ui_state_step4.dart

소스 코드

전체 코드

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

part 'ui_state_step4.freezed.dart';

@freezed
sealed class UIStateStep4 with _$UIStateStep4 {

  const factory UIStateStep4.init() = UIStateStep4Init;
  const factory UIStateStep4.retry() = UIStateStep4Retry;
  const factory UIStateStep4.checked() = UIStateStep4Checked;
}
```

1️⃣ 파일 개요

파일명: `lib/model/ui_state_step4.dart`
역할: Step4 화면의 UI 상태 관리용 데이터 클래스 정의
주요 목적:

- Step4 화면의 상태를 **불변 객체로 관리**
- `freezed` 패키지를 사용하여 **sealed class + union type** 구조 제공
- 상태 전환(`init` → `retry` → `checked`)을 명확하게 표현

사용 맥락 예시:

- Step4 화면에서 특정 동작 검증 또는 체크 후 처리
- 초기 상태, 재시도 상태, 완료/확인 상태를 관리

2️⃣ 주요 기능

1. **초기 상태(`init`)**
    - Step4 화면 진입 시 기본 상태
    - UI 초기화 및 준비 상태 표시
2. **재시도 상태(`retry`)**
    - 사용자 동작 실패 시
    - 재시도 버튼 클릭 또는 오류 발생 시 표시
3. **확인/완료 상태(`checked`)**
    - 동작 성공 후 완료 상태

- UI에서 체크 완료 또는 다음 단계 진행 표시

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

---

3 구조 분석

```
UIStateStep4 (sealed class)
 ├─ UIStateStep4Init     : 초기 상태
 ├─ UIStateStep4Retry    : 재시도 상태
 └─ UIStateStep4Checked  : 완료/확인 상태
```

특징:

- Step4 화면에서 **상태 기반 UI 전환** 용이
- `freezed`를 사용하여 타입 안전성 + 패턴 매칭 지원
- 추가 데이터가 필요 없는 경우 간단하게 상태만 관리 가능

---

4 동작 흐름

1. Step4 화면 초기화 시 `UIStateStep4.init` 상태 생성
2. 사용자 동작 실패 시 ViewModel에서 `UIStateStep4.retry` 상태로 전환
3. 동작 성공 시 `UIStateStep4.checked` 상태로 전환
4. 상태 변경 시 화면은 `ConsumerWidget` 또는 `StateNotifier`를 통해 리빌드

---

5 장점

- **단계적 상태 관리 가능:** init → retry → checked
- **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 오류 메시지, 진행률 등을 표현하기 어려움
    - 개선: 필요 시 `errorMessage`, `progress` 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 UIStateStep4 with _$UIStateStep4 {
  const factory UIStateStep4.init() = UIStateStep4Init;
  const factory UIStateStep4.retry({String? errorMessage}) = UIStateStep4Retry;
  const factory UIStateStep4.checked({bool? isConfirmed}) = UIStateStep4Checked;
}
```

- **장점:** 오류 메시지, 확인 여부 등 추가 가능
- **단점:** 간단한 경우 불필요한 필드가 생길 수 있음