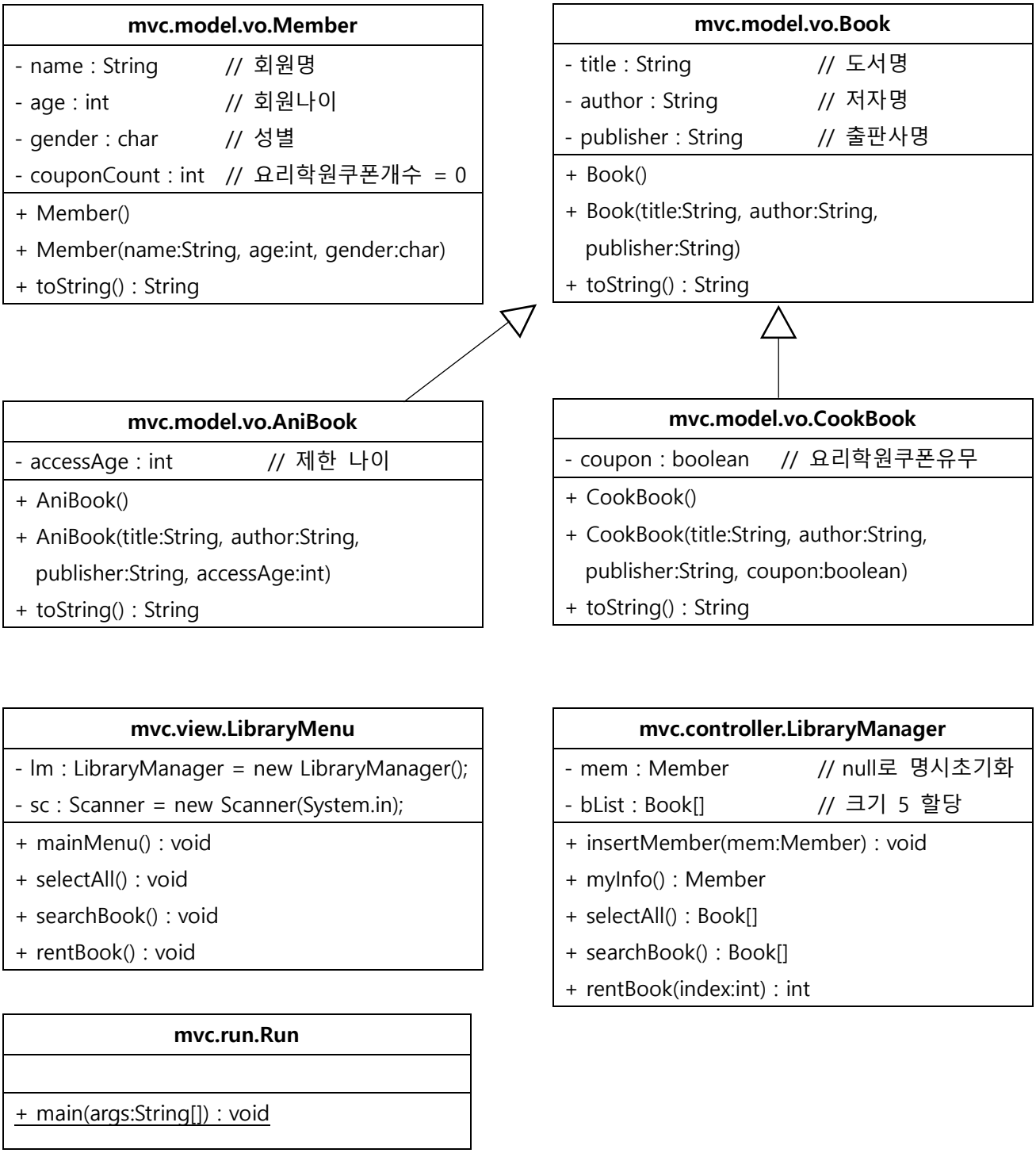
[다형성 mvc 실습문제 1] 다음과 같은 조건을 만족하는 프로그램을 작성 하시오

도서대여 관련 프로그램으로 다형성을 적용한 프로그램이다. 해당 구현 클래스 다이어그램과 클래스 구조를 참고하여 프로젝트를 완성하시오.

*** 프로젝트 명 : 09_Polymorphism_Homework_본인이름**

1. 구현 클래스 다이어그램 (Class Diagram)



**** 문제 추가 설명 ****

회원이 만화책 또는 요리책을 빌리려고 한다. 먼저 프로그램이 실행되면 회원의 인적 사항을 입력하고 회원등록을 해준다. 그 다음에 메뉴가 출력되면서 마이페이지, 도서전체조회, 도서검색, 도서대여 기능을 할 수 있다. 도서를 대여할 때 해당 도서가 만화책일 경우 나이제한이 있기 때문에 회원 나이와 만화책의 제한 나이를 비교해야 되고, 대여할 도서가 요리책일 경우 해당 도서에 요리학원 쿠폰이 있으면 쿠폰이 해당 멤버 객체에게 발급된다.

2. 구현 클래스 설명

Package명	Class명	Method	설명
mvc.run	Run	+main(args:String[]) : void	LibraryMenu의 mainMenu 호출
mvc.view	LibraryMenu	+mainMenu() : void	프로그램 실행 시 가장 먼저 호출되는 메소드로 회원 이름, 나이, 성별을 입력받아 Member객체 생성 후 해당 객체를 LibraryManager의 insertMember 메소드로 전달 무한 반복 메뉴를 출력하여 각 메뉴 버튼 선택 시 해당 메소드 호출 (클래스 구조 참고)
		+selectAll() : void	LibraryManager의 selectAll 메소드 호출 → 결과 값을 Book[] 자료형으로 받아준 뒤 for문을 통해 도서 전체 목록 출력
		+searchBook() : void	검색할 제목 "키워드"를 입력 받고 그 키워드를 LibraryManager의 searchBook 메소드로 전달 → 결과 값을 Book[] 자료형으로 받아준 뒤 향상된 for문을 이용하여 출력
		+rentBook() : void	대여할 도서 인덱스를 입력 받아 LibraryManager의 rentBook 메소드로 전달 → 결과 값을 result로 받아 result가 0일 경우, 1일 경우, 2일 경우 각각에 해당하는 출력문 출력
mvc.controller	LibraryManager	+insertMember (mem:Member) : void	전달받은 mem을 LibraryManager의 mem에 대입

		+myInfo() : Member	회원 레퍼런스(mem) 리턴
		+selectAll() : Book[]	도서 전체 목록 (bList) 주소 값 리턴
		+searchBook(keyword :String) : Book[]	전달받은 키워드가 포함된 도서가 여러 개가 존재할 수 있으니 검색된 도서를 담아줄 Book 객체 배열을 새로이 생성하고 for문을 통해 bList 안의 도서들과 전달받은 키워드를 비교하여 포함하고 있는 경우 새로운 배열에 차곡차곡 담기 → 그 배열 주소 값 리턴
		+rentBook(index:int) : int	result를 0으로 초기화 한 후 전달받은 인덱스의 도서가 만화책인 경우 회원의 나이와 해당 만화책의 제한 나이를 비교하여 회원 나이가 더 적은 경우 result를 1로 초기화 아니면 전달받은 인덱스의 도서가 요리책이면서 해당 요리책의 쿠폰 유무가 true일 경우 회원의 couponCount를 1증가 시킨 후 result 2로 초기화 → result 리턴

3. class 구조

```
public class LibraryMenu{
        private LibraryManager lm = new LibraryManager();
        private Scanner sc = new Scanner(System.in);
        public void mainMenu() {
                // 이름, 나이, 성별을 키보드로 입력 받은 후 Member 객체 생성
                // LibraryManager의 insertMember() 메소드에 전달

                ==== 메뉴 ====          // 무한 반복 실행
                1. 마이페이지           // LibraryManager의 myInfo() 호출한 후 그 결과값 출력
                2. 도서 전체 조회       // LibraryMenu의 selectAll() 호출
                3. 도서 검색            // LibraryMenu의 searchBook() 호출
                4. 도서 대여하기        // LibraryMenu의 rentBook() 호출
                0. 프로그램 종료하기
        }
```

```
	public void selectAll() {
		// LibraryManager의 selectAll() 메소드 호출하여 결과 값 Book[] 자료형에 담기
		➔ Book[] bList = lm.selectAll();

		// for문 이용하여 bList의 모든 도서 목록 출력
		// 단, i를 이용하여 인덱스도 같이 출력 → 대여할 때 도서번호를 알기 위해
		ex ) 0번도서 : 백종원의 집밥 / 백종원 / tvN / true
	}

	public void searchBook() {
		// "검색할 제목 키워드 : "		>> 입력 받음 (keyword : String)
		// LibraryManager의 searchBook() 메소드에 전달
		// 그 결과 값을 Book[] 자료형으로 받기
		➔ Book[] searchList = lm.searchBook(keyword);

		// for each문(향상된 for문)을 이용하여 검색 결과의 도서 목록 출력
		// NullPointerException 발생 시 오류 해결하시오 ㅎㅎ
	}

	public void rentBook() {
		// 도서 대여를 위해 도서번호를 알아야 된다.
		→ selectAll() 메소드 호출을 통해 도서 리스트 한번 출력 해주고

		// "대여할 도서 번호 선택 : "		>> 입력 받음 (num : int)
		// LibraryManager의 rentBook(num) 메소드에 전달
		// 그 결과 값을 result로 받고 그 result가
		// 0일 경우 → "성공적으로 대여되었습니다." 출력
		// 1일 경우 → "나이 제한으로 대여 불가능입니다." 출력
		// 2일 경우 → "성공적으로 대여되었습니다. 요리학원 쿠폰이 발급되었습니다.
				마이페이지를 통해 확인하세요" 출력
	}
}
```

```
public class LibraryManager{
	private Member mem = null;
	private Book[] bList = new Book[5];

	{ // 초기화 블록을 이용하여 샘플 데이터 초기화
		bList[0] = new CookBook("백종원의 집밥", "백종원", "tvN", true);
		bList[1] = new AniBook("한번 더 해요", "미티", "원모어", 19);
		bList[2] = new AniBook("루피의 원피스", "루피", "japan", 12);
		bList[3] = new CookBook("이혜정의 얼마나 맛있게요", "이혜정", "문학", false);
		bList[4] = new CookBook("최현석 날 따라해봐", "최현석", "소금책", true);
	}

	public void insertMember(Member mem) {
		// 전달받은 mem 주소 값을 해당 회원 레퍼런스(mem)에 대입
	}

	public Member myInfo() {
		// 회원 레퍼런스(mem) 주소 값 리턴
	}

	public Book[] selectAll() {
		// 도서 목록 레퍼런스(bList) 주소 값 리턴
	}

	public Book[] searchBook(String keyword) {
		// 검색 결과를 담아줄 새로운 Book 객체 배열 생성
		// 검색 결과 도서 목록이 최대 5개일 수 있으니 임의로 크기 5 할당

		// for문을 이용하여 bList 도서 목록들의 도서명과 전달받은 keyword 비교
		// 전달받은 keyword를 포함하고 있으면 → HINT : String 클래스의 contains() 참고
		// 검색결과의 도서목록에 담기		→ HINT : count 이용

		// 해당 검색결과의 도서목록 주소 값 리턴
	}
}
```

```
	public int rentBook(int index) {
		int result = 0;

		// 전달 받은 index의 bList 객체가 실제 AniBook 객체를 참조하고 있고
		// 해당 만화책의 제한 나이와 회원의 나이를 비교하여 회원 나이가 적을 경우
		// result를 1로 초기화 → 나이 제한으로 대여 불가하다는 의미

		// 전달 받은 index의 bList 객체가 실제 CookBook 객체를 참조하고 있고
		// 해당 요리책의 쿠폰유무가 "true"일 경우
		// 회원의 couponCount 1 증가 처리 후
		// result를 2로 초기화 → 성공적으로 대여 완료, 요리학원 쿠폰이 발급됐다는 의미

		// result 값 리턴
	}
}
```