bust 했을 때, 혹은 한쪽이 BlackJack 일 때 게임이 종료된다. 양측이 모두 Stand 인 경우 카드 점수의 비교로 승부가 결정된다. Bust 했을 시에는 바로 패배한다.

¹A는 1 또는 11의 두 점수 중 하나를 취한다.

1.1.2 미니 블랙잭 게임의 세부 규칙

- BlackJack-유저: 시작 시 받은 카드 2장 중에 1장이 A, 나머지 1장이 JQK 중 하나일 경우, BlackJack이며, 게임에서 승리한다.
- Black-딜러: 시작 시 공개한 카드 1장과 자신의 차례에 공개되는 1장의 카드가 BlackJack 일 시 승리한다.
- ACE의 점수 계산-유저: 유저는 새로운 카드를 받을 때마다 패에 랭크가 ACE인 카드가 있다면, 각각 ACE 카드가 계산될 점수 1점과 11점 중 하나로 선택한다.
- ACE의 점수 계산-딜러: 딜러는 처음 받은 2개의 카드 중 랭크가 ACE인 카드가 있다면, 11점으로 계산한다. 만약 처음 받은 2개 카드의 랭크가 모두 ACE라면 각각 11점, 1점으로 계산한다. 3번째 카드부터 랭크가 ACE인 카드를 받는다면 1점으로 계산한다.
- 유저는 자신의 차례에 원하는 만큼 Hit 한 후 Stand 한다.
- 딜러는 자신의 차례에 16점 이하일 시 Hit, 17점 이상일 때 Stand 한다.

1.2 게임 시나리오: 게임은 다음과 같은 순서로 진행된다.

1. 유저가 몇 개의 텍을 사용할 것인지 정하고 카드를 섞는다.
2. 각자 카드를 2장씩 나눠 갖고, 컴퓨터 딜러는 한 장을 공개한 후 유저의 차례가 된다. 유저의 카드를 화면에 표시하고, 세부 규칙의 BlackJack을 검사한다.
3. 유저는 Hit를 원하는 만큼 입력하고, 유저는 카드를 받을 때마다 화면에서 자신의 카드를 확인할 수 있다. Stand로 자기 차례를 마무리한다.
4. 딜러는 자신의 차례에 나머지 한장을 공개한다. 이때 세부 규칙의 BlackJack을 검사한다. 딜러는 17점 이상일 때까지 3초 간격으로 Hit 하다 Stand 한다.
5. 중간 과정에서 Bust 된 쪽은 게임에서 패배한다.
6. 딜러와 유저 모두 Stand 상태일 때는 점수를 비교하여 승패를 결정한다. 점수가 같다면 무승부 처리한다.

1.3 프로그램 요구사항

구현해야 하는 세부적인 요구사항은 다음과 같다.

Table 1: 구현해야 하는 자바 소스 코드 파일들

Card.java	카드 클래스, 그대로 사용하면 된다.
Player.java	플레이어 클래스, 유저와 딜러 객체의 생성에 사용한다.
CardPool.java	카드 풀 클래스, 게임에 사용할 카드를 만들고 뽑는 용도
ScoreTable.java	점수 테이블 클래스, 카드에 대응하는 점수를 리턴하는 용도.
BlackJack.java	블랙잭 게임 메인 클래스

Table 2: 구현 함수에 대한 세부 요구사항 요약

문 항	함수 이름	구현 사항
A	addCard	· 카드를 추가한다.
B	get	· item 번째 카드를 리턴한다.
C	getHand()	· 가지고 있는 패를 모두 리턴한다.
D	CardPool	· num_decks 개의 덱을 생성하고 셔플 한 후 추가한다.
E	drawCard	· 카드들로부터 카드를 한장 뽑는다.
F	ScoreTable()	· key=랭크: value=점수 짝의 테이블을 생성한다.
G	score	· card의 점수를 리턴한다. A처럼 두 개이상의 점수를 갖는 경우를 위해서 정수배열을 리턴한다.
H	computeScore	· table을 이용해 유저의 card들의 점수 총합을 리턴. 키보드 입력을 받아 ACE의 점수를 결정하는 것 또한 수행한다.
I	computeScoreDealer	· table을 이용해 딜러의 card들의 점수 총합을 리턴. ACE의 점수 계산은 세부규칙을 참고.
J	is_bust	· 점수를 받아 21점이 초과하는지 검사
K	checkBlackjack	· 카드들을 받아 블랙잭인지 아닌지 리턴
L	sleep	· time 만큼 pause 후 재개. (단위: 밀리세컨드)
M	main	· 메인 실행 부분

1. Card.java: 카드 클래스, 그대로 사용하면 된다.

```

1 // Card.java
2 public class Card {
3     private String rank, suit;
4
5     Card(String suit, String rank) {
6         this.suit = suit;
7         this.rank = rank;
8     }
9
10    public String getSuit(){return suit;}
11    public String getRank(){return rank;}
12    public String toString(){
13        return String.format("[%s:%s]", suit, rank);
14    }
15 }
```

Code 1: Card.java

2. Player.java: 플레이어 클래스, 유저와 딜러 객체의 생성에 사용한다.

```

1 // Player.java
2 public class Player{
3     private Vector<Card> hand = new Vector<Card>();
4
5     public void addCard(Card card) {
6         // (A)
7         // 카드를 추가한다.
8     }

```

```

9     public Card get(int item) {
10        // (B)
11        // item 번째 카드를 리턴한다.
12    }
13    public Vector<Card> getHand() {
14        // (C)
15        // 가지고 있는 패를 모두 리턴한다.
16    }
17 }
```

Code 2: Player.java

3. CardPool.java: 카드 풀 클래스, 게임에 사용할 카드를 만들고 뽑는 용도

```

1 // CardPool.java
2 public class CardPool {
3     private Stack<Card> cards = new Stack<Card>();
4
5     CardPool(int num_decks){
6         // (D)
7         // num_decks 개의 덱을 생성하고 셔플 한 후 추가한다.
8     }
9     public Card drawCard(){
10        // (E)
11        // 카드들로부터 카드를 한장 뽑는다.
12    }
13 }
```

Code 3: CardPool.java

4. ScoreTable.java: 점수 테이블 클래스, 카드에 대응하는 점수를 리턴하는 용도.

```

1 // ScoreTable.java
2 public class ScoreTable {
3     private HashMap<String, int[]> table = new HashMap<>();
4
5     ScoreTable(){
6         // (F)
7         // key=랭크: value=점수 쌍의 테이블을 생성한다.
8     }
9     public int[] score(Card card) {
10        // (G)
11        // card 의 점수를 리턴한다. A의 경우 1 또는 11이며, 이 경우에
12        // 해당 점수들로 구성된 length가 2인 정수배열이 리턴된다.
13    }
}
```

Code 4: ScoreTable.java

5. BlackJack.java: 블랙잭 게임 메인 클래스

```

1 //BlackJack.java
2 public class BlackJack {
3     static final Scanner scanner = new Scanner(System.in);
```

```

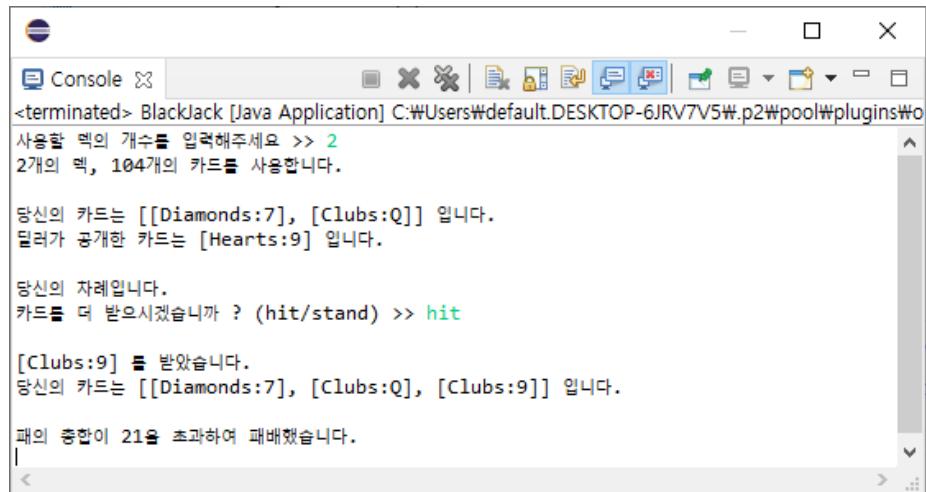
4     public static int computeScoreUser(ScoreTable table, Vector<
5         Card> cards) {
6             // (H)
7             //table을 이용해 유저의 card 들의 점수 총합을 리턴. 키보드
8                 입력을 받아 ACE의 점수를 결정하는 것 또한 수행한다.
9             }
10            public static int computeScoreDealer(ScoreTable table, Vector
11                <Card> cards) {
12                    // (I)
13                    //table을 이용해 딜러의 card 들의 점수 총합을 리턴. ACE의 점수
14                     계산은 세부규칙을 참고.
15                 }
16                 public static boolean is_bust(int score) {
17                     // (J)
18                     //점수를 받아 21점이 초과하는지 검사
19                 }
20                 public static boolean checkBlackjack(Vector<Card> cards) {
21                     // (K)
22                     //카드들을 받아 블랙잭인지 아닌지 리턴
23                 }
24                 public static void sleep(int time) {
25                     // (L)
26                     //time 만큼 pause 후 재개. (단위: 밀리세컨드)
27                 }
28                 public static void main(String[] args) {
29                     // (M)
30                     //메인 실행 부분
31                 }
32             }

```

Code 5: BlackJack.java

1.4 프로그램 실행 예

- [유저의 Bust 예시]



- [딜러의 Bust 예시]

```
<terminated> BlackJack [Java Application] C:\Users\default.DESKTOP-6JRV7V5\p2\pool\plugins\o
사용할 텍의 개수를 입력해주세요 >> 2
2개의 텍, 104개의 카드를 사용합니다.

당신의 카드는 [[Clubs:Q], [Spades:3]] 입니다.
딜러가 공개한 카드는 [Clubs:K] 입니다.

당신의 차례입니다.
카드를 더 받으시겠습니까 ? (hit/stand) >> hit

[Clubs:6] 를 받았습니다.
당신의 카드는 [[Clubs:Q], [Spades:3], [Clubs:6]] 입니다.

카드를 더 받으시겠습니까 ? (hit/stand) >> stand

당신의 차례가 끝났습니다.

딜러의 차례입니다.
딜러의 카드는 [[Clubs:K], [Clubs:2]] 입니다.

딜러는 [Diamonds:Q] 를 받았습니다.
딜러의 카드는 [[Clubs:K], [Clubs:2], [Diamonds:Q]] 입니다.

딜러 패의 총합이 21을 초과하여 승리했습니다.
```

- [유저의 BlackJack 예시]

```
<terminated> BlackJack [Java Application] C:\Users\default.DESKTOP-6JRV7V5\p2\pool\plugins\o
사용할 텍의 개수를 입력해주세요 >> 2
2개의 텍, 104개의 카드를 사용합니다.

당신의 카드는 [[Diamonds:A], [Diamonds:Q]] 입니다.
딜러가 공개한 카드는 [Spades:7] 입니다.

당신의 BlackJack으로 승리했습니다.
```

- [딜러의 BlackJack 예시]

The screenshot shows a Java application window titled "BlackJack [Java Application]". The console tab is active, displaying the following text:

```
<terminated> BlackJack [Java Application] C:\Users\default\Desktop-6JRV7V5W.p2w
사용할 맥의 개수를 입력해주세요 >> 2
2개의 맥, 104개의 카드를 사용합니다.

당신의 카드는 [[Clubs:A], [Diamonds:8]] 입니다.
딜러가 공개한 카드는 [Diamonds:A] 입니다.

당신의 차례입니다.
[Clubs:A] 의 점수를 선택해주세요. (1 / 11) >> 11
카드를 더 받으시겠습니까? (hit/stand) >> stand

당신의 차례가 끝났습니다.

딜러의 차례입니다.
딜러의 카드는 [[Diamonds:A], [Spades:K]] 입니다.

딜러의 BlackJack으로 패배했습니다.
```

- [패배 예시]

```
<terminated> BlackJack [Java Application] C:\Users\default\Desktop-6JRV7V5\p2\pool\
사용할 맥의 개수를 입력해주세요 >> 2
2개의 맥, 104개의 카드를 사용합니다.

당신의 카드는 [[Diamonds:7], [Spades:6]] 입니다.
딜러가 공개한 카드는 [Hearts:9] 입니다.

당신의 차례입니다.
카드를 더 받으시겠습니까 ? (hit/stand) >> hit

[Clubs:6] 를 받았습니다.
당신의 카드는 [[Diamonds:7], [Spades:6], [Clubs:6]] 입니다.

카드를 더 받으시겠습니까 ? (hit/stand) >> stand

당신의 차례가 끝났습니다.

딜러의 차례입니다.
딜러의 카드는 [[Hearts:9], [Clubs:A]] 입니다.

딜러의 차례가 끝났습니다.

유저: 19 vs 딜러: 20
패배했습니다.
```

- [무승부 예시]

```
<terminated> BlackJack [Java Application] C:\Users\default\Desktop-6JRV7V5\p2\pool\

사용할 맥의 개수를 입력해주세요 >> 2
2개의 맥, 104개의 카드를 사용합니다.

당신의 카드는 [[Spades:10], [Clubs:J]] 입니다.
딜러가 공개한 카드는 [Hearts:8] 입니다.

당신의 차례입니다.
카드를 더 받으시겠습니까 ? (hit/stand) >> stand

당신의 차례가 끝났습니다.

딜러의 차례입니다.
딜러의 카드는 [[Hearts:8], [Hearts:2]] 입니다.

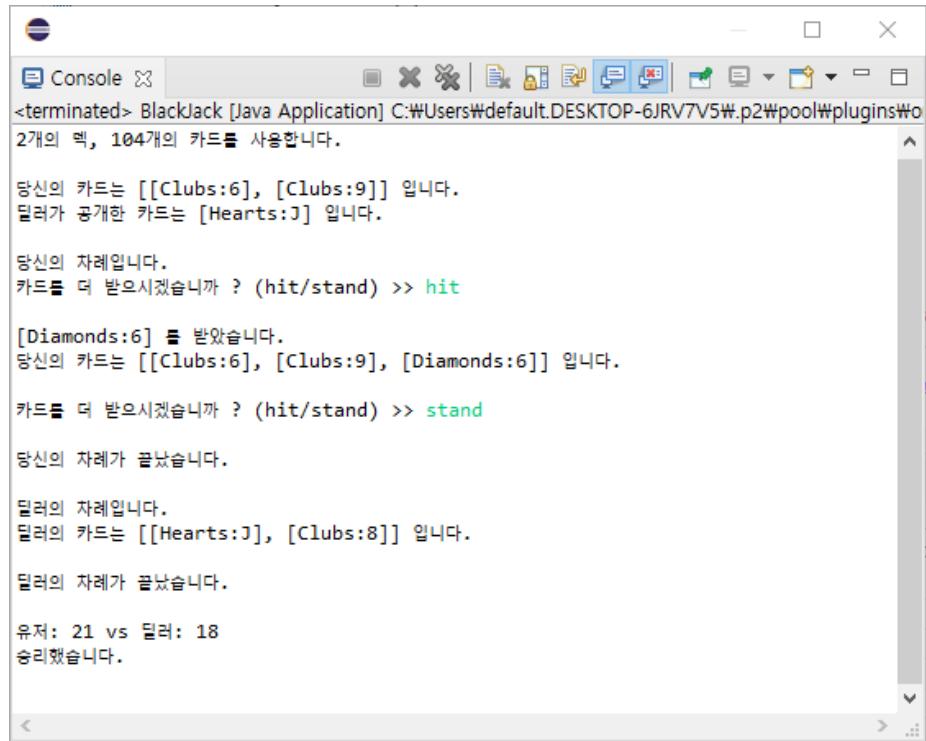
딜러는 [Clubs:A] 를 받았습니다.
딜러의 카드는 [[Hearts:8], [Hearts:2], [Clubs:A]] 입니다.

딜러는 [Hearts:9] 를 받았습니다.
딜러의 카드는 [[Hearts:8], [Hearts:2], [Clubs:A], [Hearts:9]] 입니다.

딜러의 차례가 끝났습니다.

유저: 20 vs 딜러: 20
무승부
```

- [승리 예시]



```

<terminated> BlackJack [Java Application] C:\Users\default\Desktop-6JRV7V5\P2\pool\plugins\o
2개의 렉, 104개의 카드를 사용합니다.

당신의 카드는 [[Clubs:6], [Clubs:9]] 입니다.
딜러가 공개한 카드는 [Hearts:J] 입니다.

당신의 차례입니다.
카드를 더 받으시겠습니까? (hit/stand) >> hit

[Diamonds:6] 을 받았습니다.
당신의 카드는 [[Clubs:6], [Clubs:9], [Diamonds:6]] 입니다.

카드를 더 받으시겠습니까? (hit/stand) >> stand

당신의 차례가 끝났습니다.

딜러의 차례입니다.
딜러의 카드는 [[Hearts:J], [Clubs:8]] 입니다.

딜러의 차례가 끝났습니다.

유저: 21 vs 딜러: 18
승리했습니다.

```

2 도서관 관리 프로그램

구현해야 할 도서관 관리 프로그램은 관리자 모드와 회원 모드로 나뉘며 관리자는 새로운 도서의 등록 및 재고 수량 변경 등을 담당하며 회원은 도서관 관리 프로그램을 통해 도서를 자유롭게 대여, 반납 할 수 있다.

2.1 프로그램 요구사항

Table 3: 구현해야 하는 자바 소스 코드 파일들

<code>Person.java</code>	회원과 관리자를 위한 상위 클래스
<code>Member.java</code>	회원을 위한 클래스
<code>Manager.java</code>	관리자를 위한 클래스
<code>Book.java</code>	도서정보를 위한 클래스
<code>TotalLibraryManage.java</code>	도서 통합 관리 클래스
<code>MemberManage.java</code>	회원 관리 클래스
<code>BookManage.java</code>	도서 정보 관리 클래스
<code>LibraryManagementProgram.java</code>	도서관 관리 메인 클래스

다음 각 코드의 주석란의 (A) ~ (Q)까지의 내용을 채워서 코드를 완성하면 된다.

여기서, 도서관 관리 프로그램을 이용하는 사용자를 위한 클래스는 `Person`, `Member`, `Manager`이다. `Member`는 회원을 위한 클래스, `Manager`는 관리자를 위한 클래스 그리고 `Person` 클래스는 이들의 상위 클래스이다.

1. Person.java: Person 클래스, 그대로 사용하면 된다.

```
1 // Person.java
2 public class Person {
3     private String ID;
4     private String name;
5     private String password;
6
7     public Person(String ID, String name, String password) {
8         this.ID = ID;
9         this.name = name;
10        this.password = password;
11    }
12
13    public String getID() { return ID; }
14    public String getName() { return name; }
15    public String getPassword() { return password; }
16
17    public void setID(String id) { this.ID = ID; }
18    public void setName(String name) { this.name = name; }
19    public void setPassword(String password) { this.password =
password; }
20 }
```

Code 6: Person.java

2. Manager.java: 관리자를 위한 클래스

```
1 // Manager.java
2 public class Manager extends Person {
3     // (A)
4     // 상위 클래스 생성자 호출
5 }
```

Code 7: Manager.java

3. Member.java: 회원을 위한 클래스

```
1 // Member.java
2 public class Member extends Person {
3     HashMap<String, Book> bookHash; //대여 중인 도서를 저장하는 hash
4
5     public Member(String ID, String name, String password) {
6         // (B)
7         // 생성자(구현 필요)
8     }
9
10    HashMap<String, Book> getbookHash(){ return bookHash; }
11
12    void PrintRentalList() {
13        // (C)
14        // 대출 중인 도서를 출력
15    }
16 }
```

Code 8: Member.java

4. Book.java: 도서정보를 표현하기 위한 클래스

```
1 // Book.java
2 public class Book {
3     String book_number;
4     String name;
5     String genre;
6     int stock;
7
8     public Book(String book_number, String name, String genre,
9                 int stock) {
10        this.book_number = book_number;
11        this.name = name;
12        this.genre = genre;
13        this.stock = stock;
14    }
15
16    String getBookNumber(){ return book_number; }
17    String getName(){ return name; }
18    String getGenre(){ return genre; }
19    int getStock(){return stock; }
20
21    // 원하는 수량 만큼 재고 증가
22    void updateStock(int new_stock) { this.stock += new_stock; }
23
24    void AddStock() { this.stock += 1; }
25    void SubstractStock() {this.stock -= 1; }
26 }
```

Code 9: Book.java

5. TotalLibraryManage.java: 통합 관리 클래스는 사용자, 관리자로부터 정보를 받아 사용자 정보와 도서 정보를 통합 관리하는 클래스. 회원을 관리하는 MemberManage 클래스와 BookManage 클래스의 객체를 멤버 변수로 가지며 관리자, 회원에 따른 별도의 기능(ManageRun, MemberRun)을 제공한다. MemberManage 클래스와 BookManage 클래스는 각각 회원과 도서를 저장하는 해시를 가지며 멤버함수를 통해 이용자로부터 정보를 입력받아 정보를 업데이트 한다.

```
1 // TotalLibraryManage.java
2 public class TotalLibraryManage {
3     MemberManage memberManage;
4     BookManage bookManage;
5
6     private Scanner scanner;
7
8     public TotalLibraryManage() {
9         memberManage = new MemberManage();
10        bookManage = new BookManage();
11        scanner = new Scanner(System.in);
12    }
13
14    public MemberManage getmemberManage() { return memberManage;
15    }
```

```

16   public BookManage getbookManage() { return bookManage; }
17
18   public void ManagerRun(Person person) { // 관리자
19     // (D)
20     //루프를 돌면서 관리자 기능 수행
21     // 1. 전체 도서 목록 출력
22     // 2. 도서 등록
23     // 3. 도서 재고 추가
24     // 4. 회원 목록 보기
25     // 5. 돌아가기(루프 탈출)
26   }
27
28   public void MemberRun(Person person) { // 회원
29     // (E)
30     //루프를 돌면서 회원 기능 수행
31     // 1. 전체 도서 목록 출력
32     // 2. 도서 대여
33     // 3. 도서 반납
34     // 4. 대여 도서 목록
35     // 5. 돌아가기(루프 탈출)
36   }
37 }
```

Code 10: TotalLibraryManage.java

6. BookManage.java: 도서 관리 클래스

```

1  // BookManage.java
2  public class BookManage {
3
4    private Scanner scanner;
5    private HashMap<String, Book> bookHash;
6
7    public BookManage() {
8      this.scanner = new Scanner(System.in);
9      this.bookHash = new HashMap<String, Book>();
10    }
11
12   void printBookList() {
13     // (F)
14     // bookHash에 저장된 모든 도서 정보 출력
15   }
16
17   boolean AddBook() {
18     // (G)
19     // scanner 객체를 통해 도서 정보를 입력 받아 bookHash에 저장
20     // 현재 보유중인 도서가 아닌 도서만 추가 가능하도록 제약 필요
21     // 원하는 재고 수량이 0보다 작으면 추가 불가능 제약 필요
22   }
23
24   boolean UpdateBookStock() {
25     // (H)
26     // scanner를 통해 도서 번호로 bookHash에 접근하여 재고
27     // 업데이트
28     // 현재 보유중인 도서만 가능하도록 제약 필요
29 }
```

```

28         // 추가하고자 하는 재고 수량이 0보다 작으면 갱신 불가능 제약
29         필요
30     }
31
32     boolean ReturnBook(Member member) {
33         // (I)
34         // 도서 번호를 입력 받아 현재 회원이 대여중인 도서 반납
35         // 현재 회원이 대여중인 도서만 반납 가능하도록 제약 필요
36         // 재고는 다시 올려줌
37     }
38
39     boolean RentalBook(Member member) {
40         // (J)
41         // 도서 번호를 입력 받아 현재 도서 대여
42         // 도서가 존재하지 않거나 재고가 0개 이면 대여 불가능
43         // 대여하는 책의 수량은 1개만 가능
44     }

```

Code 11: BookManage.java

7. MemberManage.java: 회원 관리 클래스

```

1  // MemberManage.java
2  public class MemberManage {
3      private Scanner scanner;
4      private HashMap<String, Member> memberHash;
5
6      public MemberManage() {
7          this.scanner = new Scanner(System.in);
8          this.memberHash = new HashMap<String, Member>();
9      }
10
11     public void signup() {
12         // (K)
13         // 회원 ID, 이름, 비밀번호를 입력받아 memberHash에 추가
14         // 루프를 돌면서 ID가 이미 존재하면 “이미 존재하는 ID입니다”
15         // 출력 후
16         // 재입력. 재입력된 ID가 중복되지 않으면 루프 탈출
17     }
18     public Member Login() {
19         Member member = null;
20         // (L)
21         // ID가 존재하지 않거나 비밀번호가 틀릴 경우 메시지 출력 후
22         // null 반환
23         // 로그인 성공 시 해당 멤버 객체 반환
24     }
25
26     public void PrintMemberList() {
27         // (M)
28         // memberHash로부터 회원 리스트 출력
29     }

```

Code 12: MemberManage.java

8. **LibraryManagementProgram.java**: 도서 관리 메인 클래스. 메인 메뉴를 시작으로 관리자 모드와 회원 모드로 나뉘며 각각 로그인 한 후 해당하는 작업을 수행할 수 있다. 관리자는 별도의 회원 가입을 필요로 하지 않으며 미리 정의한(pre-defined) 관리자 정보와 입력한 ID, 비밀번호가 일치하면 로그인 할 수 있다.

```

1 // LibraryManagementProgram.java
2 public class LibraryManagementProgram {
3     private static Scanner scanner = new Scanner(System.in);
4     private static final String managerID = "0000";
5     private static final String managerName = "홍길동";
6     private static final String managerPWD = "1234";
7
8     private static TotalLibraryManage libManager = new
9         TotalLibraryManage();
10    static Person currentPerson = null;
11
12    public static void main(String[] args) {
13        // (N)
14        //
15        while (true) {
16            printMenu();
17            System.out.print("입력 >> ");
18            int selectNum = scanner.nextInt();
19
20            if (selectNum == 1) { // 관리자 모드
21                // 로그인 후 이용 가능. 정보가 일치하지 않으면 초기
22                // 메뉴로...
23                while {
24                    // 1. 도서 관리
25                    // 2. Logout (currentPerson = null; 후 break;)
26                    }
27                } else if (selectNum == 2) { // 회원 메뉴
28                    while {
29                        //
30                        // 1. 회원 가입
31                        // 2. 로그인
32                        // 3. 도서 대출(로그인 후 이용 가능 제약)
33                        // 4. Logout (currentPerson = null; 후 break;)
34                    }
35                } else if (selectNum == 3) { // 종료
36                    System.out.println("프로그램을 종료합니다.");
37                    break;
38                }
39            }
40
41            public static void printMainMenu() {
42                // (0)
43                // 메인 메뉴 작성
44            }
45
46            public static void printManagerMenu() {
47                // (P)

```

```

47         // 관리자 메뉴 작성
48     }
49
50     public static void printMemberMenu() {
51         // (0)
52         // 회원 메뉴 작성
53     }
54 }
```

Code 13: LibraryManagementProgram.java

2.2 프로그램 실행 예

- [메인 화면]

-----전북대학교 컴퓨터 공학과 도서관 관리 프로그램입니다.-----
 1. 관리자 메뉴 | 2. 회원 메뉴 | 3. 종료

 입력 >> |

- [관리자 로그인(실패)]

-----전북대학교 컴퓨터 공학과 도서관 관리 프로그램입니다.-----
 1. 관리자 메뉴 | 2. 회원 메뉴 | 3. 종료

 입력 >> 1
 -----관리자 모드입니다. 로그인 하십시오.-----
 ID : 0000
 비밀번호 : 1111
 관리자 아이디 혹은 비밀번호가 일치하지 않아 초기 화면으로 돌아갑니다.
 -----전북대학교 컴퓨터 공학과 도서관 관리 프로그램입니다.-----
 1. 관리자 메뉴 | 2. 회원 메뉴 | 3. 종료

 입력 >> |

- [관리자 로그인(성공) ⇒ 도서 관리]

-----전북대학교 컴퓨터 공학과 도서관 관리 프로그램입니다.-----
1. 관리자 메뉴 | 2. 회원 메뉴 | 3. 종료

입력 >> 1
-----관리자 모드입니다. 로그인 하십시오.-----
ID : 0000
비밀번호 : 1234
로그인 되었습니다.
----- 관리자 모드 -----
1.도서 관리 | 2. 로그아웃

입력 >> 1

1.전체 도서 목록 출력 | 2.도서 등록 | 3. 도서 재고 추가 | 4. 회원 목록 보기 | 5. 종료

- [도서등록]

1.전체 도서 목록 출력 | 2.도서 등록 | 3. 도서 재고 추가 | 4. 회원 목록 보기 | 5. 종료

<선택> 2
도서 번호 : 3361
도서 이름 : 나루토
도서 장르 : 만화
도서 수량 : 3

- [전체 도서 목록 보기]

<선택> 1
도서번호 : 4172, 도서 명 : 개미, 도서 장르 : 소설, 재고 : 2
도서번호 : 3361, 도서 명 : 나루토, 도서 장르 : 만화, 재고 : 3

1.전체 도서 목록 출력 | 2.도서 등록 | 3. 도서 재고 추가 | 4. 회원 목록 보기 | 5. 종료

- [전체 재고 추가]

<선택> 1
도서번호 : 4172, 도서 명 : 개미, 도서 장르 : 소설, 재고 : 2
도서번호 : 3361, 도서 명 : 나루토, 도서 장르 : 만화, 재고 : 3

1.전체 도서 목록 출력 | 2.도서 등록 | 3. 도서 재고 추가 | 4. 회원 목록 보기 | 5. 종료

- [돌아가기 ⇒ 로그아웃]

1. 전체 도서 목록 출력 | 2. 도서 등록 | 3. 도서 재고 추가 | 4. 회원 목록 보기 | 5. 돌아가기

<선택> 5

프로그램을 종료합니다.

----- 관리자 모드 -----

1. 도서 관리 | 2. 로그아웃

입력 >> 2

----- 전북대학교 컴퓨터 공학과 도서관 관리 프로그램입니다. -----

1. 관리자 메뉴 | 2. 회원 메뉴 | 3. 종료

입력 >> |

- [회원 메뉴(입력 과정 생략) ⇒ 회원가입]

----- 회원 모드 -----

1. 회원가입 | 2. 로그인 | 3. 도서 대출 | 4. 로그아웃

입력 >> 1

ID : 1111

이름 : 흥길동

비밀번호 : 1111

회원가입 완료

----- 회원 모드 -----

1. 회원가입 | 2. 로그인 | 3. 도서 대출 | 4. 로그아웃

입력 >> 1

ID : 2222

이름 : 마징가

비밀번호 : 2222

회원가입 완료

- [로그인 없이 도서 대출 메뉴 클릭]

----- 회원 모드 -----

1. 회원가입 | 2. 로그인 | 3. 도서 대출 | 4. 로그아웃

입력 >> 3

로그인 해야 이용할 수 있습니다.

- [도서 대출 메뉴 ⇒ 전체 도서 목록 보기]

1. 전체 도서 목록 출력 | 2. 도서 대여 | 3. 도서 반납 | 4. 대여 도서 목록 | 5. 돌아가기

<선택> 1

도서번호 : 4172, 도서명 : 개미, 도서장르 : 소설, 재고 : 2

도서번호 : 3361, 도서명 : 나루토, 도서장르 : 만화, 재고 : 3

• [2편의 도서 대여(재고 변화)]

1. 전체 도서 목록 출력 | 2. 도서 대여 | 3. 도서 반납 | 4. 대여 도서 목록 | 5. 돌아가기

<선택> 2

도서 번호 : 4172

[도서번호 : 4172] + [도서명 : 개미]이 정상적으로 대여되었습니다.

1. 전체 도서 목록 출력 | 2. 도서 대여 | 3. 도서 반납 | 4. 대여 도서 목록 | 5. 돌아가기

<선택> 1

도서번호 : 4172, 도서명 : 개미, 도서장르 : 소설, 재고 : 1

도서번호 : 3361, 도서명 : 나루토, 도서장르 : 만화, 재고 : 3

1. 전체 도서 목록 출력 | 2. 도서 대여 | 3. 도서 반납 | 4. 대여 도서 목록 | 5. 돌아가기

<선택> 2

도서 번호 : 3361

[도서번호 : 3361] + [도서명 : 나루토]이 정상적으로 대여되었습니다.

1. 전체 도서 목록 출력 | 2. 도서 대여 | 3. 도서 반납 | 4. 대여 도서 목록 | 5. 돌아가기

<선택> 1

도서번호 : 4172, 도서명 : 개미, 도서장르 : 소설, 재고 : 1

도서번호 : 3361, 도서명 : 나루토, 도서장르 : 만화, 재고 : 2

1. 전체 도서 목록 출력 | 2. 도서 대여 | 3. 도서 반납 | 4. 대여 도서 목록 | 5. 돌아가기

• [대여 목록 보기]

1. 전체 도서 목록 출력 | 2. 도서 대여 | 3. 도서 반납 | 4. 대여 도서 목록 | 5. 돌아가기

<선택> 4

도서번호 : 4172, 도서명 : 개미, 도서장르 : 소설

도서번호 : 3361, 도서명 : 나루토, 도서장르 : 만화

• [도서 반납 ⇒ 전체 도서 목록 출력(재고 증가)]

1. 전체 도서 목록 출력 | 2. 도서 대여 | 3. 도서 반납 | 4. 대여 도서 목록 | 5. 돌아가기

<선택> 3

도서 번호 : 4172

[도서번호 : 4172] + [도서명 : 개미]이 정상적으로 반납되었습니다.

1. 전체 도서 목록 출력 | 2. 도서 대여 | 3. 도서 반납 | 4. 대여 도서 목록 | 5. 돌아가기

<선택> 1

도서번호 : 4172, 도서 명 : 개미, 도서 장르 : 소설, 재고 : 2

도서번호 : 3361, 도서 명 : 나루토, 도서 장르 : 만화, 재고 : 2

1. 전체 도서 목록 출력 | 2. 도서 대여 | 3. 도서 반납 | 4. 대여 도서 목록 | 5. 돌아가기

- [돌아가기 ⇒ 로그아웃]

1. 전체 도서 목록 출력 | 2. 도서 대여 | 3. 도서 반납 | 4. 대여 도서 목록 | 5. 돌아가기

<선택> 5

이전 화면으로 돌아갑니다.

----- 회원 모드 -----

1. 회원가입 | 2. 로그인 | 3. 도서 대출 | 4. 로그아웃

입력 >> 4

----- 전북대학교 컴퓨터 공학과 도서관 관리 프로그램입니다. -----

1. 관리자 메뉴 | 2. 회원 메뉴 | 3. 종료

입력 >>

- [관리자 재로그인 ⇒ 도서 관리]

----- 전북대학교 컴퓨터 공학과 도서관 관리 프로그램입니다. -----
1. 관리자 메뉴 | 2. 회원 메뉴 | 3. 종료

입력 >> 1

----- 관리자 모드입니다. 로그인 하십시오. -----

ID : 0000

비밀번호 : 1234

로그인 되었습니다.

----- 관리자 모드 -----

1. 도서 관리 | 2. 로그아웃

입력 >> 1

1. 전체 도서 목록 출력 | 2. 도서 등록 | 3. 도서 재고 추가 | 4. 회원 목록 보기 | 5. 돌아가기

<선택> |

- [회원 목록 보기]

```
-----  
1.전체 도서 목록 출력 | 2.도서 등록 | 3. 도서 재고 추가 | 4. 회원 목록 보기 | 5. 돌아가기
```

<선택> 4

ID : 1111, 이름 : 흥길동
ID : 2222, 이름 : 마징가

```
-----  
1.전체 도서 목록 출력 | 2.도서 등록 | 3. 도서 재고 추가 | 4. 회원 목록 보기 | 5. 돌아가기
```

<선택>

- [전체 도서 목록 출력 ⇒ 도서 재고 추가 ⇒ 전체 도서 목록 출력(업데이트 확인)]

```
-----  
1.전체 도서 목록 출력 | 2.도서 등록 | 3. 도서 재고 추가 | 4. 회원 목록 보기 | 5. 돌아가기
```

<선택> 1

도서번호 : 4172, 도서 명 : 개미, 도서 장르 : 소설, 재고 : 2
도서번호 : 3361, 도서 명 : 나루토, 도서 장르 : 만화, 재고 : 2

```
-----  
1.전체 도서 목록 출력 | 2.도서 등록 | 3. 도서 재고 추가 | 4. 회원 목록 보기 | 5. 돌아가기
```

<선택> 3

추가하고자 하는 도서 번호 : 4172

추가하고자 하는 수량 : 3

[도서번호 : 4172] + [도서명 : 개미]의 수량이 2에서 5로 증가되었습니다.

```
-----  
1.전체 도서 목록 출력 | 2.도서 등록 | 3. 도서 재고 추가 | 4. 회원 목록 보기 | 5. 돌아가기
```

<선택> 1

도서번호 : 4172, 도서 명 : 개미, 도서 장르 : 소설, 재고 : 5
도서번호 : 3361, 도서 명 : 나루토, 도서 장르 : 만화, 재고 : 2

```
-----  
1.전체 도서 목록 출력 | 2.도서 등록 | 3. 도서 재고 추가 | 4. 회원 목록 보기 | 5. 돌아가기
```

<선택>

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)