

표 13-5 클래스 설계 설명

타입	클래스 이름	설명
class	MainApp	메인 메소드를 가지는 클래스 프로그램 실행의 기점
interface	Menu	화면 출력과 이동을 통해 프로그램이 동작하게끔 유도
abstract class	AbstractMenu	Menu 인터페이스를 구현하는 추상 클래스 메인 메뉴(MainMenu)와 관리자 메뉴(AdminMenu)의 부모 클래스
class	MainMenu	메인 메뉴의 출력과 입력에 따른 처리를 담당
class	AdminMenu	관리자 메뉴의 출력과 입력에 따른 처리를 담당
class	Movie	영화 정보를 관리하는 클래스 영화의 파일 입출력을 담당
class	Reservation	예매 정보를 관리하는 클래스 예매의 파일 입출력을 담당
class	Seats	예매 좌석을 관리하는 클래스

CH13

실전 프로젝트: 영화 예매 관리 프로그램

01 프로젝트 시작과 분석

02 프로젝트 설계

03 프로젝트 구현

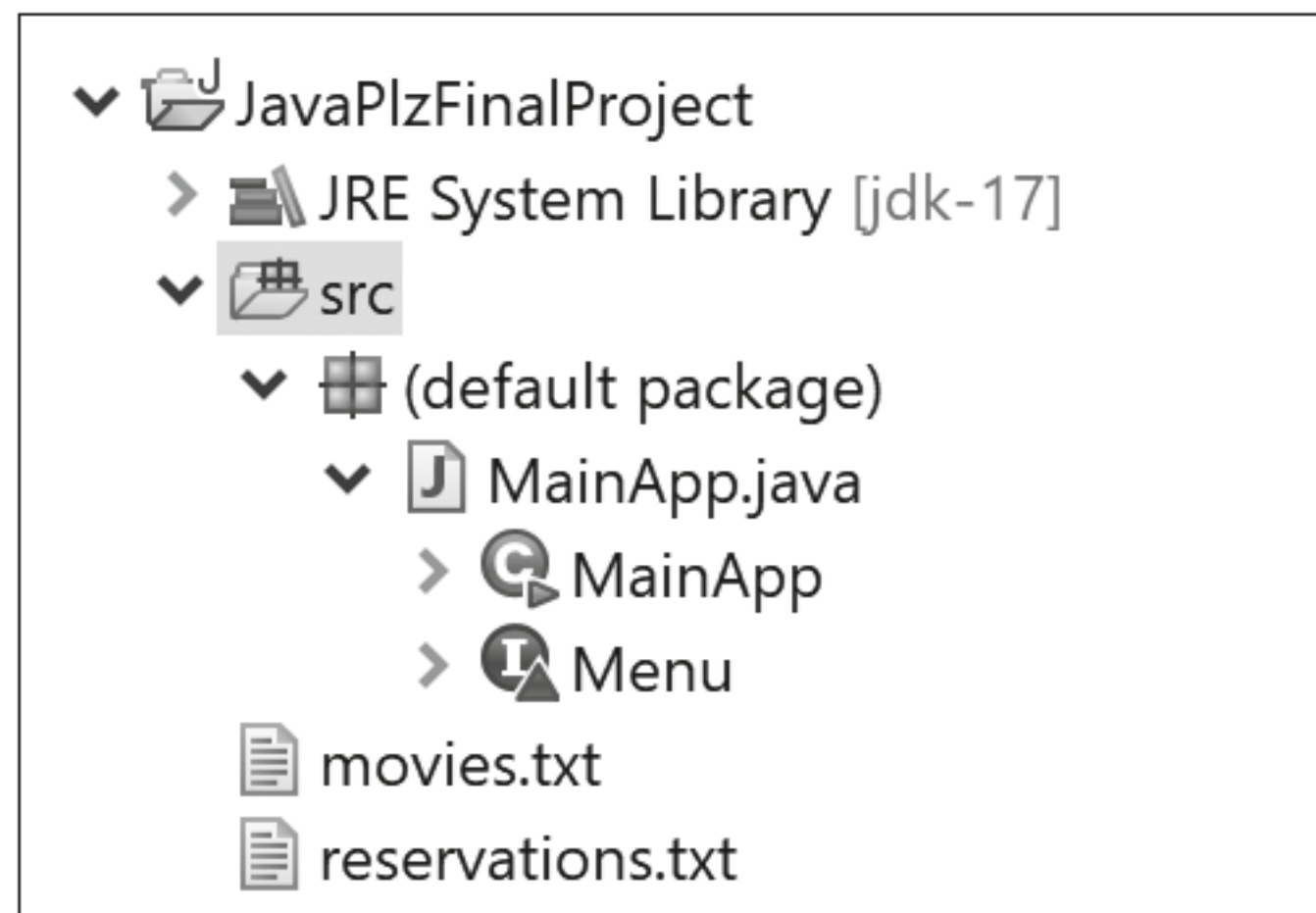
프로젝트 구현

- ① 프로젝트 기본 파일
- ② 프로그램 실행과 종료
- ③ 메뉴간 이동
- ④ 영화 목록 조회
- ⑤ 영화 등록
- ⑥ 영화 삭제
- ⑦ 예매 확인
- ⑧ 프로젝트 기본 파일
- ⑨ 영화 예매
- ⑩ 전체 완성 코드

I. 프로젝트 기본 파일

새 프로젝트 'JavaPlzFinalProejct'를 만들고

- MainApp 클래스와
- 데이터 파일(moves.txt, reservations.txt)을 생성



TIP 프로그래밍 코드는 src 디렉터리 내부에, 텍스트 파일은 프로젝트에 직접 추가

그림 13-5 기본 파일 구현 후 프로젝트 파일 구조

```
01 public class MainApp {
02     public static void main(String[] args) {
03         System.out.println("프로그램을 시작합니다!"); //
04         Menu menu = MainMenu.getInstance(); //
05         //-----메뉴가 없을 때까지 반복
06         while (menu != null) { /
07             menu.print(); /
08             menu = menu.next(); /
09         }
10         System.out.println("프로그램이 종료됩니다."); //
11     }
12 }
13
14 interface Menu {
15     void print(); // 메뉴 출력
16     Menu next(); // 다음 메뉴로 이동
17 }
```

MainApp.java 생성

MainApp 클래스를 작성. 메뉴 전환을 반복하여 전체 프로그램을 동작시킴.

- 03행: 프로그램 실행 문구를 출력
- 04행: 메인 메뉴 객체를 생성
- 06~09행: 메뉴 출력과 전환을 반복하여 프로그램을 수행
- 14~17행: 메뉴의 역할을 인터페이스로 정의

01 1627175707, 어벤져스, 판타지 ----- 영화 장르

영화 대포깅 영화 제목

02 1627175946, 컨저링, 호러

03 1627175983, 러부 액츄얼리, 로맨스

04 ----- 개행 문자 '\n'에 의해 빈 줄 추가

movies.txt 생성

프로젝트에 movies.txt 파일을 추가한 후, 샘플 데이터를 작성.

01 1627263849001, 1627175707, 어벤져스, B-4 ----- 좌석명

발급번호(대포깅) ----- 예매 영화의 대포깅 ----- 예매 영화 제목

02 1627376546871, 1627175946, 컨저링, D-6

03 1627392342611, 1627175983, 러부 액츄얼리, C-4

04 ----- 개행 문자 '\n'에 의해 빈 줄 추가

reservations.txt 생성

프로젝트에 reservations.txt 파일을 추가한 후, 샘플 데이터를 작성.

II. 프로그램 실행과 종료

프로그램 실행과 종료 기능을 구현

- 이를 위해 부모 클래스 AbstractMenu와
- 자식 클래스 MainMenu를 작성

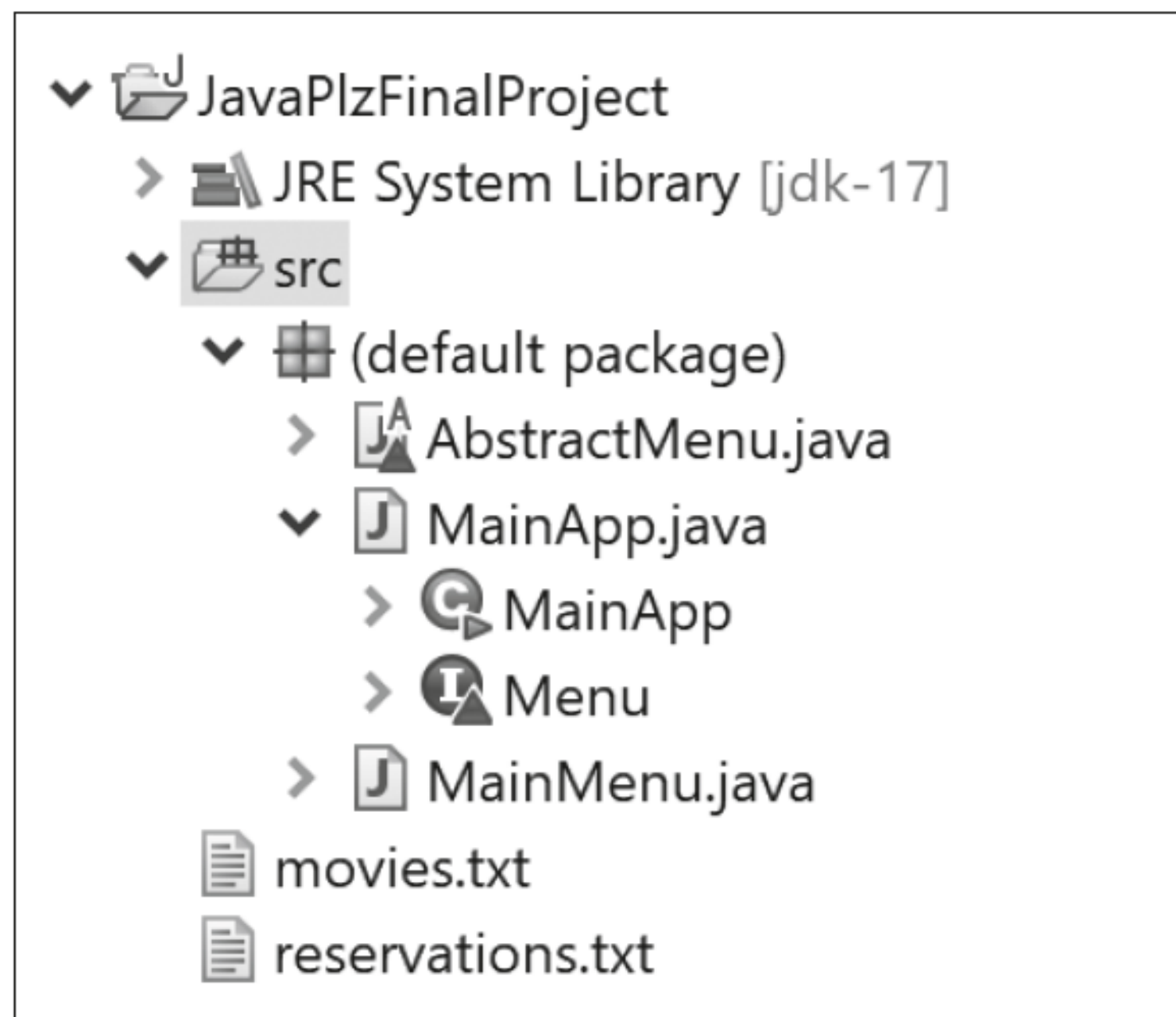



그림 13-6 프로그램 실행과 종료 구현 후 프로젝트 파일 구조

```
01 import java.util.Scanner;
02
03 abstract class AbstractMenu implements Menu {
04     protected String menuText;    // 기본 문구
05     protected Menu prevMenu;      // 이전 메뉴
06     protected static final Scanner scanner = new Scanner(System.in);
07
08     public AbstractMenu(String menuText, Menu prevMenu) { // 생성자
09         this.menuText = menuText;
10         this.prevMenu = prevMenu;
11     }
12
13     public void print() {
14         System.out.print("\n" + menuText);    // 메뉴 출력
15     }
16
17     public void setPrevMenu(Menu prevMenu) { // setter 메소드
18         this.prevMenu = prevMenu;
19     }
20 }
```

 **TIP** 추상 클래스는 인터페이스의 추상 메소드를 재정의하지 않아도 에러가 발생하지 않음

AbstractMenu.java 생성

Abstract 클래스는 추상 클래스로 작성하는데, 이는 Menu 인터페이스를 구현하며, 추후 메인 메뉴(MainMenu)와 관리자 메뉴(AdminMenu)의 부모 클래스가 됨.

- **04~06행:** 기본 문구와 이전 메뉴를 인스턴스변수로, 키보드 입력을 위한 스캐너 객체를 클래스변수로 가짐
- **13~15행:** Menu 인터페이스의 print() 메소드를 재정의한 것. 메뉴의 기본 문구를 출력
- **17~18행:** setter 메소드. 메뉴를 종료한 후 돌아갈 이전 메뉴를 설정


```

01 class MainMenu extends AbstractMenu {-----AbstractMenu를 상속받음
02     private static final MainMenu instance = new MainMenu(null);---자기 자신의
03     private static final String MAIN_MENU_TEXT = // 기본 문구 객체 생성
04         "1: 영화 예매하기\n" +
05         "2: 예매 확인하기\n" +
06         "3: 예매 취소하기\n" +
07         "4: 관리자 메뉴로 이동\n" +
08         "q: 종료\n\n" +
09         "메뉴를 선택하세요: ";
10     -----private 생성자(외부에서 객체 생성 불가)
11     private MainMenu(Menu prevMenu) {
12         super(MAIN_MENU_TEXT, prevMenu);          // 부모 생성자 호출
13     }
14
15     public static MainMenu getInstance() {
16         return instance;                          // 메뉴 객체 반환
17     }
18
19     public Menu next() {
20         switch (scanner.nextLine()) {-----사용자 입력을 행 단위로 가져옴
21             case "q": return prevMenu;           // q 입력 시, prevMenu를 반환
22             default: return this; // 그 외 입력은 메인 메뉴(this)로 돌아감
23         }
24     }
25 }

```

MainMenu.java 생성

메인 메뉴 클래스를 작성하는데, 이는 AbstractMenu의 자식 클래스로 사용자와의 상호작용을 담당함. 메인 메뉴의 출력과 여러 선택지에 따른 기능을 관리. 해당 클래스는 점진적으로 개선될 예정.

- **02행:** 클래스변수를 통해 자기 자신의 객체를 생성
- **11~13행:** private 생성자. 생성자가 외부에 노출되지 않아 생성자 호출 불가 즉, 객체를 생성할 수 없음. 오직 02행의 클래스변수 초기화 시 단 한번 호출되어 하나의 객체만을 생성. 이러한 패턴을 **싱글턴**(singleton)이라 함.
- **19~24행:** Menu 인터페이스의 next() 메소드를 재정의한 것. 사용자 입력에 따른 다음 메뉴로의 진행을 결정. 사용자 입력이 q인 경우 02행에 의해 prevMenu가 null이므로, 메인 메소드의 08행에서 이를 받아 프로그램을 종료. 그 외의 입력은 다시 자기 자신을 반환

II. 프로그램 실행과 종료

중간 점검: '프로그램 실행과 종료' 기능 시연

① 프로그램 실행	프로그램을 시작합니다!
② 메인 메뉴 출력	1: 영화 예매하기 2: 예매 확인하기 3: 예매 취소하기 4: 관리자 메뉴로 이동 q: 종료
③ 이동할 메뉴 선택	메뉴를 선택하세요: q
④ 프로그램 종료	프로그램이 종료됩니다.



그림 13-7 현재까지 구현된 기능

III. 메뉴간 이동

메인 메뉴에서 관리자 메뉴로, 또는 그 반대로 이동 기능을 구현

- 이를 위해 MainMenu 기능을 추가하고
- AdminMenu 클래스를 생성한다

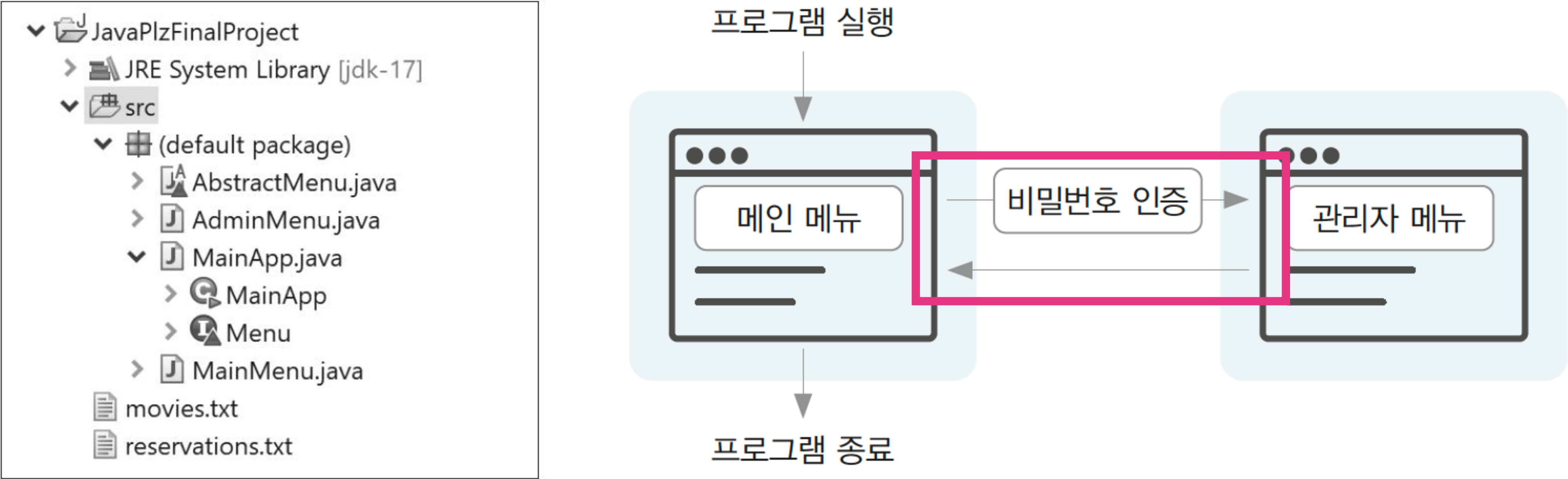


그림 13-8 메뉴 간 이동 구현 후 프로젝트 파일 구조


```
01 class MainMenu extends AbstractMenu {
...
19 public Menu next() {
20     switch (scanner.nextLine()) {
21         case "4":
22             if ( ! checkAdminPassword()) { // 관리자 비밀번호 확인
23                 System.out.println(">> 비밀번호가 틀렸습니다.");
24                 return this; // 실패한 경우 메인 메뉴 객체 반환
25             }
26             AdminMenu adminMenu = AdminMenu.getInstance();
27             adminMenu.setPrevMenu(this); // 메인 메뉴를 이전 메뉴로 등록
28             return adminMenu; // 관리자 메뉴 객체를 반환
29         case "q": return prevMenu; // q 입력 시, null을 반환
30         default: return this; // 그 외 입력은 다시 메인 메뉴로 돌아감
31     }
32 }
33
34 private boolean checkAdminPassword() {
35     System.out.print("관리자 비밀번호를 입력하세요: ");
36     return "admin1234".equals(scanner.nextLine());
37 }
38 }
```

새로 추가된 코드

관리자 비밀번호 비교할 사용자 입력

MainMenu.java 기능 추가

관리자 메뉴로 이동하는 기능을 추가. 비밀번호 인증을 거쳐야 이동할 수 있음. 해당 클래스는 점진적으로 개선 될 예정.

- **21~28행:** next() 메소드에 4번 선택지를 추가하여 관리자 메뉴로의 이동을 구현. 22행의 비밀번호 인증이 통과되면 26행에서 관리자 객체를 가져옴. 27행은 현재 메인 메뉴를 관리자 객체가 복귀할 이전 메뉴로 등록. 28행은 관리자 메뉴 객체를 반환
- **34~37행:** checkAdminPassword() 메소드. 관리자 메뉴 진입 전 비밀번호를 인증. equals() 메소드를 사용해 문자열을 비교

여기서 자카니 문자열이 비교

```
String str1 = "Hello";
String str2 = "Hello";
String str3 = new String("Hello");
System.out.println("str1 == str2 ? " + (str1 == str2)); // true
System.out.println("str1 == str3 ? " + (str1 == str3)); // false
```

```

01 public class AdminMenu extends AbstractMenu {-----AbstractMenu를 상속받음
02     private static final AdminMenu instance = new AdminMenu(null);
03     private static final String ADMIN_MENU_TEXT = // 기본 문구
04         "1: 영화 등록하기\n" +
05         "2: 영화 목록 보기\n" +
06         "3: 영화 삭제하기\n" +
07         "b: 메인 메뉴로 이동\n\n" +
08         "메뉴를 선택하세요: ";
09     -----private 생성자(외부에서 객체 생성 불가)
10     private AdminMenu(Menu prevMenu) {
11         super(ADMIN_MENU_TEXT, prevMenu); // 부모 생성자 호출
12     }
13
14     public static AdminMenu getInstance() {
15         return instance; // 메뉴 객체를 반환
16     }
17
18     public Menu next() {
19         switch (scanner.nextLine()) {
20             case "b": return prevMenu; // b 입력 시, 이전 메뉴를 반환
21             default: return this;
22         }
23     }
24 }

```

AdminMenu.java 생성

관리자 메뉴(AdminMenu) 클래스를 작성. AbstractMenu의 자식 클래스로, 사용자와의 상호작용을 담당하며 관리자 메뉴의 여러 선택지에 따른 기능을 관리. 해당 클래스는 점진적으로 개선될 예정.

- **02행:** 클래스변수 초기화를 통해 자기 자신의 객체를 생성
- **10~12행:** private 생성자. 외부에서 객체 생성이 불가능. 02행의 클래스변수와 함께 싱글턴 패턴을 구현
- **18~23행:** Menu 인터페이스의 next() 메소드를 재정의한 것. 사용자 입력에 따른 다음 메뉴로의 진행을 결정한다. 사용자 입력이 b인 경우, prevMenu를 반환. 03행에 의해 prevMenu의 초기값은 null이지만 실행 중 세터에 의해 언제든지 바뀔 수 있음

III. 메뉴간 이동

중간 점검: '메뉴간 이동' 기능 시연

① 메인 메뉴 출력	1: 영화 예매하기 2: 예매 확인하기 3: 예매 취소하기 4: 관리자 메뉴로 이동 q: 종료
② 이동할 메뉴 선택	메뉴를 선택하세요: 4
③ 비밀번호 인증	관리자 비밀번호를 입력하세요: admin1234
④ 관리자 메뉴 출력	1: 영화 등록하기 2: 영화 목록 보기 3: 영화 삭제하기 b: 메인 메뉴로 이동
⑤ 이동할 메뉴 선택	메뉴를 선택하세요: b
⑥ 메인 메뉴 복귀	1: 영화 예매하기 2: 예매 확인하기 3: 예매 취소하기 4: 관리자 메뉴로 이동 q: 종료 메뉴를 선택하세요:

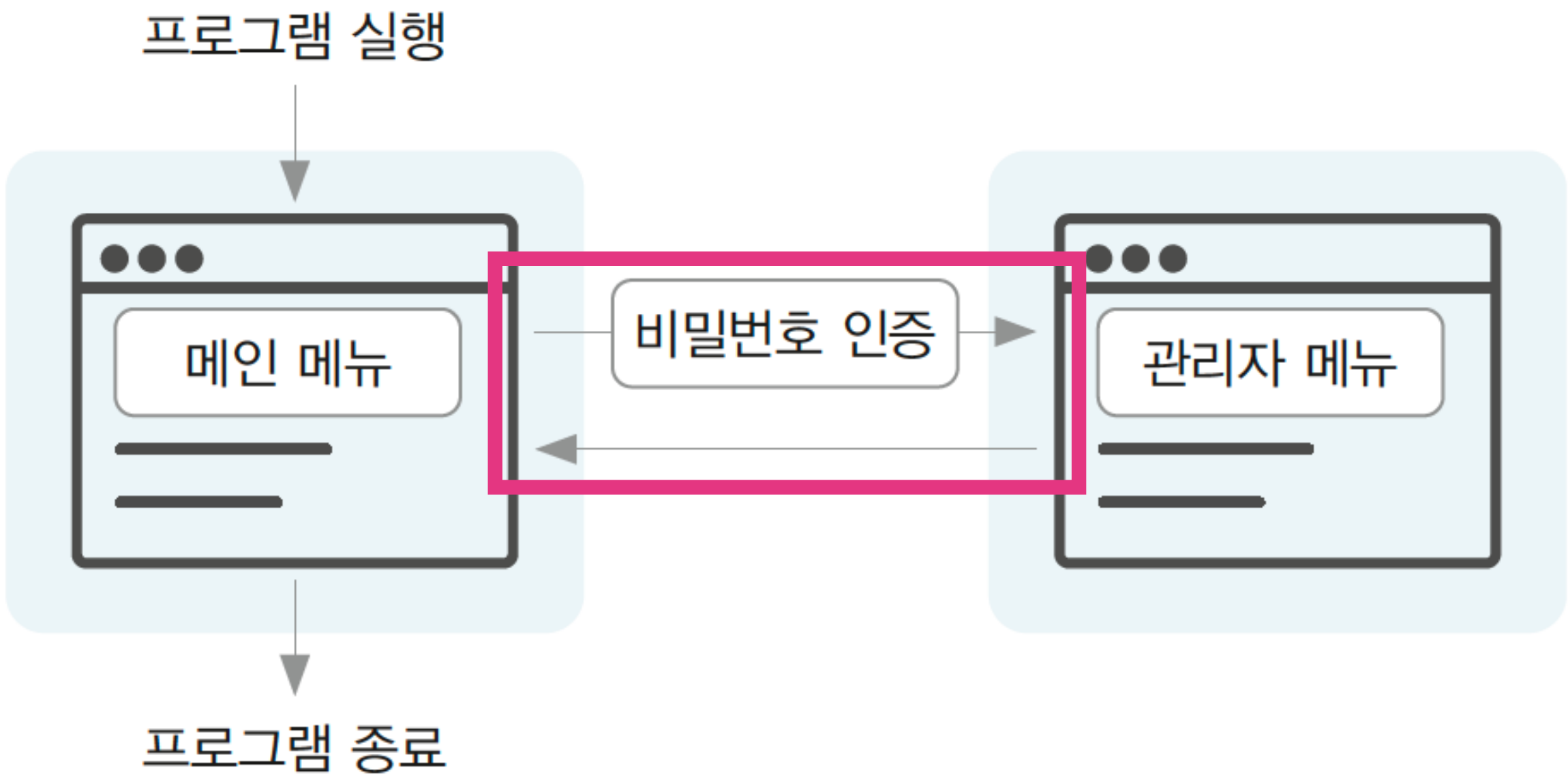


그림 13-9 현재까지 구현된 기능