

Java Programming II

Lab3

514770-1

Fall 2026

9/22/2026

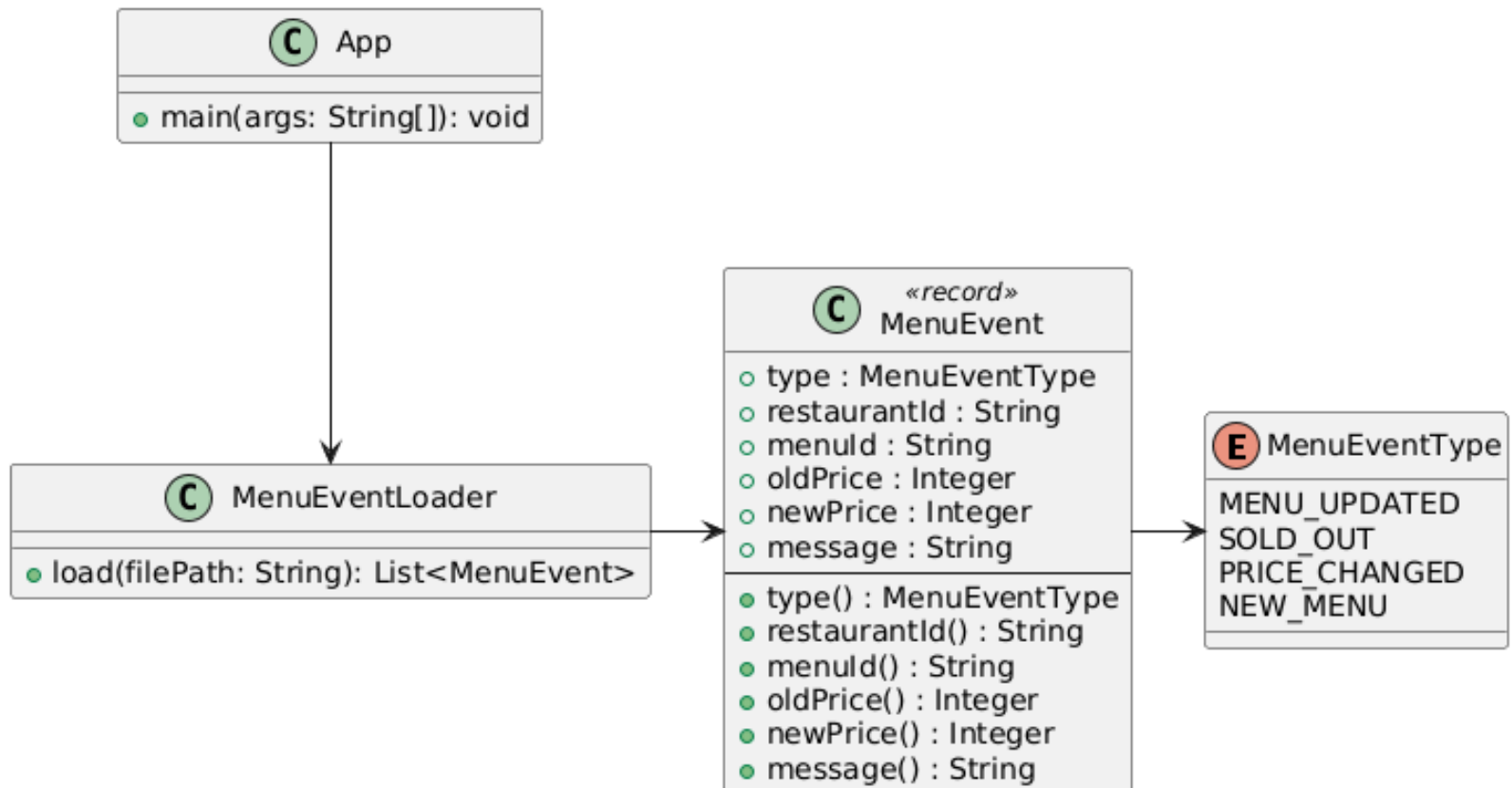
Kyoung Shin Park
Computer Engineering
Dankook University

Lab3

- 메뉴 변경, 품질, 가격 변동 등 학생식당 메뉴 이벤트를 등록된 관찰자 전체에 뿌리는 **허브**와, 그 이벤트를 받아 해당 식당 구독자에게만 통지하는 **구독 관리자**를 둔 2단 Observer 체인을 구현하라.
 - **CampusMenuHub**는 **IMenuEventSubject**로서 메뉴 변경, 품질, 가격 변동 등 메뉴 이벤트가 발생하면 이를 모든 **IMenuEventObserver**들에게 통지
 - **SubscriptionManager**는 **IMenuEventObserver**이자 동시에 **ISubscriptionSubject**로서 허브로부터 메뉴 이벤트를 수신하면 특정 식당을 구독한 사용자들만 이벤트를 전달
 - 즉, 최종적으로 **CampusMenuHub -> SubscriptionManager -> 구독된 사용자 뷰**

Lab3

- **App**은 MenuEventLoader를 이용하여 events.json 데이터를 로딩하여 List<MenuEvent>로 저장한다.



Lab3

```
{"type": "MENU_UPDATED", "restaurantId": "R01", "message": "51장국밥 오늘의 메뉴가 갱신되었습니다"}  
{"type": "SOLD_OUT", "restaurantId": "R06", "menuId": "6", "message": "오색비빔밥 품절"}  
{"type": "PRICE_CHANGED", "restaurantId": "R09", "menuId": "53", "oldPrice": 3500, "newPrice": 4000, "message": "폭풍라면 3500→4000"}  
{"type": "NEW_MENU", "restaurantId": "R05", "menuId": "99", "message": "신메뉴: 춘천닭갈비덮밥(킹)"}  
{"type": "SOLD_OUT", "restaurantId": "R03", "menuId": "18", "message": "특 등심왕돈카츠 품절"}  
{"type": "MENU_UPDATED", "restaurantId": "R08", "message": "중식대장 점심 메뉴 갱신"}  
{"type": "PRICE_CHANGED", "restaurantId": "R02", "menuId": "10", "oldPrice": 6500, "newPrice": 6200, "message": "돼지김치찌개 할인"}
```

Lab3

```
public enum MenuEventType {  
    MENU_UPDATED,  
    SOLD_OUT,  
    PRICE_CHANGED,  
    NEW_MENU  
}
```

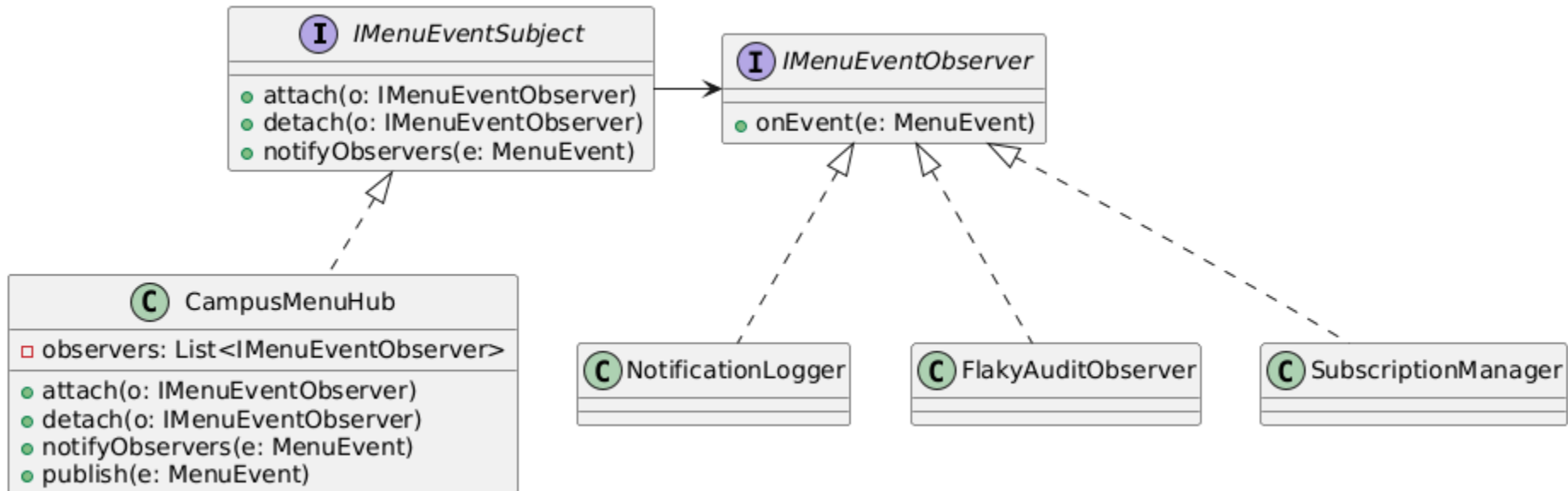
```
public record MenuEvent(  
    MenuEventType type,  
    String restaurantId,  
    String menuId,  
    Integer oldPrice,  
    Integer newPrice,  
    String message  
) {}
```

Lab3

```
public class MenuEventLoader {
    public static List<MenuEvent> load(String path)
throws Exception {
    List<MenuEvent> events = new ArrayList<>();
    try (BufferedReader br = new BufferedReader(
        new InputStreamReader(new
FileInputStream(path), StandardCharsets.UTF_8))) {
        String line;
        while ((line = br.readLine()) != null) {
            if (line.isBlank()) continue;
            events.add(parse(line));
        }
    }
    return events;
} ...
```

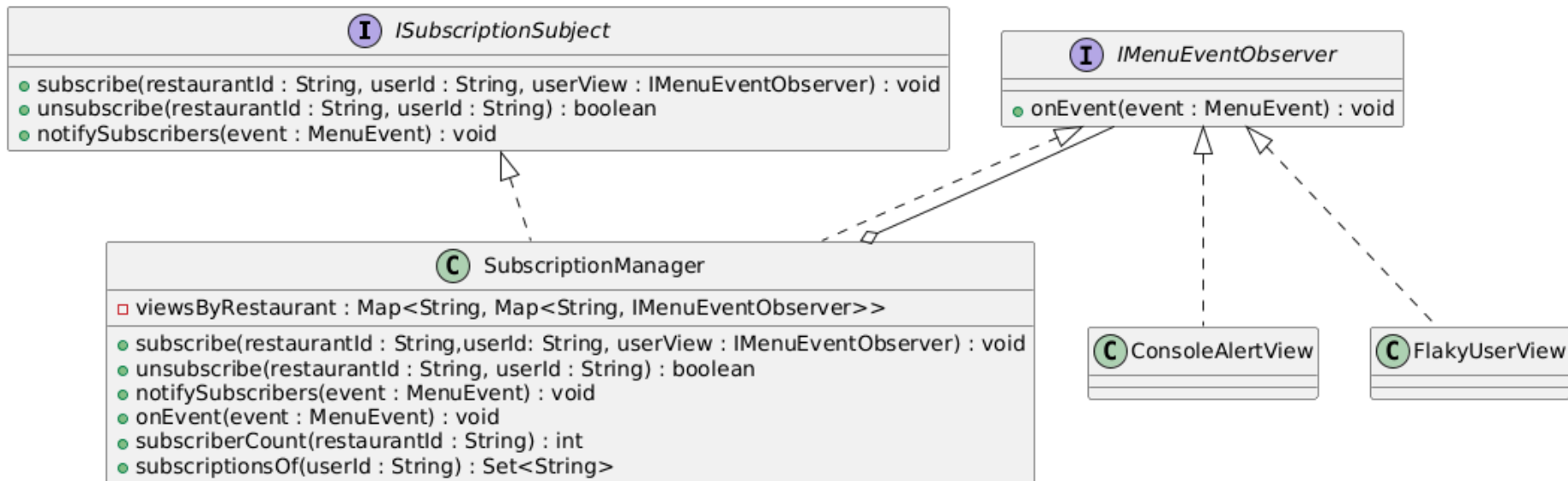
Lab3

- ❑ **CampusMenuHub**는 NotificationLogger, FlakyAuditObserver, SubscriptionManager 옵저버를 추가하고, 메뉴 이벤트 publish 를 호출한다.
 - **publish(MenuEvent event)**는 notifyObservers를 호출한다.
 - **notifyObservers(MenuEvent event)**는 통지중 옵저버가 스스로를 detach해도 안전하도록 복사본 List.copyOf(observers)을 순회한다.



Lab3

- **SubscriptionManager**는 각 식당에 대해 ConsoleAlertView와 FlakyUserView 등 옵저버를 추가하고, CampusMenuHub로부터 이벤트가 호출되면 (onEvent) notifySubscribers로 구독된 사용자에게 통지한다. (즉, restaurantId에 맞는 구독자들만 골라서 다시 onEvent를 전달)



Lab3

```
public class SubscriptionManager implements
IMenuEventObserver, ISubscriptionSubject {
    private Map<String, Map<String,
IMenuEventObserver>> viewsByResaurant = new
HashMap<>();

    ...
    @Override
    public void subscribe(String restaurantId,
String userId, IMenuEventObserver userView) {
        // 셋 중 하나라도 null이면 구독 자체가 성립하지
않는다.
        // 저장 구조는 Map<식당, Map<사용자, 뷰>> 의 2단
중첩
        // 식당별 구독자 맵에 사용자→뷰를 등록한다. 맵이
없으면 생성하고, 같은 사용자면 뷰를 교체한다.
    }
}
```

Lab3

```
public class ConsoleAlertView implements
IMenuEventObserver { // 사용자별 콘솔 알람뷰
    private final String userId;

    public ConsoleAlertView(String userId) {
        this.userId = userId;
    }

    @Override
    public void onEvent(MenuEvent event) {
        System.out.print f("[ALERT:%s] (%s) %s –
        %s%n", userId, event.type(), event.restaurantId(),
event.message());
    }
}
```

Lab3

```
public class FlakyUserView implements
IMenuEventObserver { // 일부러 실패하는 사용자 뷰
    private final String userId;

    public FlakyUserView(String userId) {
        this.userId = userId;
    }

    @Override
    public void onEvent(MenuEvent event) {
        throw new IllegalStateException(userId + " 단말 연결
끊김");
    }
}
```

Lab3

```
public static void main(String[] args) throws Exception {  
    List<MenuEvent> events = ..  
    // Subject  
    CampusMenuHub hub = new CampusMenuHub();  
    // Observer  
    NotificationLogger logger = new NotificationLogger();  
    FlakyAuditObserver flaky = new FlakyAuditObserver();  
    SubscriptionManager subs = new SubscriptionManager();  
    hub.attach(logger);  
    hub.attach(flaky);  
    hub.attach(subs);  
    // View Observer  
    ConsoleAlertView alice = new ConsoleAlertView("alice");  
    ConsoleAlertView bob = new ConsoleAlertView("bob");  
    subs.subscribe("R01", "alice", alice);  
    subs.subscribe("R06", "alice", alice);  
    subs.subscribe("R09", "bob", bob);  
    subs.subscribe("R03", "bob", bob);  
}
```

Lab3

```
// event replay..  
for (MenuEvent event : events) {  
    hub.publish(event);  
    Thread.sleep(500); // sleep  
}  
// A broken view must not stop its siblings  
List<String> got = new ArrayList<>();  
subs.subscribe("R01", "carol", e -> got.add("carol"));  
subs.subscribe("R01", "dave", new FlakyUserView("dave"));  
subs.subscribe("R01", "erin", e -> got.add("erin"));  
hub.publish(new MenuEvent(MenuEventType.SOLD_OUT, "R01",  
"12", null, null, "돈까스 품절"));  
// detach ..  
// add new MenuEvent..
```

Lab3

```
[LOG] MENU_UPDATED|R01|51장국밥  오늘의 메뉴가 갱신되었습니다
[ALERT:alice] (MENU_UPDATED) R01 - 51장국밥  오늘의 메뉴가 갱신되었습니다
[LOG] SOLD_OUT|R06|오색비빔밥  품절
[ALERT:alice] (SOLD_OUT) R06 - 오색비빔밥  품절
[LOG] PRICE_CHANGED|R09|폭풍라면  3500→4000
[WARN] 관찰자 처리 실패: FlakyAuditObserver - 감사 서버 응답 없음
[ALERT:bob] (PRICE_CHANGED) R09 - 폭풍라면  3500→4000
[LOG] NEW_MENU|R05|신메뉴: 춘천닭갈비덮밥(킹)
[LOG] SOLD_OUT|R03|특 등심왕돈카츠 품절
[ALERT:bob] (SOLD_OUT) R03 - 특 등심왕돈카츠 품절
[LOG] MENU_UPDATED|R08|중식대장 점심 메뉴 갱신
[LOG] PRICE_CHANGED|R02|돼지김치찌개  할인
[WARN] 관찰자 처리 실패: FlakyAuditObserver - 감사 서버 응답 없음

...
[LOG] SOLD_OUT|R01|돈까스  품절
[ALERT:alice] (SOLD_OUT) R01 - 돈까스  품절
[WARN] 구독라우터[R01] → FlakyUserView  통지 실패: dave  단말 연결 끊김

...
```

Submit to e-learning

- 메인에 Observer 추가 삭제 루틴 포함할 것
- Lab3 과제에 yourcode (e.g.: 다른 Observer 추가 등)를 추가 (yourcode 없을시 10점에서 -1점 감점)
- **Java26-2-HW3-YourID-YourName.zip** 과제(보고서에 반드시 yourcode 설명 포함)를 e러닝에 제출 (**due by 9/28**).