Config.dart

소스 코드

전체 코드

```dart
class Config {
  static final Config instance = Config._internal();
  bool isDebugMode = false;

  Config._internal();

  void setDebugMode(bool value) {
    isDebugMode = value;
  }
}
```

역할 & 목적

- **전역(Global) 설정값을 저장**하기 위한 싱글톤(Singleton) 비슷한 구조
- 앱 초기 실행(`main()`)에서 생성한 Config 인스턴스를
  어디서든 `Config.instance`를 통해 접근할 수 있도록 하는 패턴
- **현재 용도: 디버그 모드 여부 저장**

동작 방식 설명

1) `Config` 생성자 호출 시

```dart
Config(isDebugMode: true);
```

- 객체가 하나 생성됨
- 그리고 생성자 내부에서

```dart
instance = this;
```

를 통해 **그 객체를 static 변수에 저장**
→ 다른 파일에서도 어디서나 `Config.instance.isDebugMode` 사용 가능.

2) static late final

```dart
static late final Config instance;
```

- `late`: 나중에 초기화할 수 있음 (초기화 전 접근하면 에러)
- `final`: **한 번만 대입 가능**
- 즉, 앱 실행 내내 **Config는 한 번만 초기화 가능**함

디자인 패턴 관점

이건 명확히 **싱글톤 패턴이 아님**
(싱글톤이라면 보통 생성자를 private으로 막음)

하지만 효과적으로는 "전역 단일 설정 객체" 역할을 수행함.

주석 버전 코드

```dart
/// App 전역 설정을 관리하기 위한 Config 클래스.
/// 생성자 호출 시 자신을 static 변수(instance)에 저장하여
/// 어디서나 Config.instance 로 접근 가능하도록 하는 구조.
///
/// 주의:
/// - 사실상 싱글톤처럼 동작하지만, 생성자가 public이므로
///   여러 번 호출하면 second initialization error가 발생할 수 있음.
/// - main()에서 한 번만 생성해야 한다.
class Config {
  /// 전역에서 접근할 단일 instance
  static late final Config instance;

  /// 디버그 모드 활성 여부
  final bool isDebugMode;

  /// 생성자: Config 생성 시 현재 객체를 static instance 에 저장한다.
  /// 앱 전체에서 하나의 Config만 쓰는 구조.
  Config({ required this.isDebugMode }) {
    instance = this;
  }
}
```