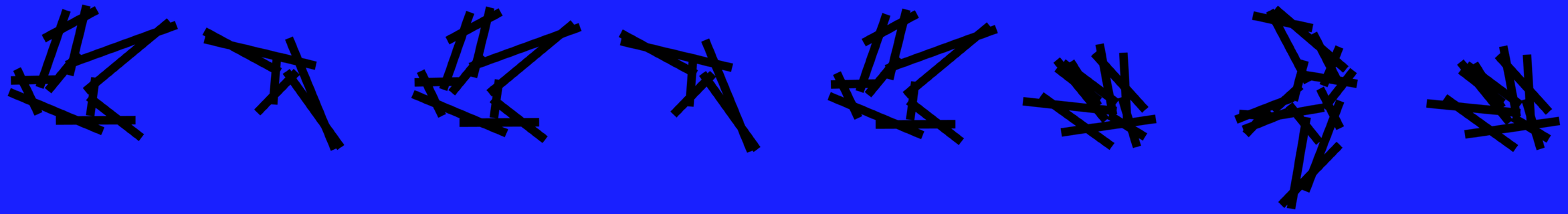
SyncSwift
Conference
2022

# Modular Architecture 시작하기

**오강훈,** 당근마켓



Contents

# 모듈 알아보기

모듈 알아보기

# Modular Architecture?

**Modular programming** is a software design technique that emphasizes separating the functionality of a program into independent, interchangeable **modules**, such that each contains everything necessary to execute only one aspect of the desired functionality.

- wikipedia -

모듈 알아보기

# Modular Architecture?

**모듈형 프로그래밍**은 프로그램의 기능을 **독립적이고 교체 가능한** 모듈로 분리하는 것을 강조하는 소프트웨어 설계 기법

모듈 알아보기

# 모듈이란?

**앱에서 재사용하기 위한 코드 덩어리**

- 여기서는 Library, Framework, Package 를 의미

예) 계정 모듈 (Account)

- 계정 정보, 로그인, 로그아웃 등 계정 관련 코드가 모여있는 곳

# 왜 모듈을 분리하나요?

배경

## 커져가는 팀

1개의 팀, 3명의 iOS Engineer

배경

## 커져가는 팀

10개의 팀, 12명의 iOS Engineer

배경

## 커져가는 팀

팀 규모는 4배가 되었지만

생산성 저하가 발생

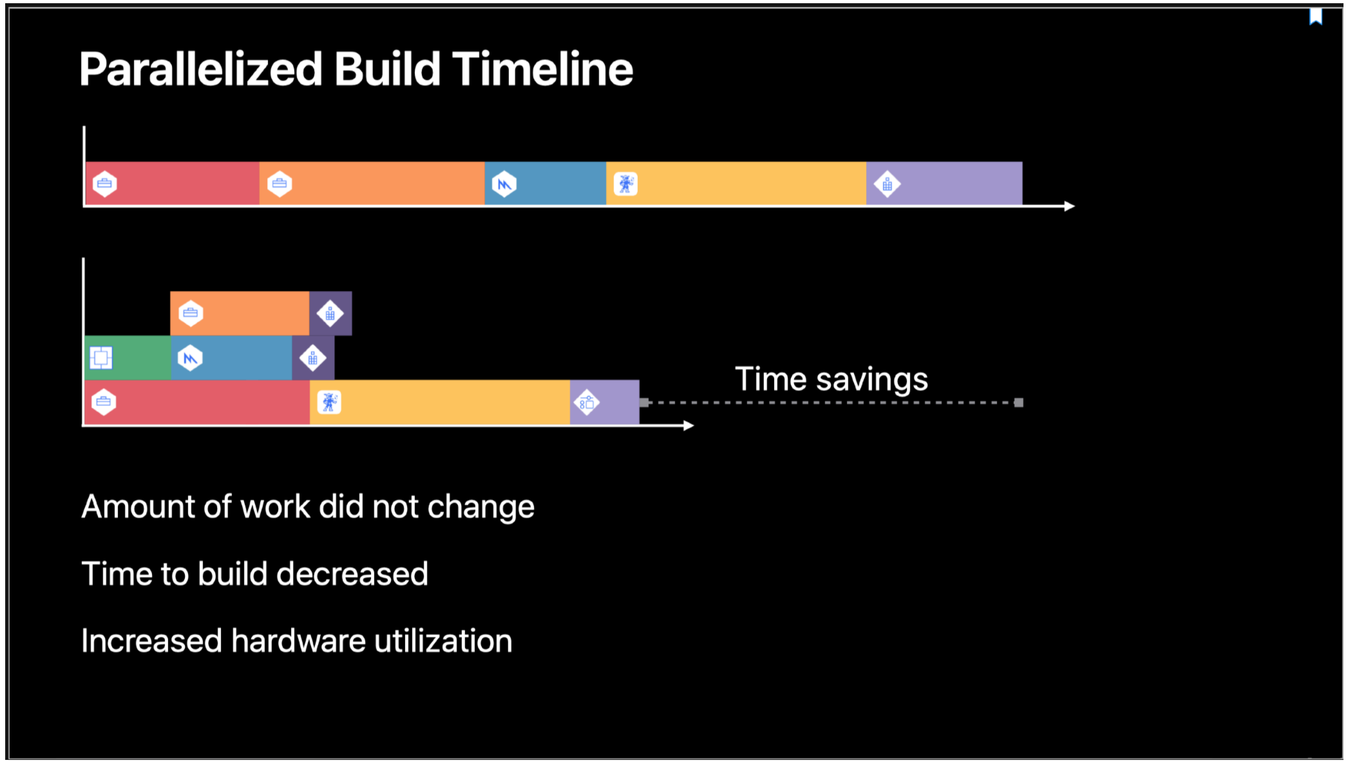
왜 모듈을 분리하나요?

# 1. 생산성 증대

- 빌드 속도 개선
- 무거운 모듈을 Mock (모조품) 으로 바꾸어 개발 가능
- 중고거래, 커뮤니티 등 기능별 샘플앱 활용 가능
    - 전체 프로젝트 빌드 X

왜 모듈을 분리하나요?

# 1. 생산성 증대




WWDC : Building Faster in Xcode

모듈을 분리하면

병렬 빌드의 장점을 살릴 수 있다

왜 모듈을 분리하나요?

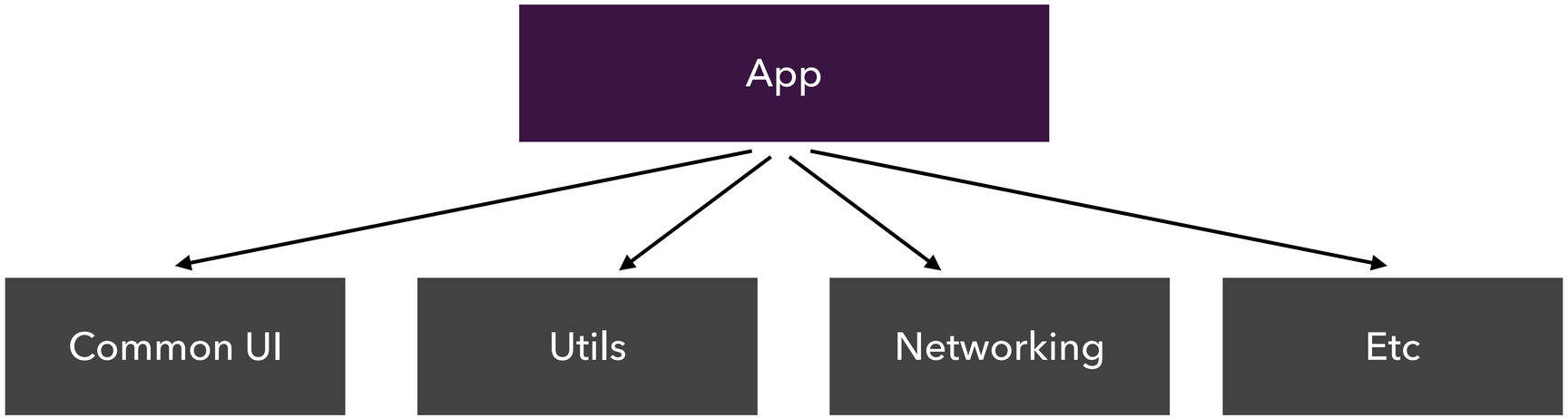
# 2. 복잡도 낮추기

- 프로젝트 복잡도를 낮춤
  - 동일한 코드를 줄이고, 객체의 응집도를 높임
- 모듈을 분리해 코드간 의존도를 낮춤
- 격리된 환경 (분리된 모듈)에서 독립적으로 개발 가능

# 모듈을 어떻게 분리하나요?

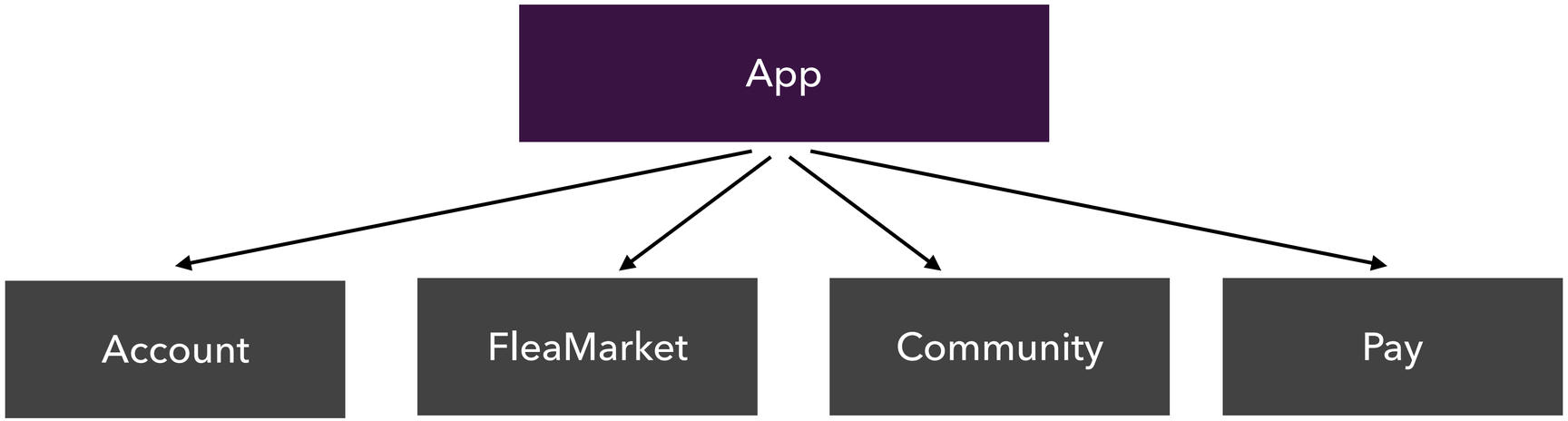
모듈을 어떻게 분리하나요

# Step 1. 재사용 관점에 따른 분리



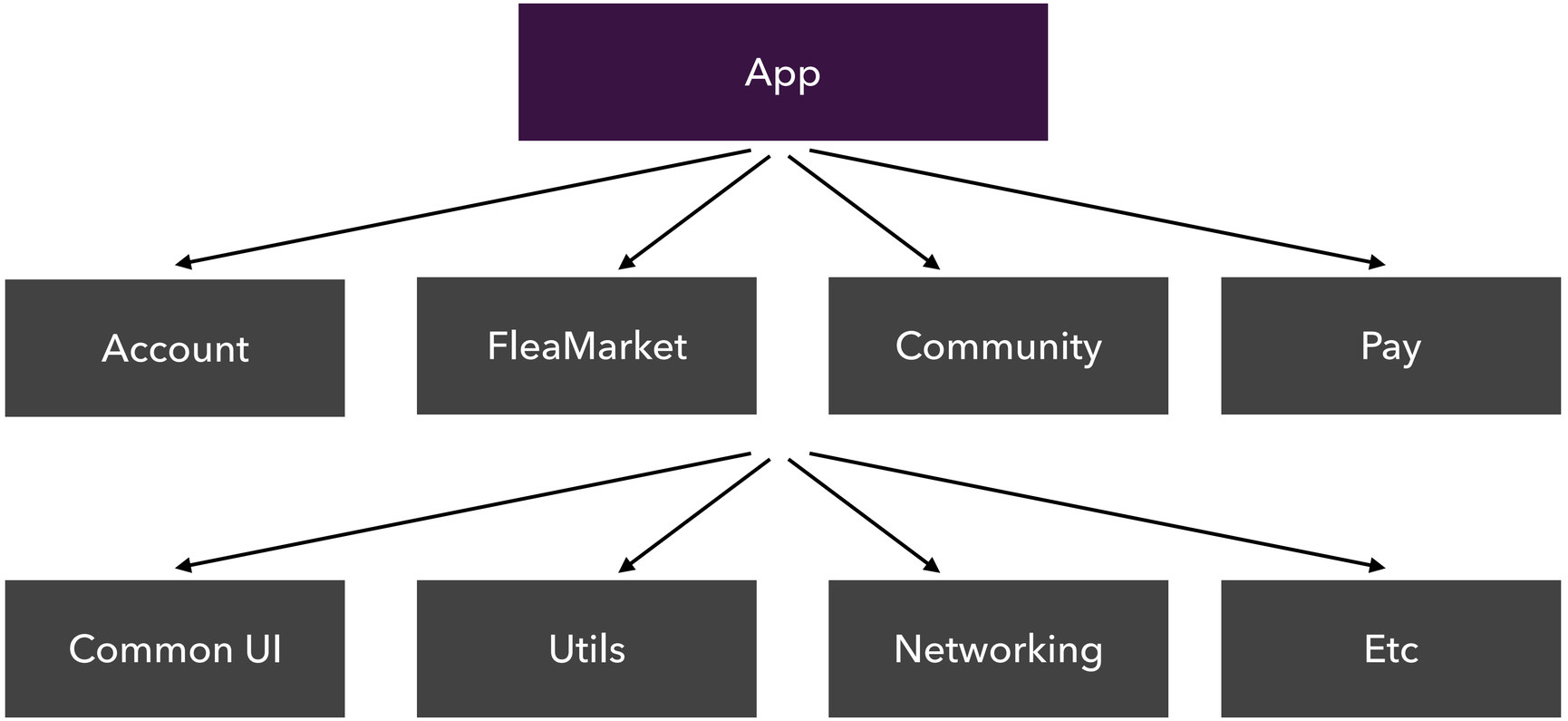
모듈을 어떻게 분리하나요

# Step 2. 도메인 관점에 따른 분리



모듈을 어떻게 분리하나요

# 합치면

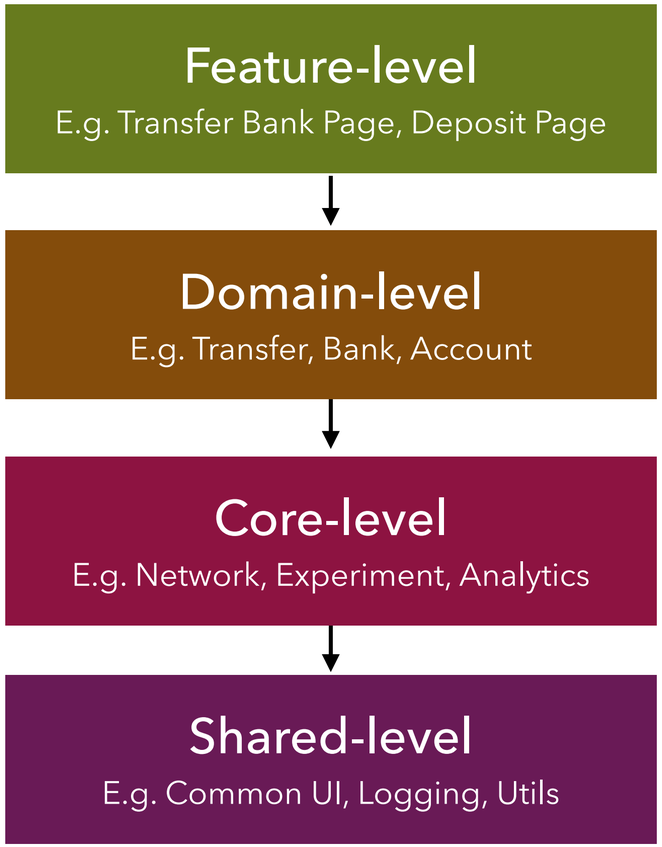


## 모듈 계층 구조

이를 계층으로 정리해보면

모듈을 어떻게 분리하나요

# 모듈 계층 구조

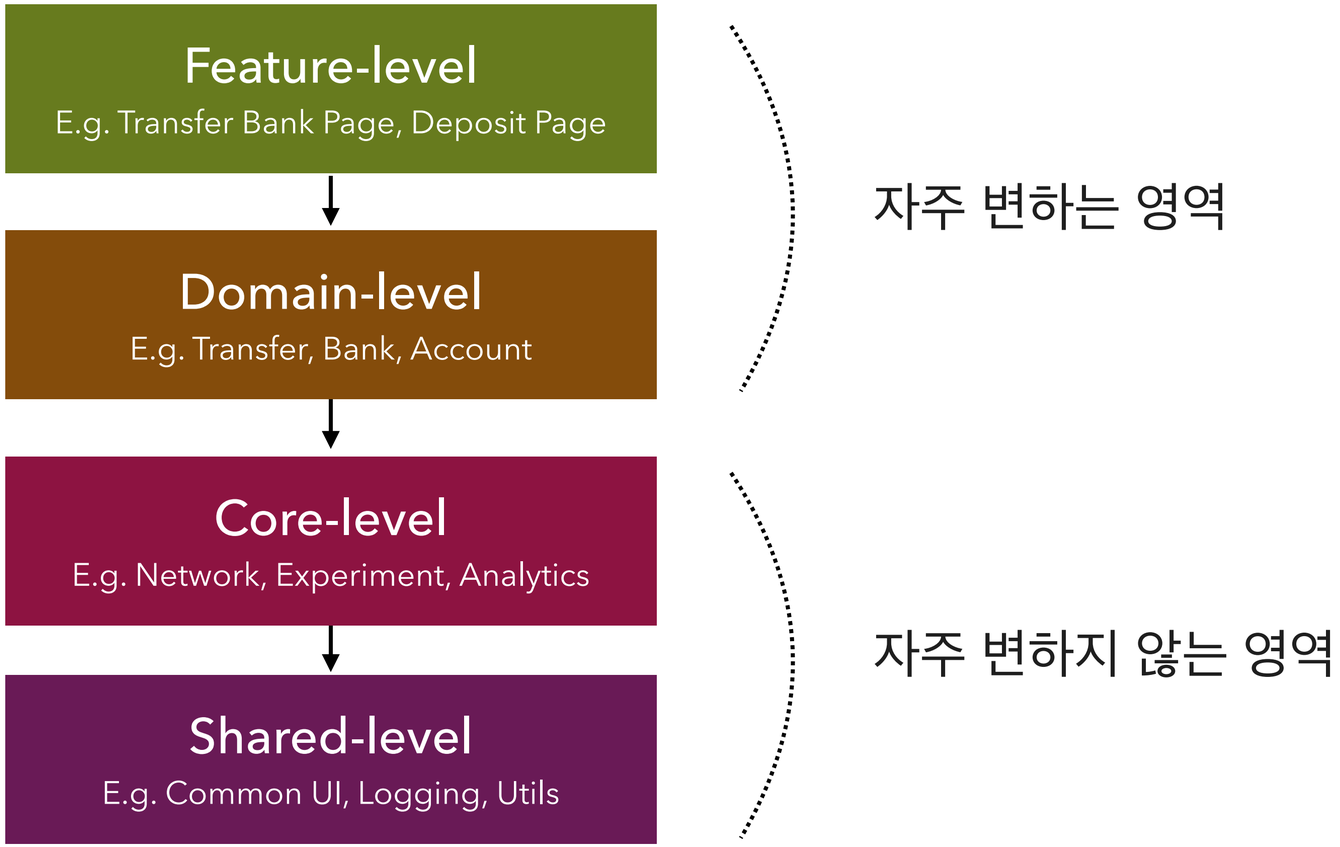



도메인 관점에 따른 분리

재사용 관점에 따른 분리

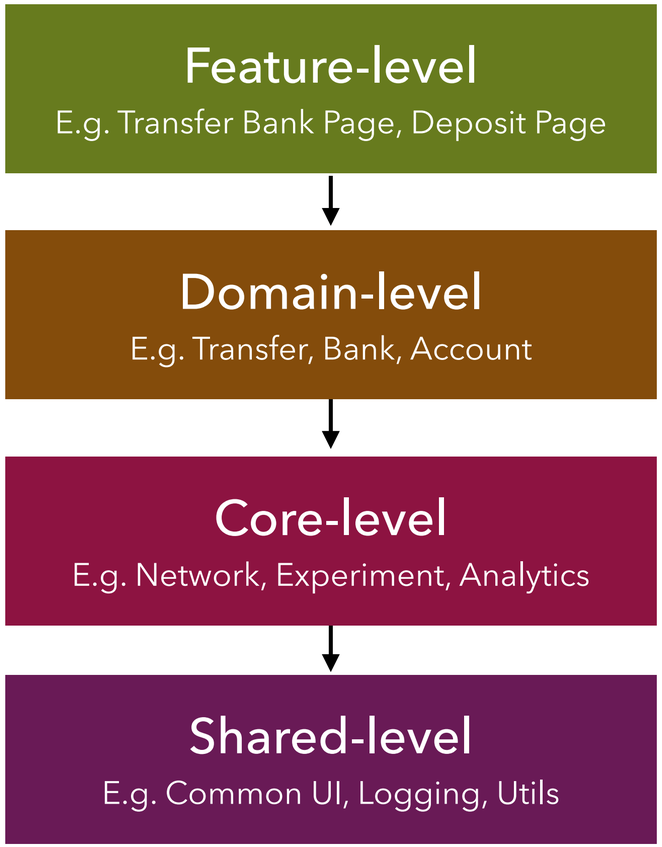
모듈을 어떻게 분리하나요

# 모듈 계층 구조



모듈을 어떻게 분리하나요

# 모듈 계층 구조

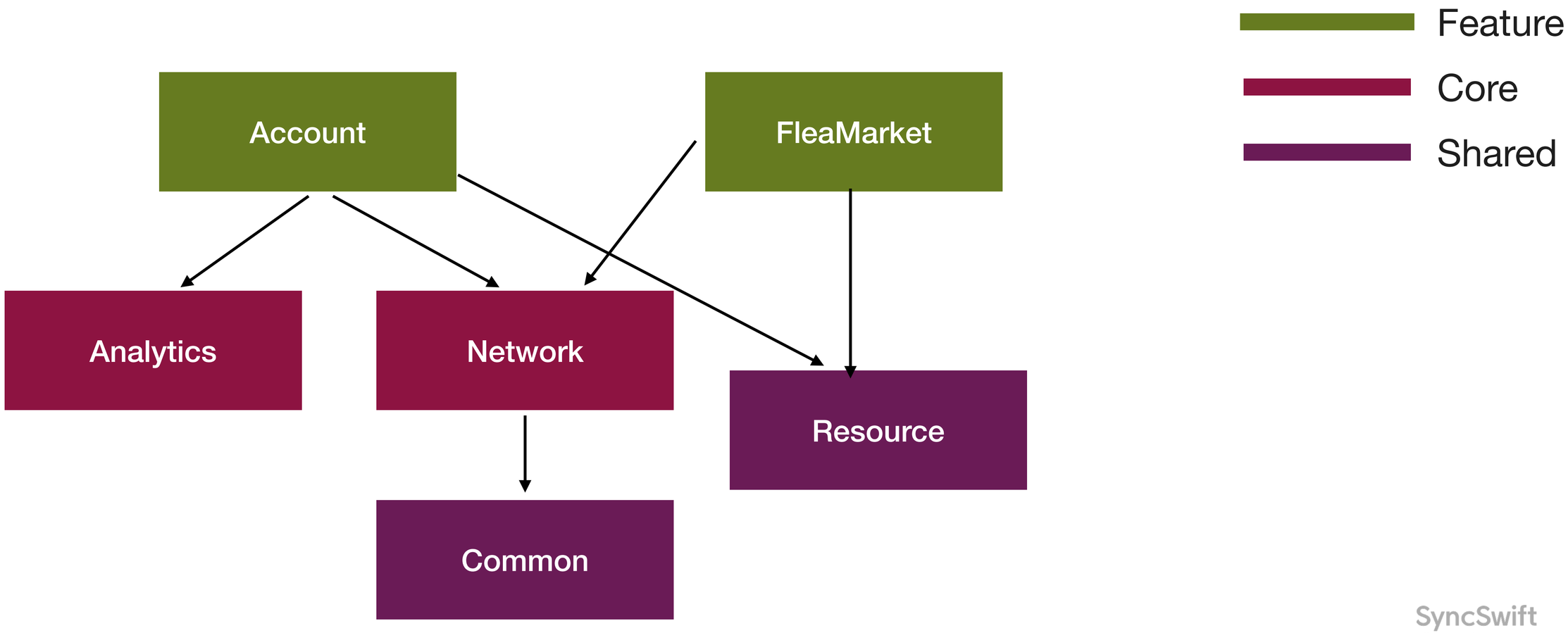



아래 계층의 모듈은

위 계층의 모듈을 알 수 없음

모듈을 어떻게 분리하나요

# 모듈 구조 예시



- 실전편 -

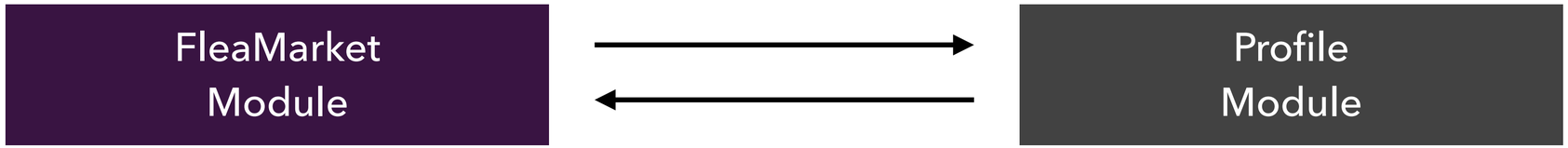
# 모듈화 시 만나는 문제들

## 유의사항

설명에 사용된 예시는
실제 구성과 차이가 있음

모듈화 시 만나는 문제들

# 순환 참조 문제

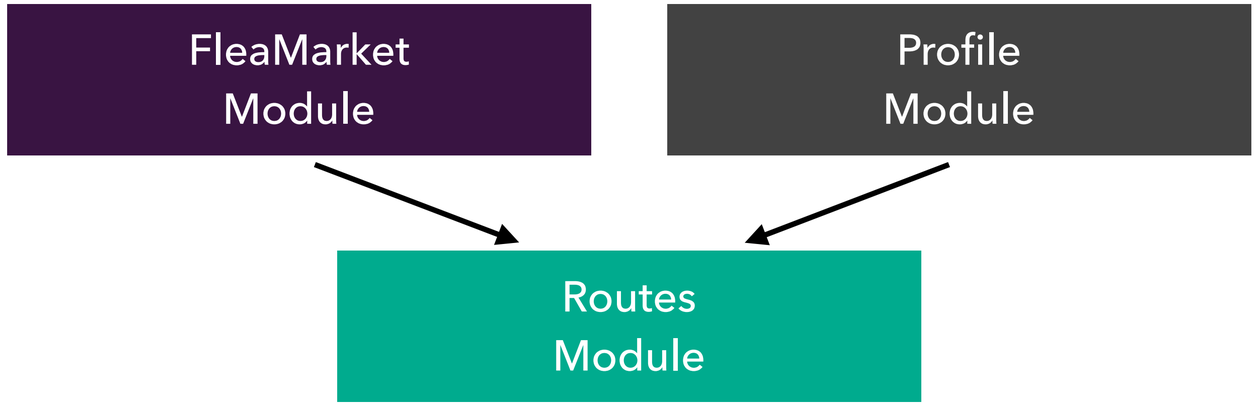



예시) 각 Feature 모듈간의 라우팅이 필요한 상황

- 모듈은 서로를 참조할 수 없음
- 모듈 분리를 시작하면 많은 양의 순환 참조를 마주하게 됨

모듈화 시 만나는 문제들

# 순환 참조 문제

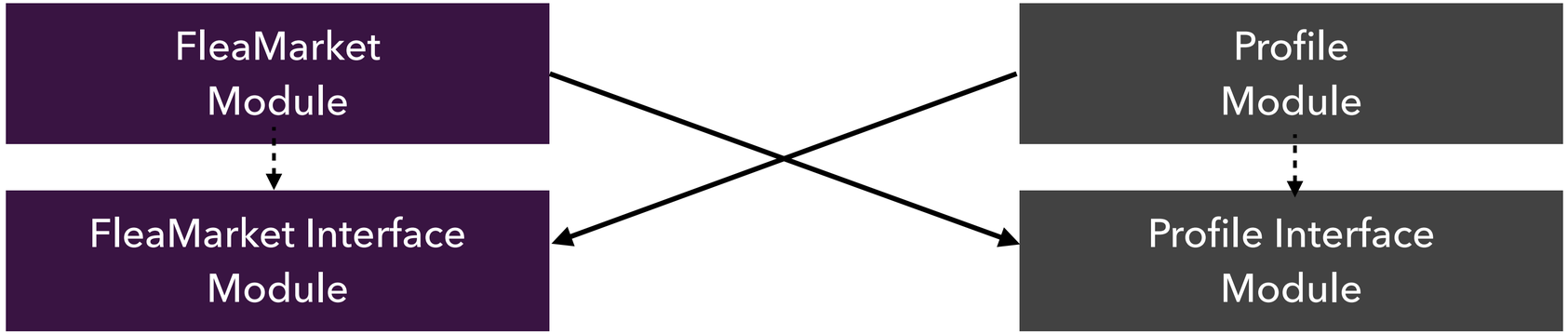



앱 전체의 라우팅을 담당하는 모듈을 만들어서 해결이 가능하지만

이는 라우팅 모듈에 대한 의존을 강하게 만들 수 있음

모듈화 시 만나는 문제들

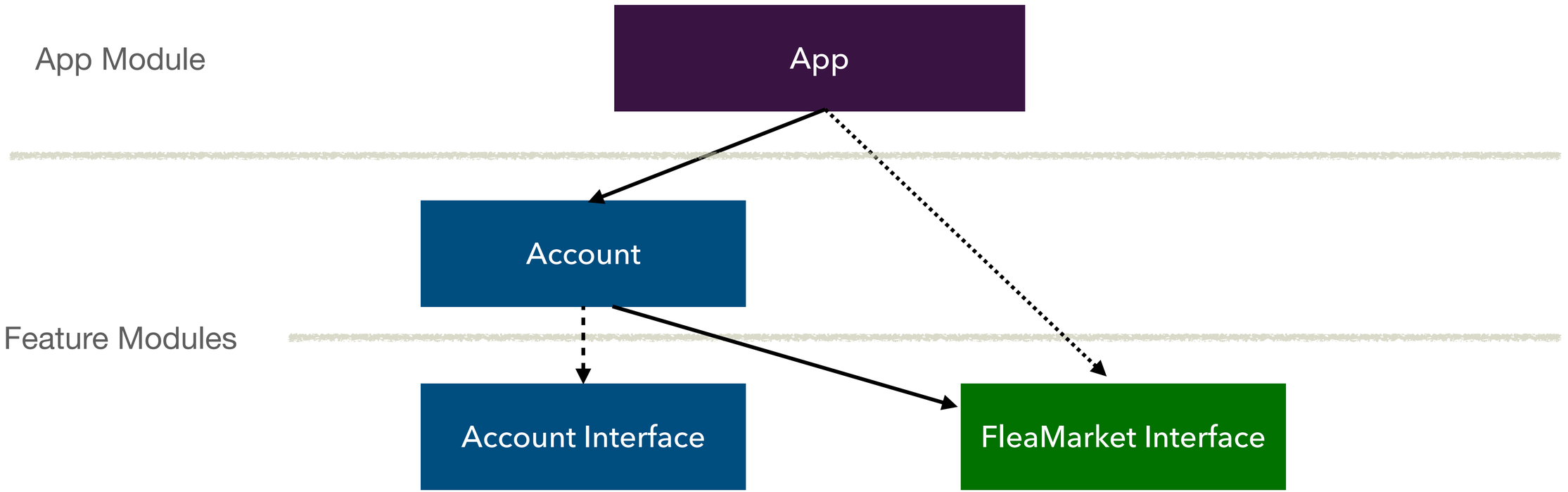
# 순환 참조 문제




- 모듈을 인터페이스 모듈과 구현 모듈로 분리한다
- 각 규현 모듈은 서로의 인터페이스 모듈을 참조한다

모듈화 시 만나는 문제들

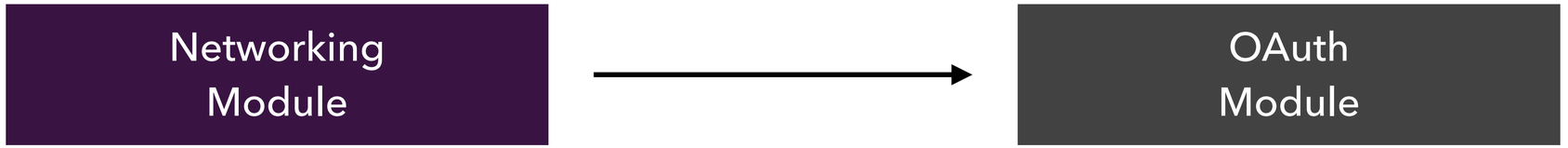
# 인터페이스 모듈 활용하기




1. App 에 있는 구현체는 남기고 FleaMarket Interface 모듈만 먼저 분리
2. FleaMarket Interface 모듈을 이용해 Account 모듈을 분리

모듈화 시 만나는 문제들

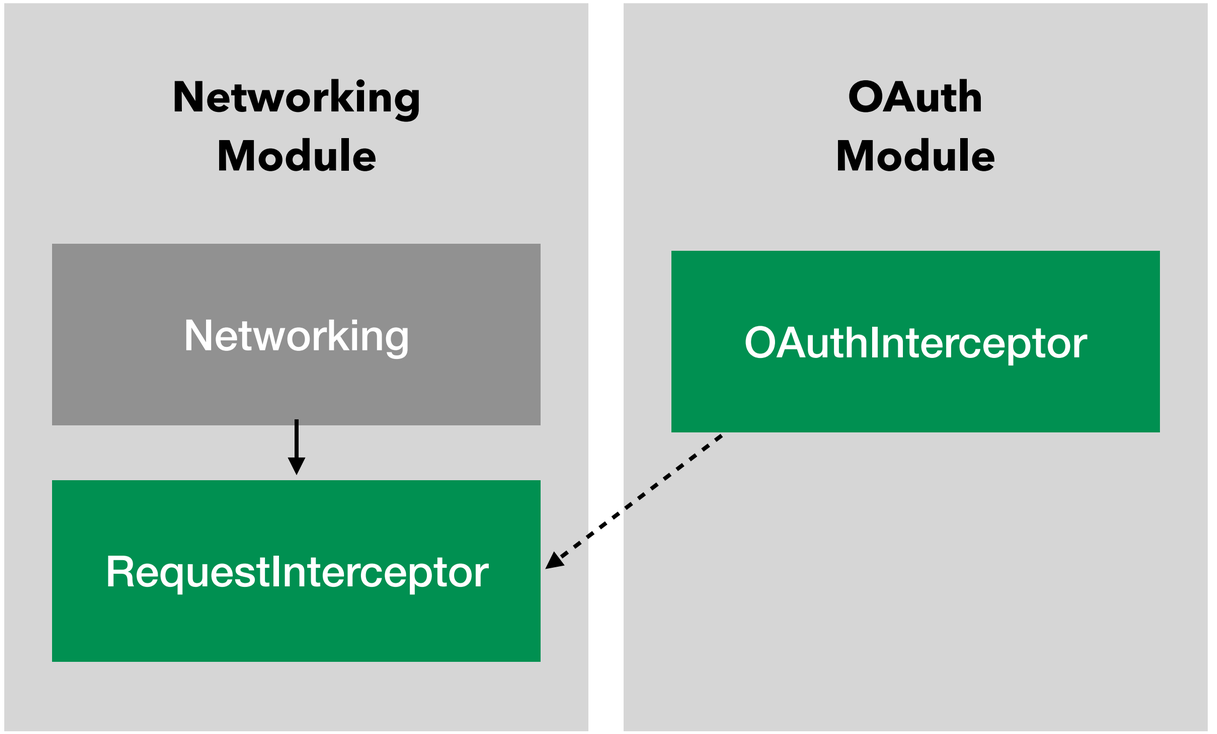
# 의존성 방향 문제




- Networking 모듈이 OAuth 모듈에 의존
- 추상화된 모듈이 구현 모듈에 의존
- 모듈의 관계를 직관적으로 이해하기 어렵다

모듈화 시 만나는 문제들

# 의존성 방향 문제

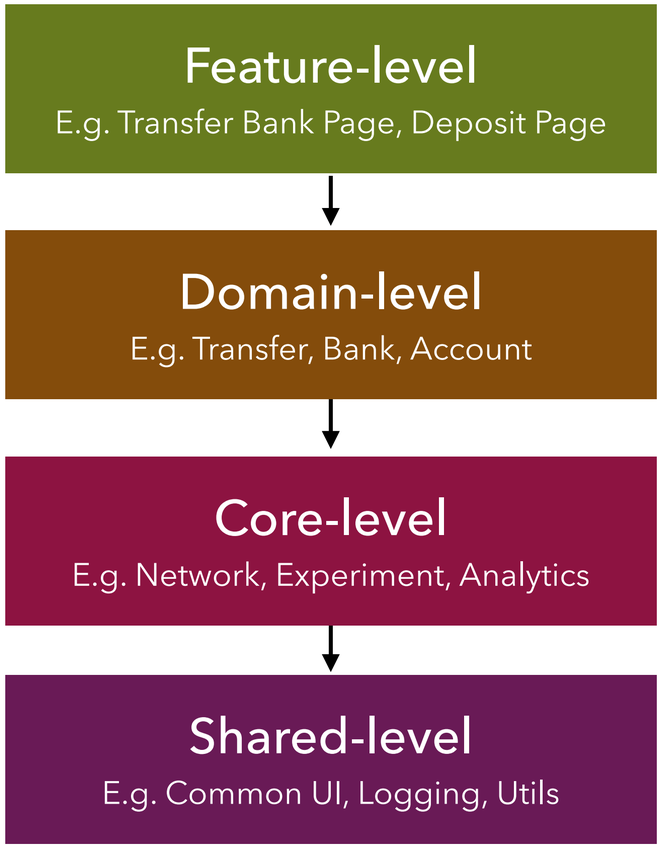



- 의존성을 역전시켜 문제를 해결

# 다시 돌아와서

모듈을 어떻게 분리하나요

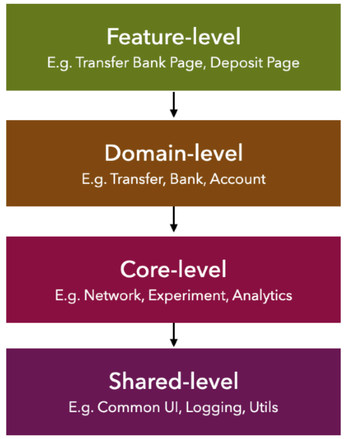
# 다시 돌아와서




오늘은 두 계층에 집중

모듈을 어떻게 분리하나요

# Shared Level




**Shared-level**
E.g. Common UI, Logging, Utils

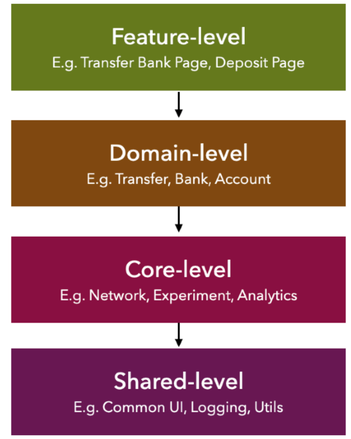
Sources Array+Safe.swift

Tests Array+SafeTests.swift

- 모든 계층에서 사용 가능한 모듈
- 의존성 없이 단독으로 사용 가능한 코드

모듈을 어떻게 분리하나요

# Core Level




**Core-level**
E.g. Network, Experiment, Analytics

- Interface — Networking.swift
- Sources — NetworkingImpl.swift
- Testing — NetworkingMock.swift
- Tests — NetworkingTests.swift

- 인터페이스 모듈과 구현체 모듈을 구분
- Test Double 을 구성한 Testing 모듈이 존재

# 마무리

마무리

## 모듈 분리 순서

1. 대상 코드 선정
2. 대상 코드의 의존성 문제를 해결
3. 모듈을 인터페이스 / 구현 모듈 페어로 만들어야 하는지 결정
4. 의존성 주입 구조 만들기
5. 신뢰 가능할 정도의 유닛 테스트 추가
6. 모듈을 만들고 코드를 이전

마무리

## 언제 모듈화를 시작하나요?

지금부터

마무리

# 지금 모듈화를 시작하면 좋은 이유

- 공통 로직부터 모듈 분리를 시작하자 (Shared, Core 계층)
- 모듈을 분리하는데 리소스가 많이 들지 않는다
- 모듈화는 시작이 늦을수록 고통스럽다
- 모듈 계층 구조를 통해 아키텍처 문제를 사전에 발견 가능하다
- 코드간 의존도를 낮춰 복잡도를 낮출 수 있다
- 테스트하기 좋은 구조를 만들 수 있다

마무리

## 언제 모듈화를 시작하나요?

그러나,

학습 없이 시작하자는 것은 아닙니다

마무리

# 추가로 학습해야할 것들

- Xcode 에서 빌드가 실행될 때 어떤 일이 일어나는지 알아보기
- Swift Package, Dynamic Framework, Static Library 차이를 알기
- 의존성 역전, 의존성 주입 이해하기
- 유닛 테스트 작성하기

마무리

# 추가로 학습해야할 것들

Xcode 를 이용한 기본 구현을 모두 경험한 뒤
필요에 따라 여러 도구를 사용해보는 것을 추천합니다

- Tuist / XcodeGen
- Swinject
- Needle
- Mockolo
- Etc..

# 감사합니다

SyncSwift
Conference
2022