Building Business Layer with CLEAN Architecture

Kaushal Dhruw
APP DEVELOPER / AUTHOR

@drulabs linkedin.com/in/kaushal-dhruw/

Overview

CLEAN Architecture concepts

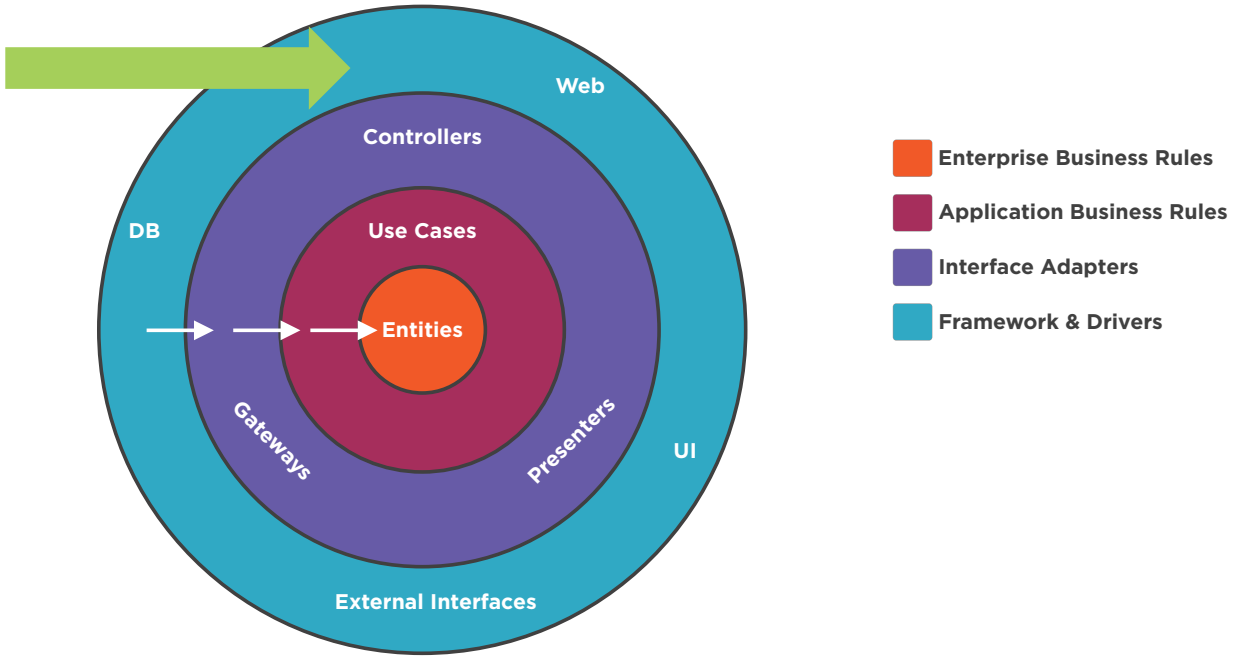
The dependency rule

Basics of RxJava

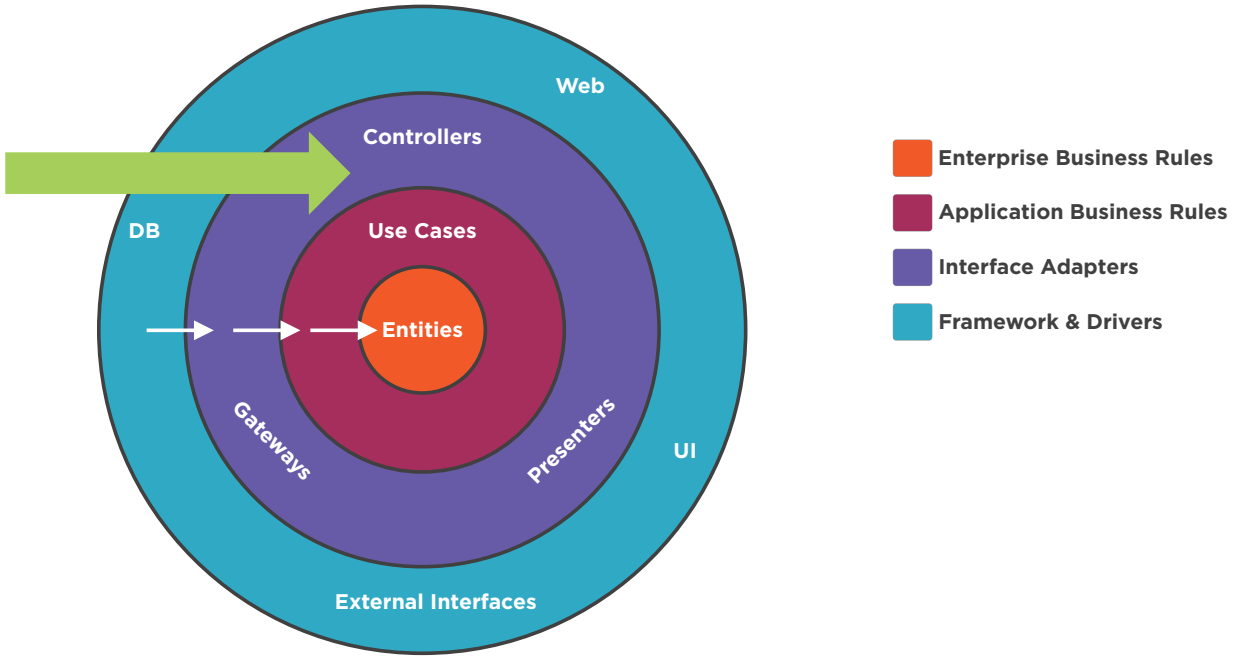
Domain layer for Banking app

Testing the domain layer

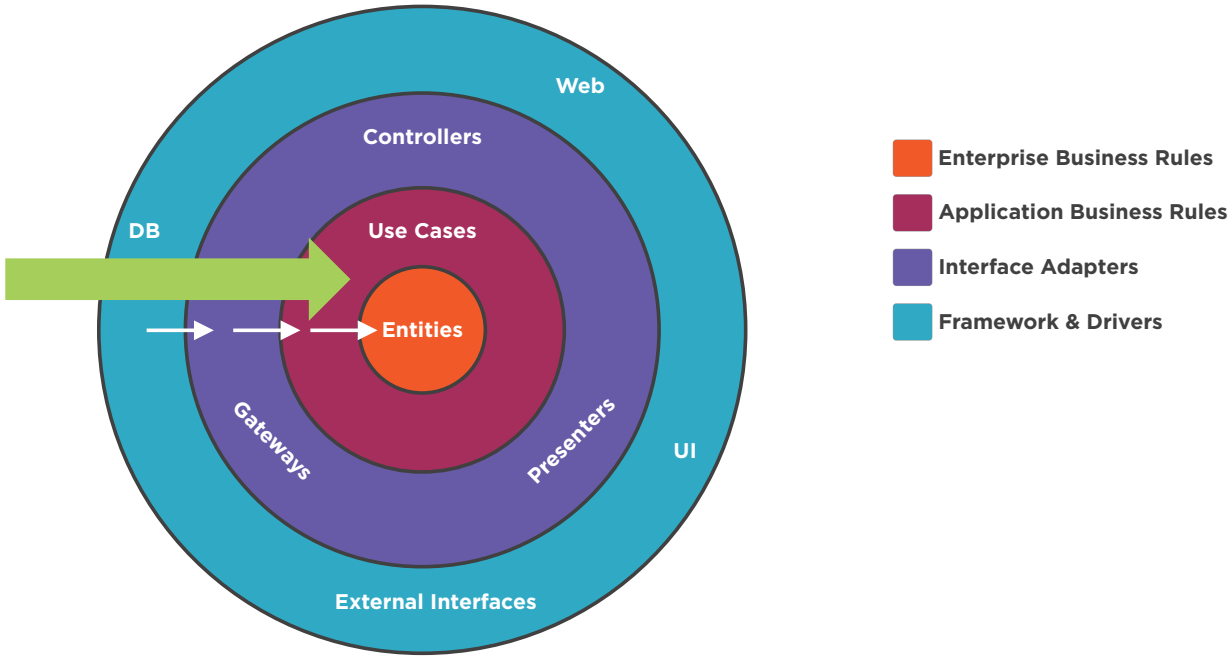
CLEAN Architecture by Robert C. Martin



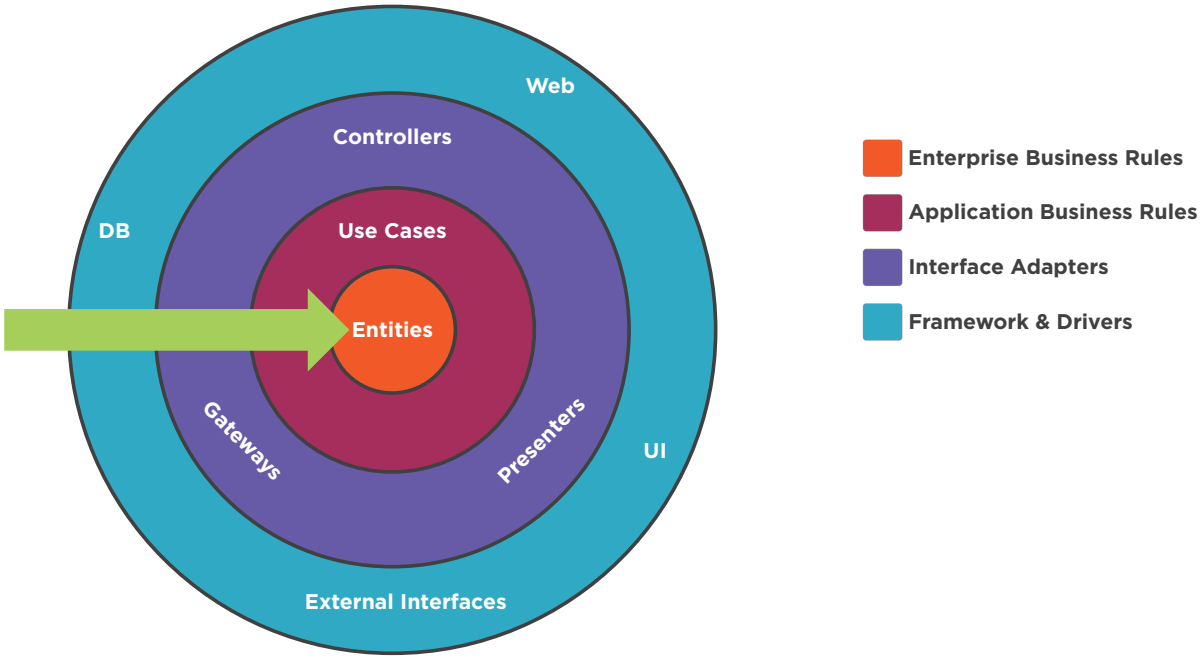
CLEAN Architecture by Robert C. Martin



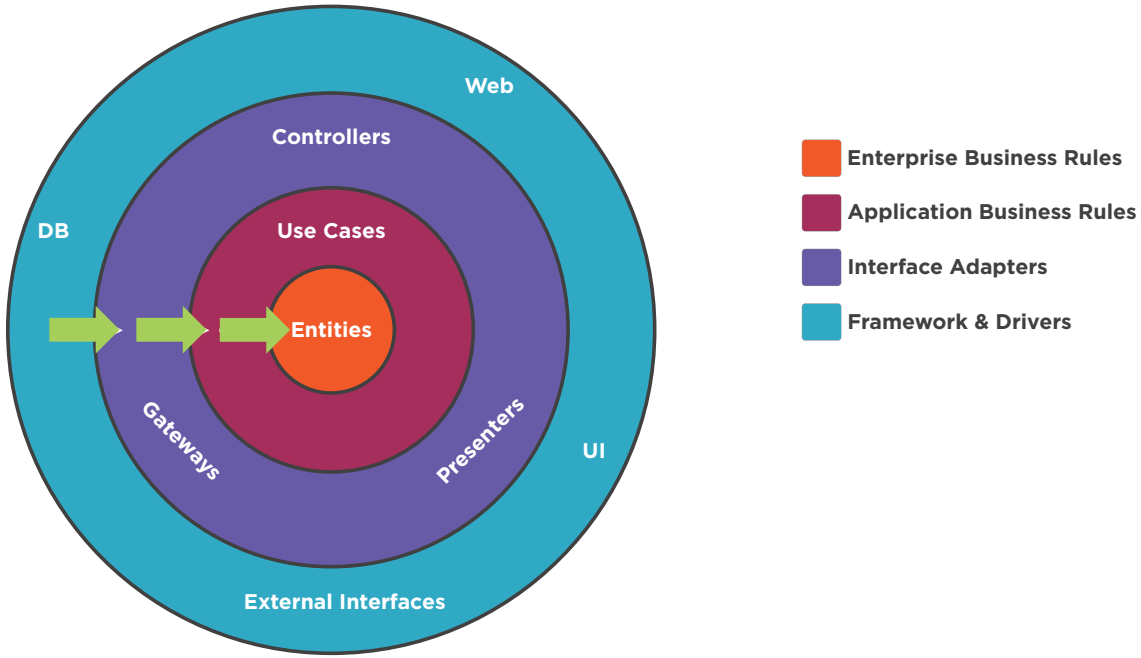
CLEAN Architecture by Robert C. Martin

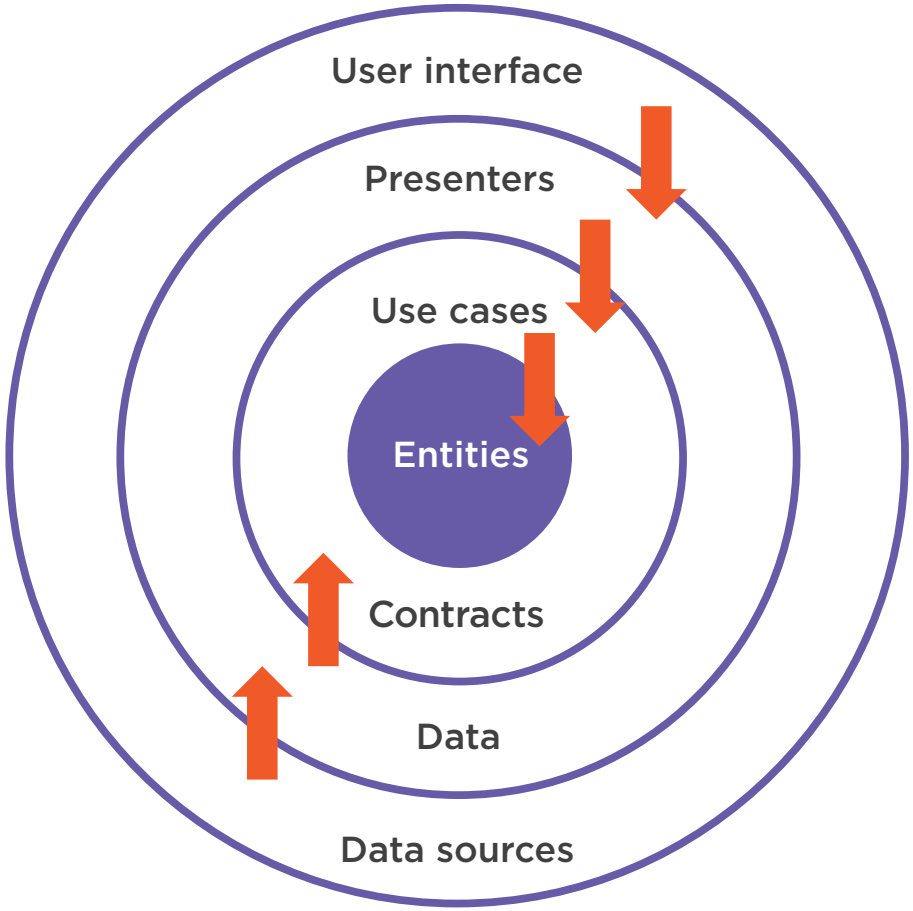


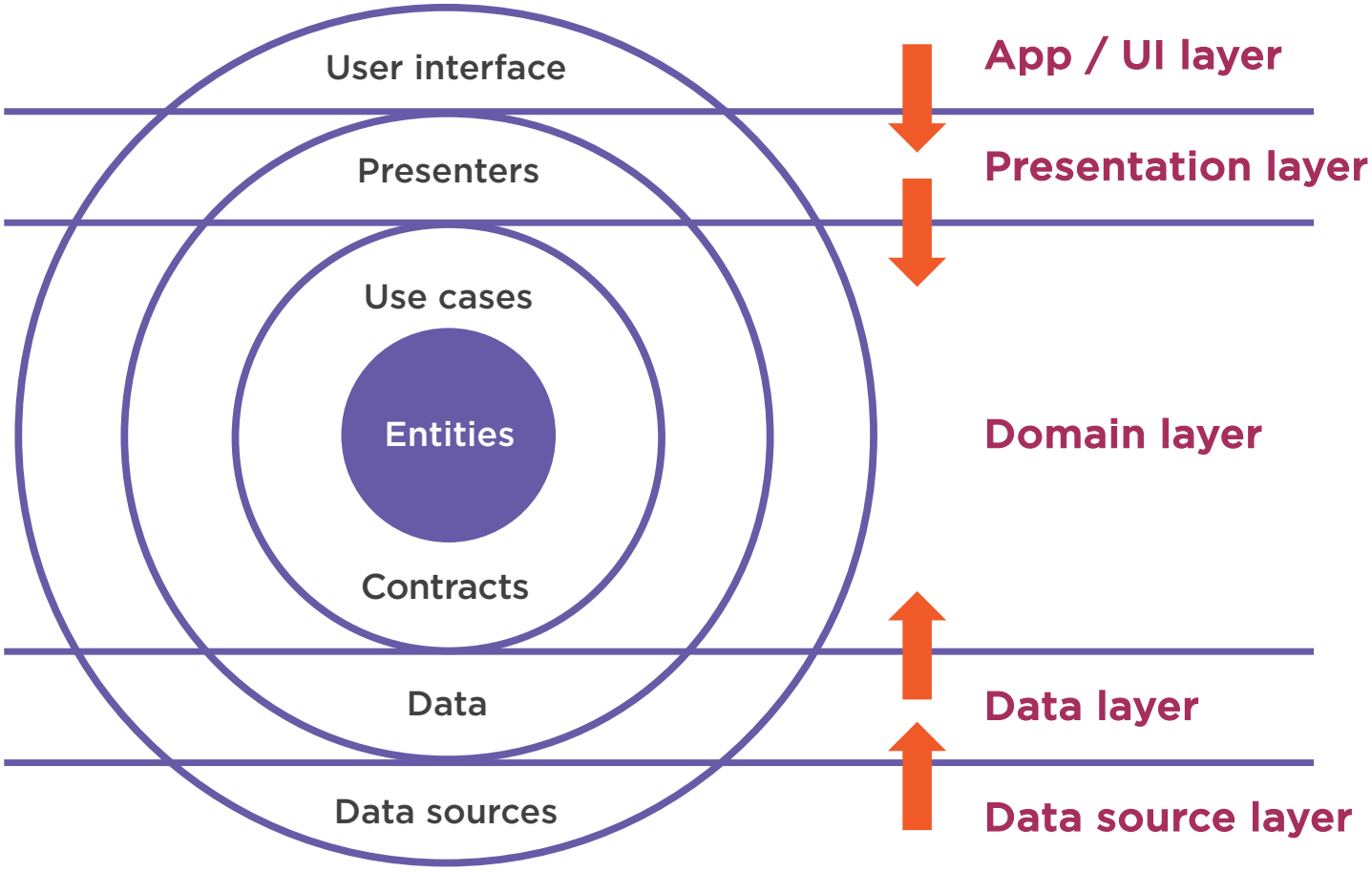
CLEAN Architecture by Robert C. Martin



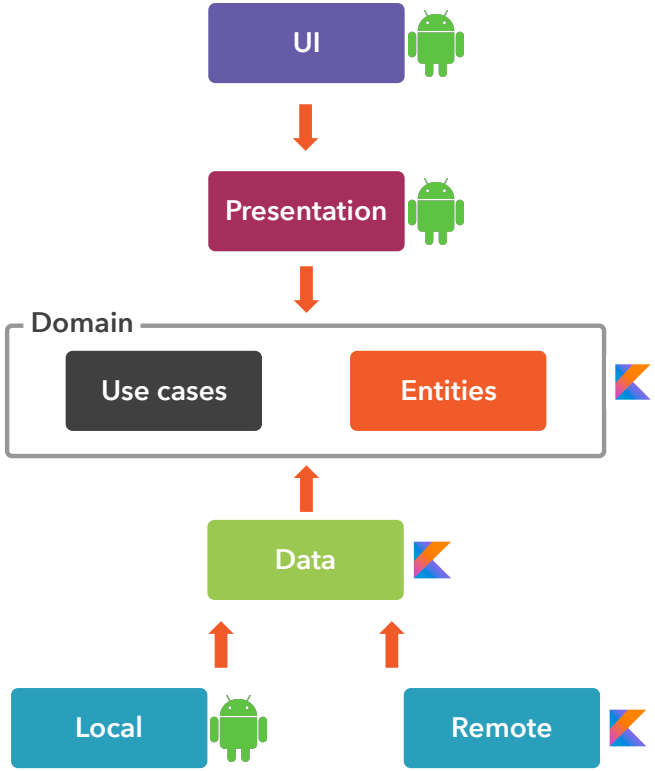
CLEAN Architecture by Robert C. Martin




User interface
Presenters
Use cases
Entities
Contracts
Data
Data sources


User interface
Presenters
Use cases
Entities
Contracts
Data
Data sources
App / UI layer
Presentation layer
Domain layer
Data layer
Data source layer

CLEAN Architecture in Android



The Dependency Rule

The SOLID Principles

Single Responsibility

Open-Closed

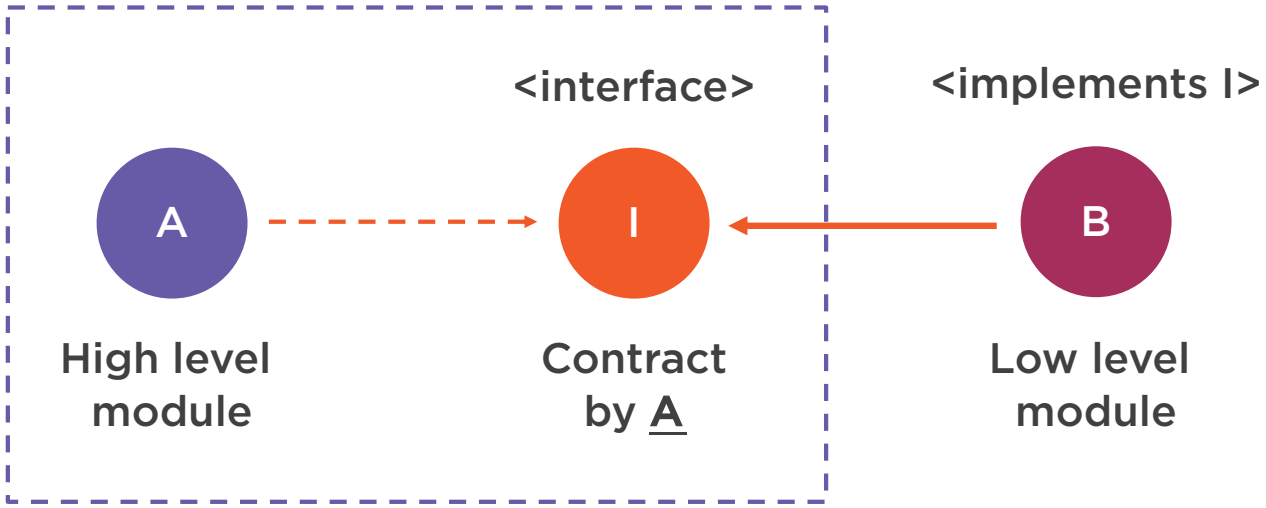
Liskov Substitution

Interface Segregation

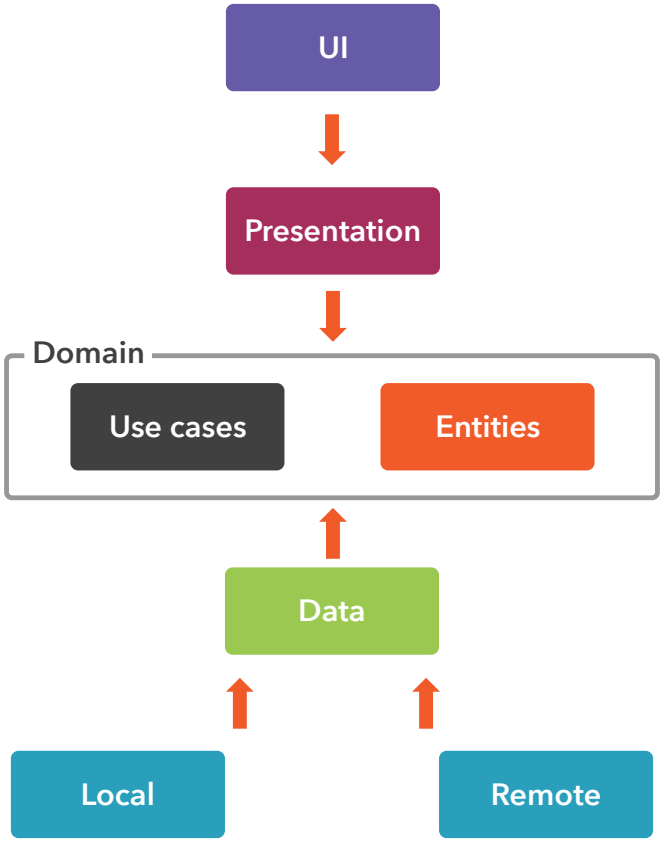
Dependency Inversion

Dependency Inversion Principle

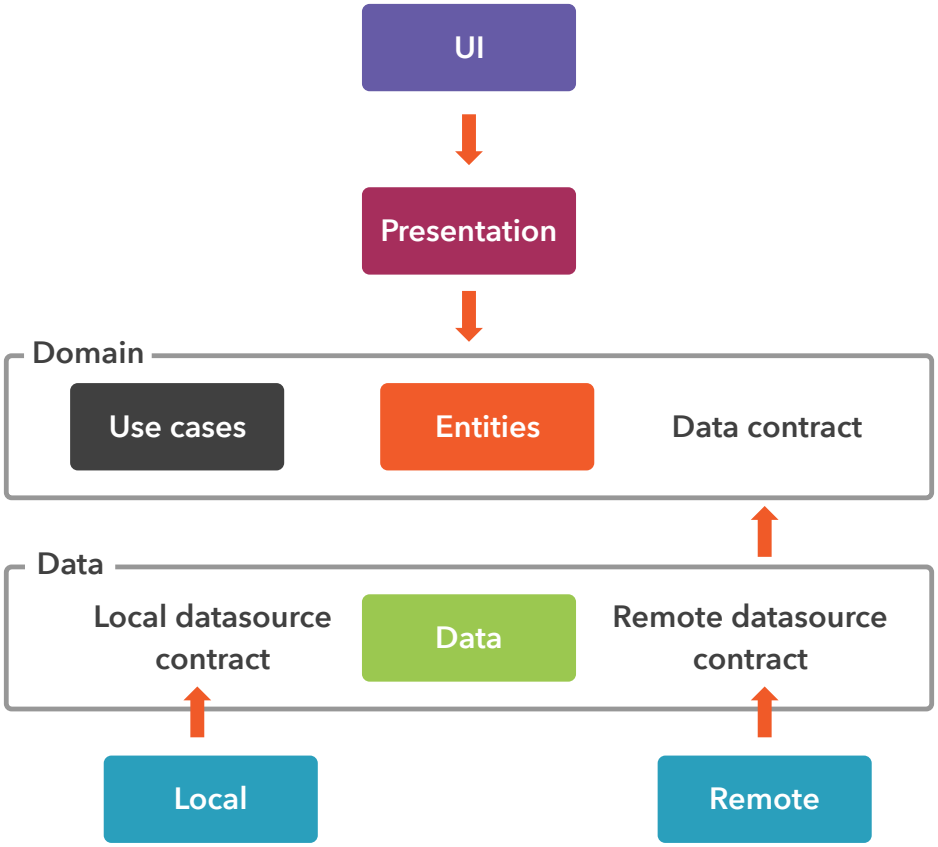
Dependency Inversion Principle



Dependency Inversion in CLEAN Architecture



Dependency Inversion in CLEAN Architecture



Dependency Rule

Dependencies can only point inwards

Inner layers are rules and policies

Outer layers are mechanisms and tools

Inner layers are oblivious to outer layers

Dependencies must point towards stability and abstraction

A Brief Introduction to RxJava

Reactive Programming with RxJava

Observer pattern

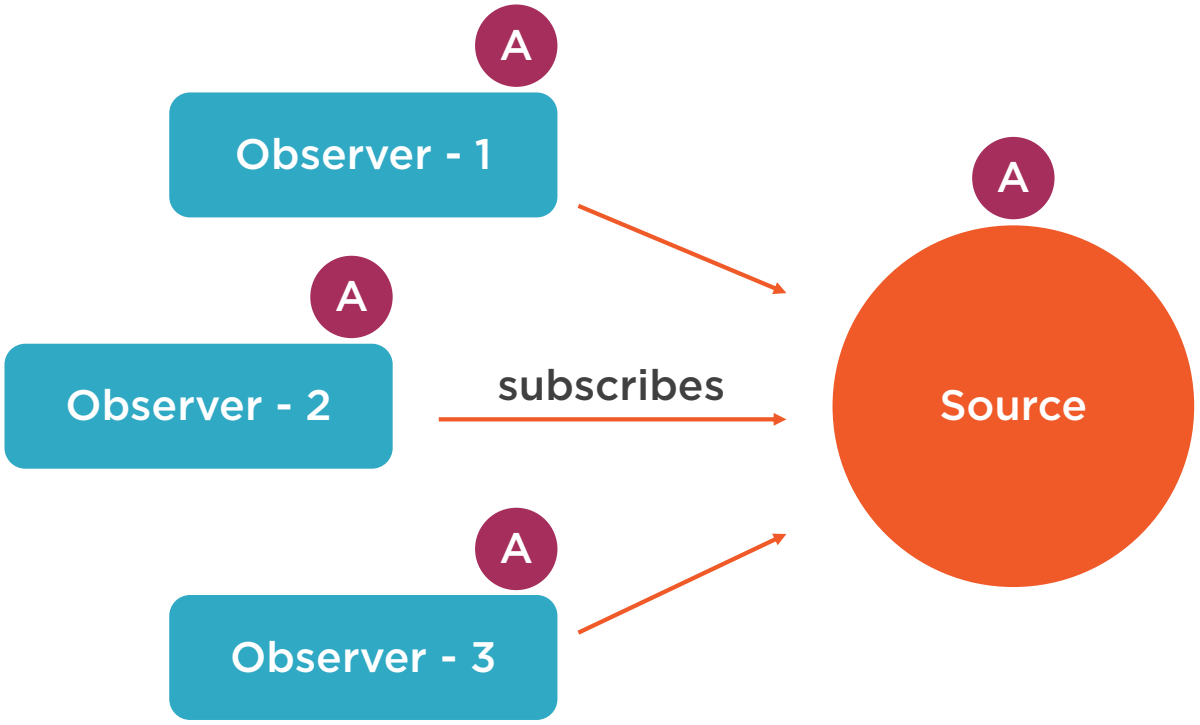
Iterator pattern

Resilience

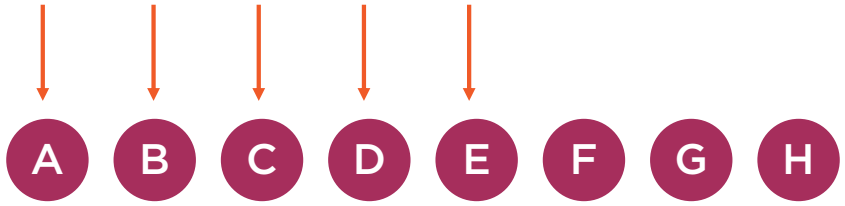
Operators

Scheduling

Observer Pattern



Iterator Pattern



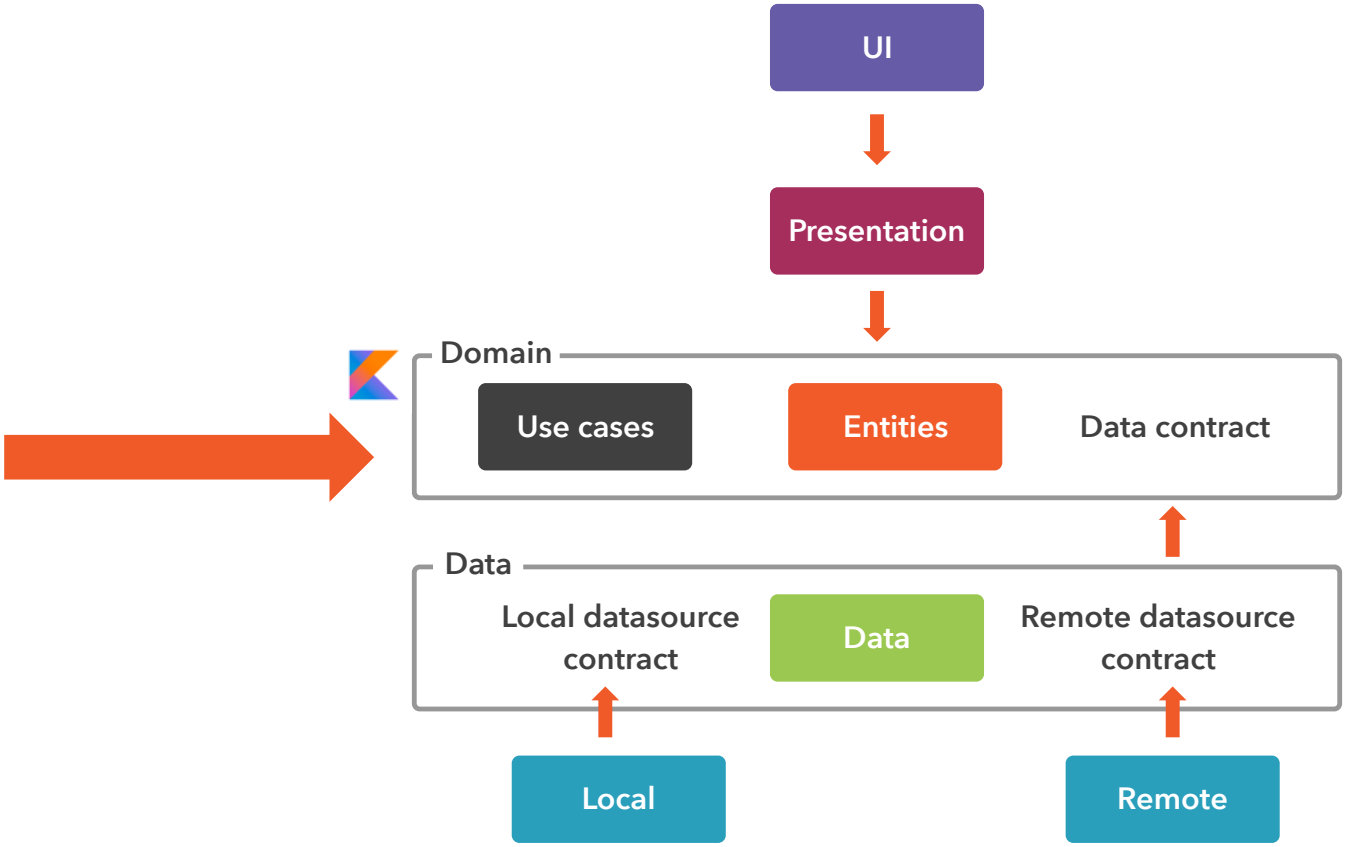
RxJava Observable

RxJava Observable Example

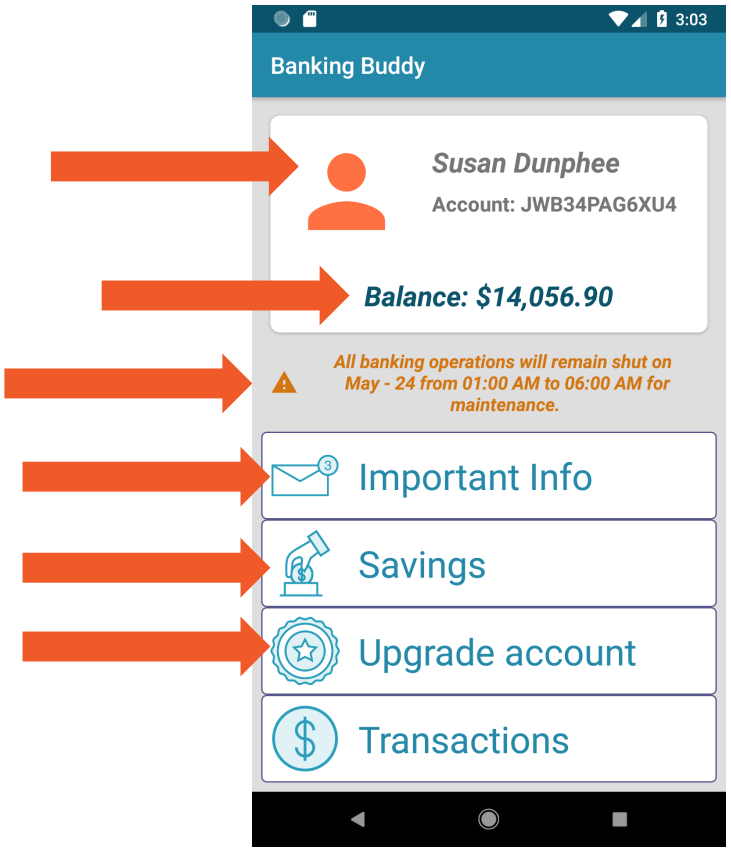
```
val userInfo: Observable<UserInfo> = getInfoFromRemote()
userInfo.subscribeOn(Schedulers.io())
    .observeOn(AndroidSchedulers.mainThread())
    .map { data -> transformData(data) }
    .doOnNext { newData -> updateUI(newData) }
    .doOnError { err -> handleError(err) }
    .subscribe(…)
```

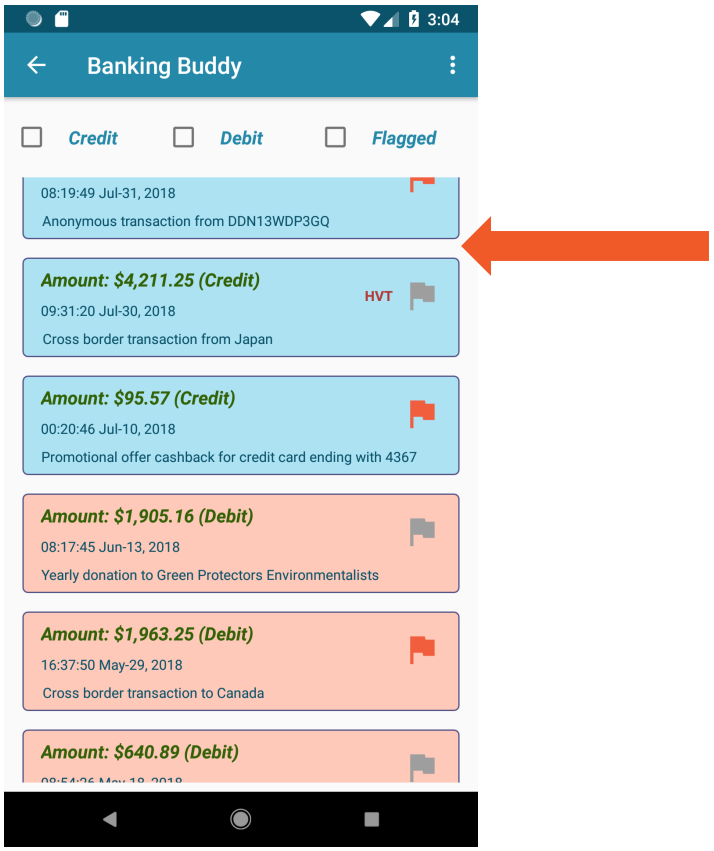
Building the Domain Layer

CLEAN Architecture



Banking App Screens





Banking App Use Cases

Get user information

Get list of transactions

Filter transactions

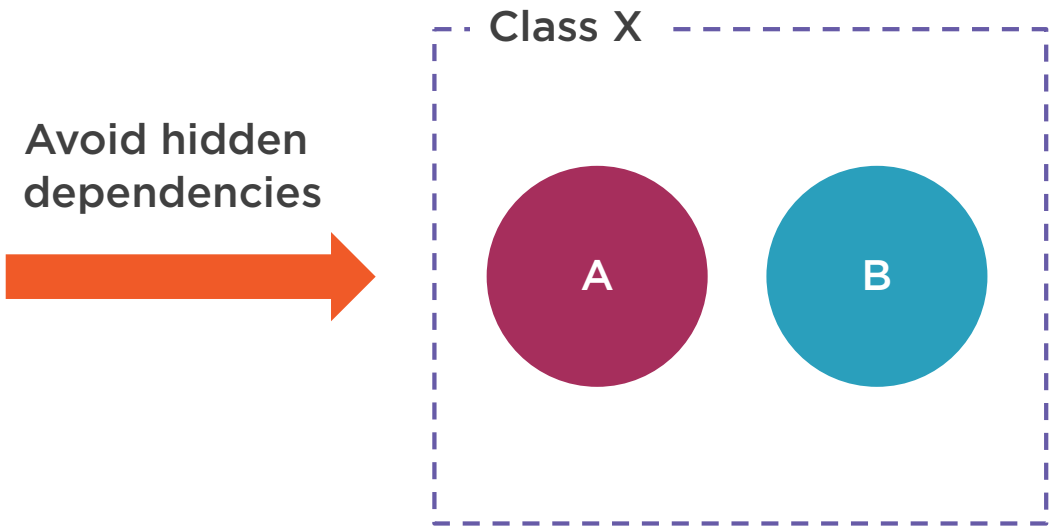
Change transaction flagged status

Demo

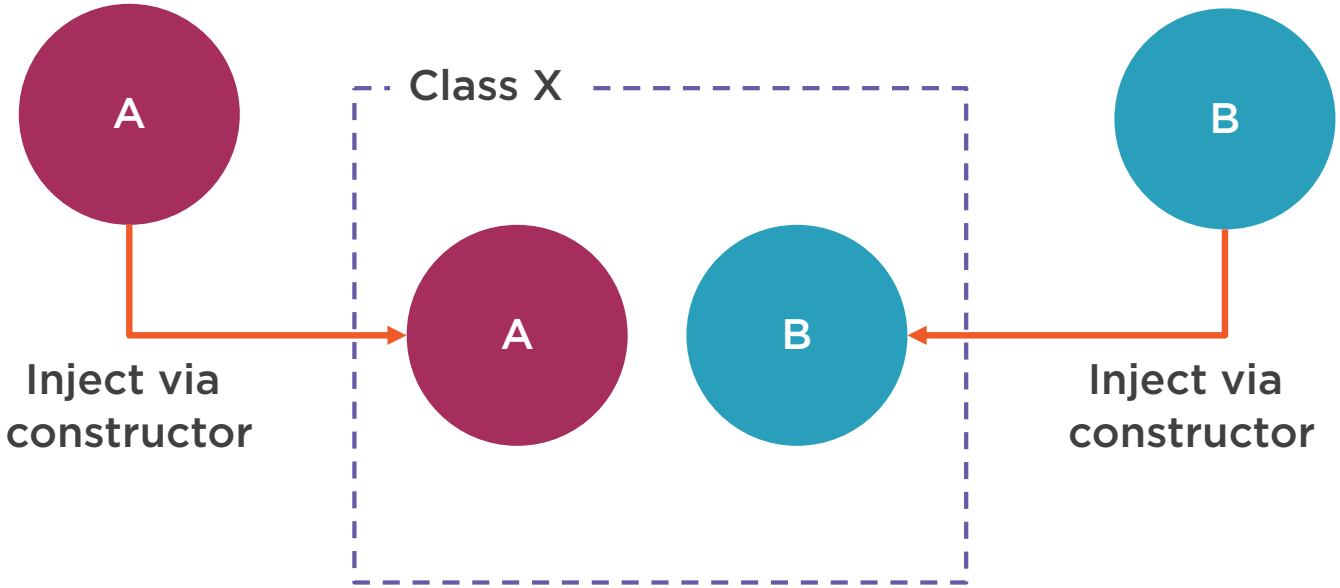
- Project structure in Android studio
- Business rules and use cases
- Use cases of the Banking app
- Contract for the data layer

Testing the Domain Layer

Software Testing

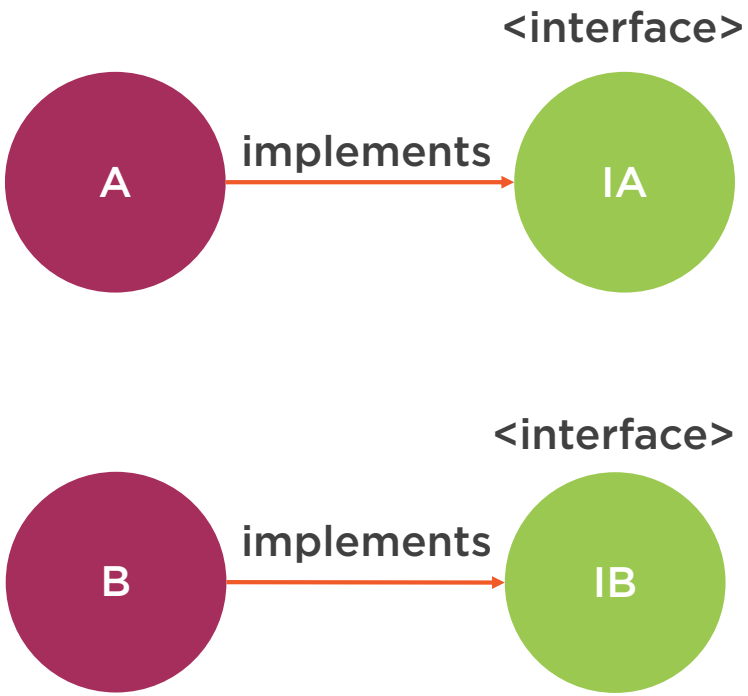


Software Testing

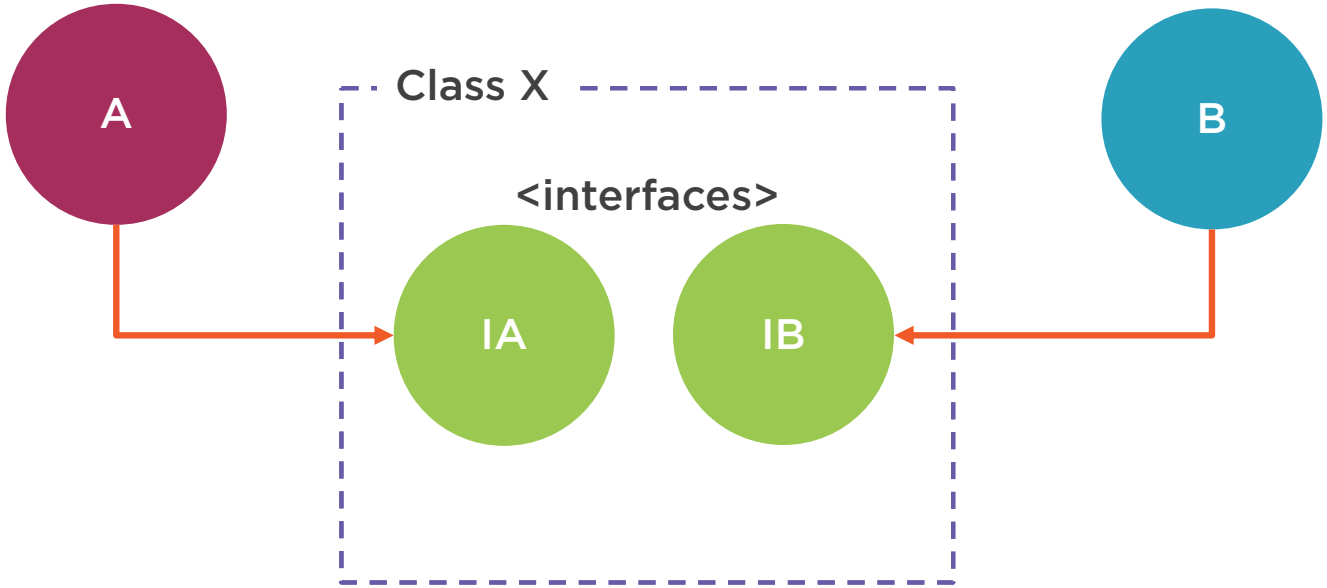


Better but... class X is still tightly coupled with A and B

Software Testing

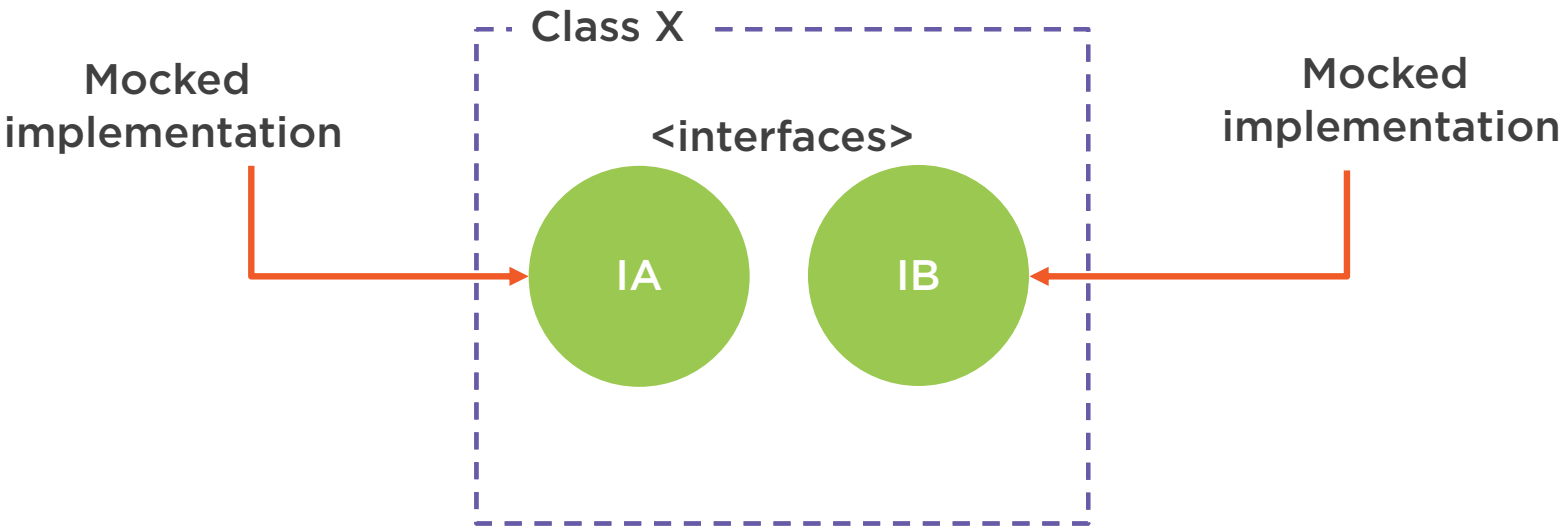


Software Testing



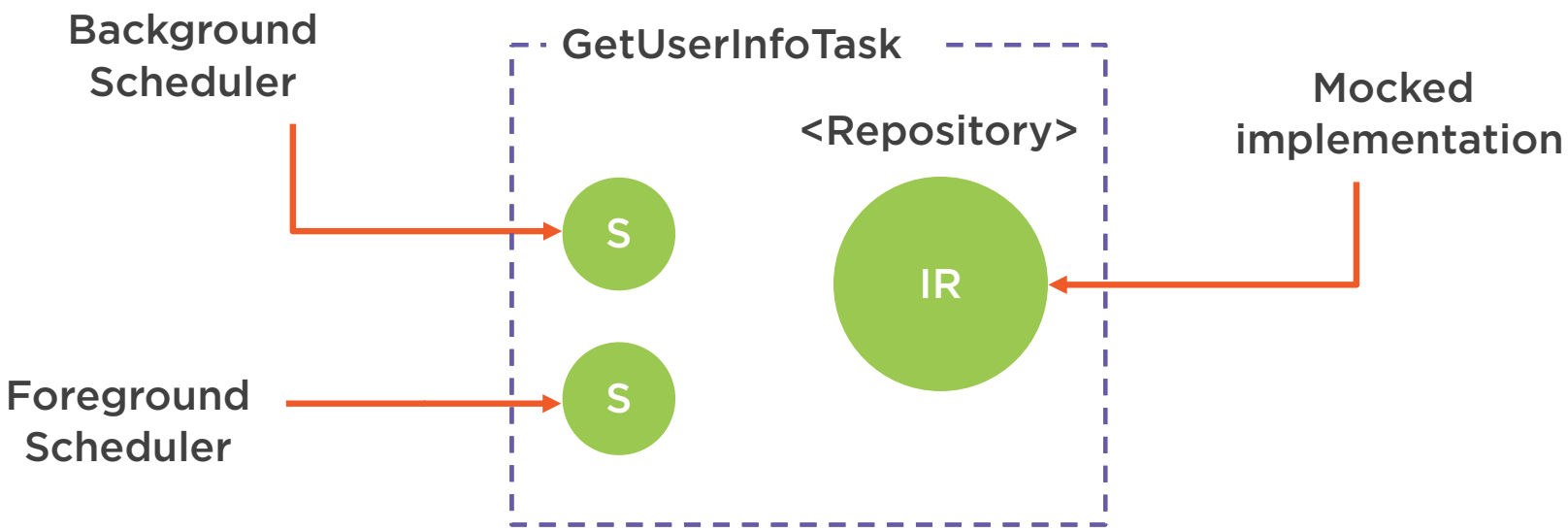
Low coupling

Software Testing



Easy testing

GetUserInfoTask Use Case



Demo

Write a JUnit test case class

Test all use cases

Write a failing test

Summary

CLEAN Architecture Basics

The Dependency Rule

A Brief Introduction to RxJava

Building the Domain Layer

Testing the Domain Layer