ui_state_step1.dart

소스 코드

전체 코드

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

part 'ui_state_step1.freezed.dart';

@freezed
sealed class UIStateStep1 with _$UIStateStep1 {

  const factory UIStateStep1.input({
    @Default("") String phoneNumber,
    required TextEditingController controller,
    @Default(false) bool showErrorToast,
  }) = UIStateStep1Input;

  const factory UIStateStep1.completed({
    @Default("") String phoneNumber,
    required TextEditingController controller,
    @Default(false) bool showErrorToast,
  }) = UIStateStep1Completed;
}
```

1️⃣ 파일 개요

파일명: `lib/model/ui_state_step1.dart`
역할: Step1 화면의 UI 상태 관리용 데이터 클래스 정의
주요 목적:

- 전화번호 입력 화면의 상태를 **불변 객체로 관리**
- `freezed` 패키지를 사용하여 **sealed class + union type** 구조 제공
- 상태 전환(`input` → `completed`)을 명확하게 표현
- `TextEditingController`를 상태에 포함시켜 **텍스트 동기화 및 관리**

사용 맥락 예시:

- Step1 화면에서 전화번호 입력 UI 구현
- 입력 상태, 완료 상태, 에러 토스트 표시 상태 관리

2️⃣ 주요 기능

1. **입력 상태(`input`)**
    - 전화번호(`phoneNumber`)를 실시간 저장

- `TextEditingController`를 포함하여 텍스트 필드와 동기화
- 에러 토스트 표시 여부(`showErrorToast`) 포함

2. **완료 상태(`completed`)**
   - 입력이 완료된 후 상태 표현
   - 전화번호와 컨트롤러 상태 유지
   - 에러 토스트 표시 여부 유지
3. **불변성 유지**
   - `copyWith` 메서드 자동 생성으로 안전하게 상태 갱신 가능

---

3 구조 분석

```
UIStateStep1 (sealed class)
 ├─ UIStateStep1Input (input 상태)
 |     ├─ phoneNumber: String
 |     ├─ controller: TextEditingController
 |     └─ showErrorToast: bool
 └─ UIStateStep1Completed (completed 상태)
       ├─ phoneNumber: String
       ├─ controller: TextEditingController
       └─ showErrorToast: bool
```

특징:

- `input`과 `completed` 두 가지 상태만 존재
- `TextEditingController` 포함으로 텍스트 입력 동기화 가능
- `freezed`를 사용하여 **타입 안전성 + 패턴 매칭** 지원

---

4 동작 흐름

1. Step1 화면 초기화 시 `UIStateStep1.input` 상태 생성
2. 사용자가 전화번호 입력 시 `TextEditingController`가 `phoneNumber`를 갱신
3. 완료 버튼 눌렀을 때 `UIStateStep1.completed`로 상태 전환
4. 에러 발생 시 `showErrorToast: true`로 상태 갱신 후 토스트 표시
5. 상태 변경 시 화면은 `ConsumerWidget` 또는 `StateNotifier`를 통해 리빌드

---

5 장점

- **상태 구분 명확:** 입력/완료 상태를 union type으로 구분
- **컨트롤러 포함:** 텍스트 필드와 상태를 일치시킬 수 있음
- **불변 객체 관리:** 안전한 상태 갱신 가능 (`copyWith`)
- **패턴 매칭 지원:** `when` / `maybeWhen` / `map` 사용 가능

6 단점 / 개선점

1. **컨트롤러 관리 부담**
    - 상태 객체가 `TextEditingController`를 직접 포함
    - 메모리 관리(`dispose`)를 ViewModel 또는 위젯에서 신경 써야 함
2. **상태 확장성 제한**
    - 현재는 `input`과 `completed` 두 가지 상태만
    - 필요 시 로딩, 에러, 검증 실패 등 상태 추가 가능
3. **UI 의존성 존재**
    - 상태 객체에 Flutter 위젯(`TextEditingController`) 포함
    - 테스트 시 Flutter 환경 의존성이 생김
    - 개선: 컨트롤러 대신 String만 관리하고, 위젯에서 연결 가능

---

7 개선 구조 예시 (개념)

```
@freezed
sealed class UIStateStep1 with _$UIStateStep1 {
  const factory UIStateStep1.input({
    @Default("") String phoneNumber,
    @Default(false) bool showErrorToast,
  }) = UIStateStep1Input;

  const factory UIStateStep1.completed({
    @Default("") String phoneNumber,
  }) = UIStateStep1Completed;
}
```

- **장점:** 테스트 용이, UI 의존성 제거
- **단점:** TextEditingController 동기화는 View 또는 ViewModel에서 별도 처리 필요