ui_state_step3.dart

소스 코드

전체 코드

```dart
import 'package:flutter/material.dart';
import 'package:freezed_annotation/freezed_annotation.dart';

part 'ui_state_step3.freezed.dart';

@freezed
sealed class UIStateStep3 with _$UIStateStep3 {

  const factory UIStateStep3.init() = UIStateStep3Init;
  const factory UIStateStep3.closeDoor() = UIStateStep3CloseDoor;
  const factory UIStateStep3.completed() = UIStateStep3Completed;
}
```

1️⃣ 파일 개요

파일명: `lib/model/ui_state_step3.dart`
역할: Step3 화면의 UI 상태 관리용 데이터 클래스 정의
주요 목적:

- Step3 화면의 상태를 **불변 객체로 관리**
- `freezed` 패키지를 사용하여 **sealed class + union type** 구조 제공
- 상태 전환(`init` → `closeDoor` → `completed`)을 명확하게 표현

사용 맥락 예시:

- Step3 화면에서 장치 문 닫기(Close Door) 작업 처리
- 초기 상태, 문 닫기 중 상태, 완료 상태를 관리

2️⃣ 주요 기능

1. **초기 상태(`init`)**
    - Step3 화면 진입 시 기본 상태
    - UI 초기화 및 준비 상태 표시
2. **문 닫기 상태(`closeDoor`)**
    - 사용자가 문 닫기 버튼 클릭 시
    - 문 닫는 애니메이션 또는 처리 중 상태 표시
3. **완료 상태(`completed`)**
    - 문 닫기 작업 완료 후 상태

- 다음 단계 진행 또는 완료 표시

4. **불변성 유지**
    - `copyWith` 메서드 자동 생성으로 안전하게 상태 갱신 가능

---

3 구조 분석

```
UIStateStep3 (sealed class)
 ├─ UIStateStep3Init       : 초기 상태
 ├─ UIStateStep3CloseDoor  : 문 닫기 중 상태
 └─ UIStateStep3Completed  : 완료 상태
```

특징:

- Step3는 **단계적 상태 전환**이 필요
- `freezed`를 사용하여 타입 안전성 + 패턴 매칭 지원
- 추가 데이터가 필요 없는 경우 간단하게 상태만 관리 가능

---

4 동작 흐름

1. Step3 화면 초기화 시 `UIStateStep3.init` 상태 생성
2. 사용자가 문 닫기 버튼 클릭 시 `Step3` ViewModel에서 처리
3. `UIStateStep3.closeDoor` 상태로 전환 → UI에서 문 닫는 애니메이션 표시 가능
4. 처리 완료 시 `UIStateStep3.completed` 상태로 전환
5. 상태 변경 시 화면은 `ConsumerWidget` 또는 `StateNotifier`를 통해 리빌드

---

5 장점

- **단계적 상태 관리 가능:** init → closeDoor → completed
- **패턴 매칭 지원:** `when` / `maybeWhen` / `map` 사용 가능
- **불변 객체 관리:** 안전한 상태 갱신 가능 (`copyWith`)
- 테스트 및 유지보수 용이

---

6 단점 / 개선점

1. **데이터 없음**
    - 상태에 추가 정보가 없어서 진행률, 에러 메시지 등을 표현하기 어려움
    - 개선: 필요 시 `progress`, `errorMessage` 등 필드 추가 가능
2. **UI 의존성 없음**
    - 장점이지만, View에서 필요한 데이터를 직접 관리해야 함
    - ViewModel에서 로직 처리 후 상태만 갱신하도록 명확히 해야 함

7️⃣ 개선 구조 예시 (개념)

```
@freezed
sealed class UIStateStep3 with _$UIStateStep3 {
  const factory UIStateStep3.init() = UIStateStep3Init;
  const factory UIStateStep3.closeDoor({@Default(0.0) double progress}) =
UIStateStep3CloseDoor;
  const factory UIStateStep3.completed({String? resultMessage}) = UIStateStep3Completed;
}
```

- **장점:** 진행률 표시, 완료 메시지 등 추가 가능
- **단점:** 간단한 경우 불필요한 필드가 생길 수 있음