CleanArchitecture

카카오톡 #탭 적용기

#탭 개발자 Latte

Developer

Application

Android Application

iOS Application

Server Application

[Platform] Application

Todo Application

Calendar Application

[UseCase] Application

What is the focus?

Platform

UseCase

To do Application

Android?

To do Application

iOS?

To do Application

[To do] Application

[UseCase] Application

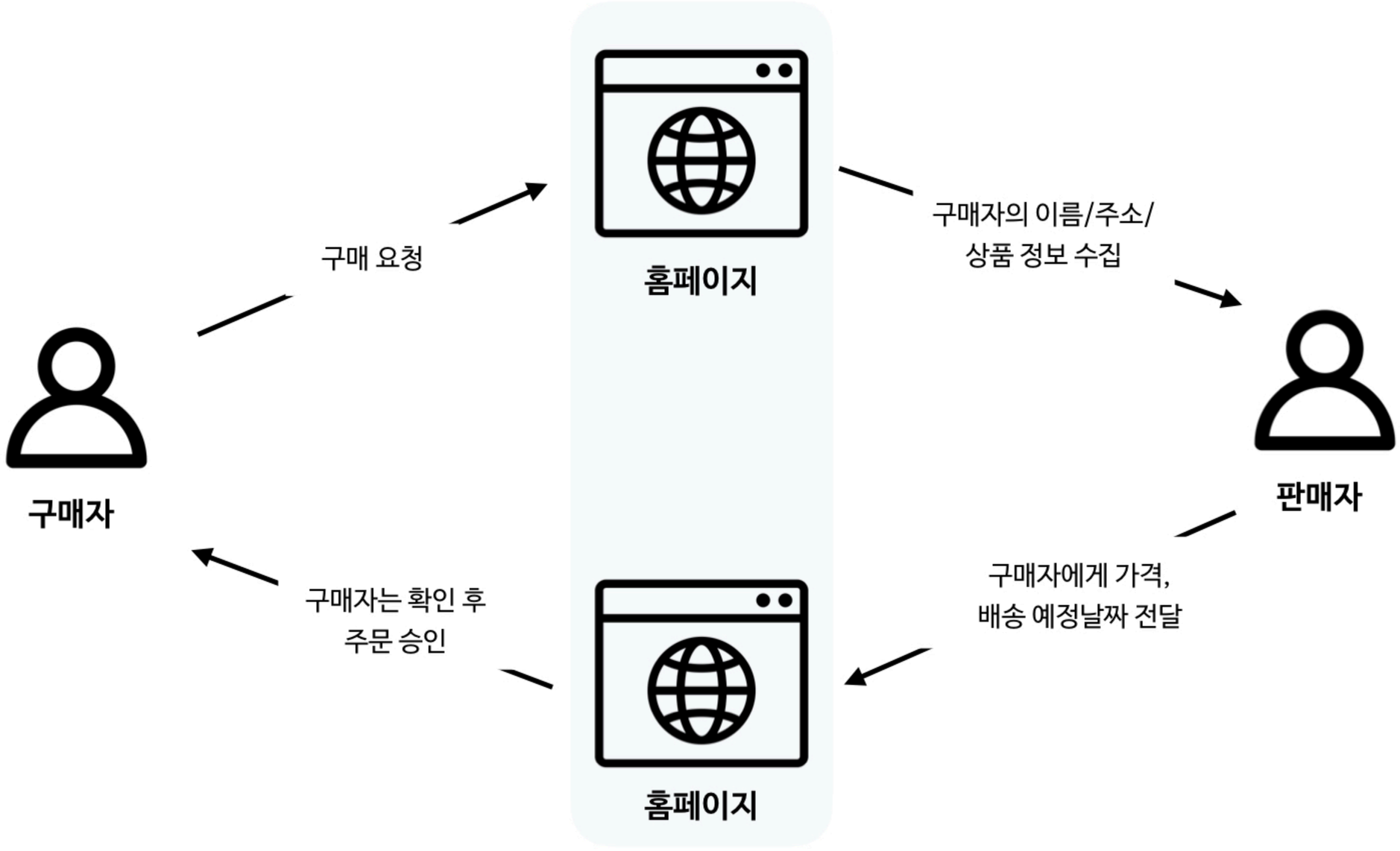
[UseCase]

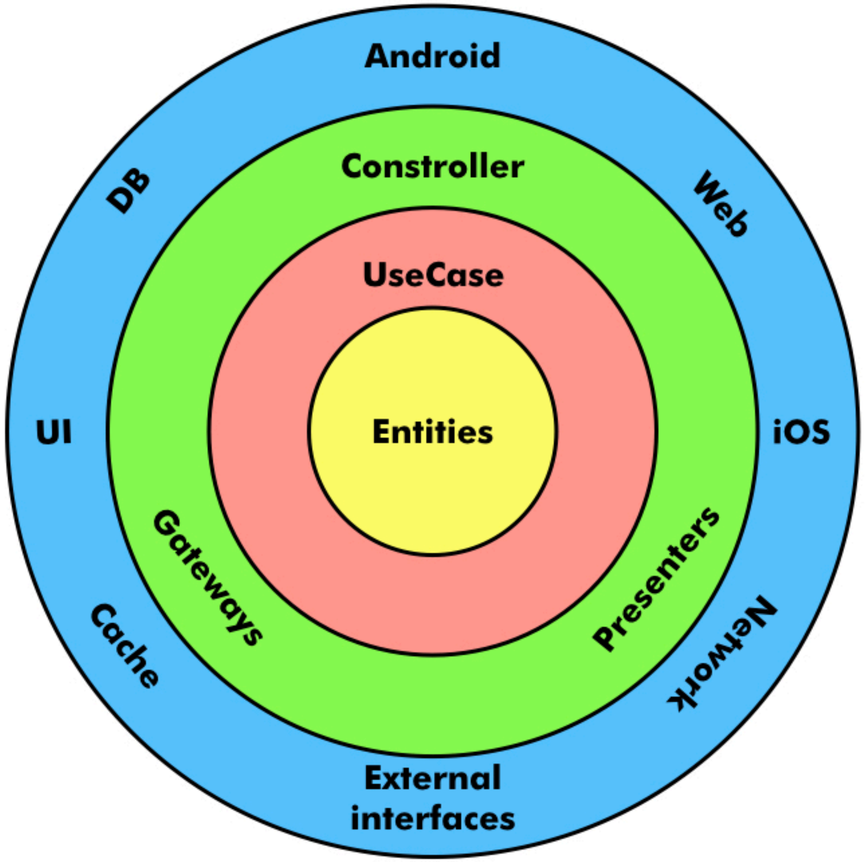
Is my **[UseCase]**
Bound to Platform?

[Architecture]

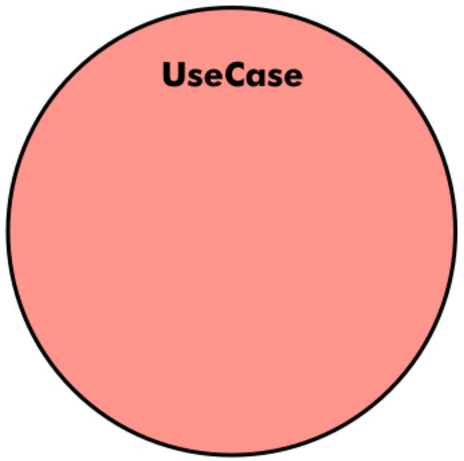
"Architecture is about intent"

- Uncle Bob

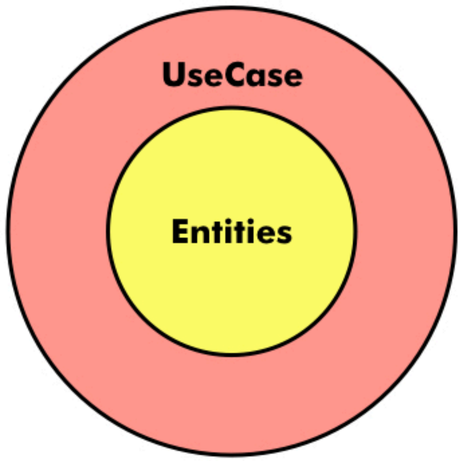

구매 요청
홈페이지
구매자의 이름/주소/
상품 정보 수집
구매자
판매자
구매자는 확인 후
주문 승인
홈페이지
구매자에게 가격,
배송 예정날짜 전달



The Clean Architecture

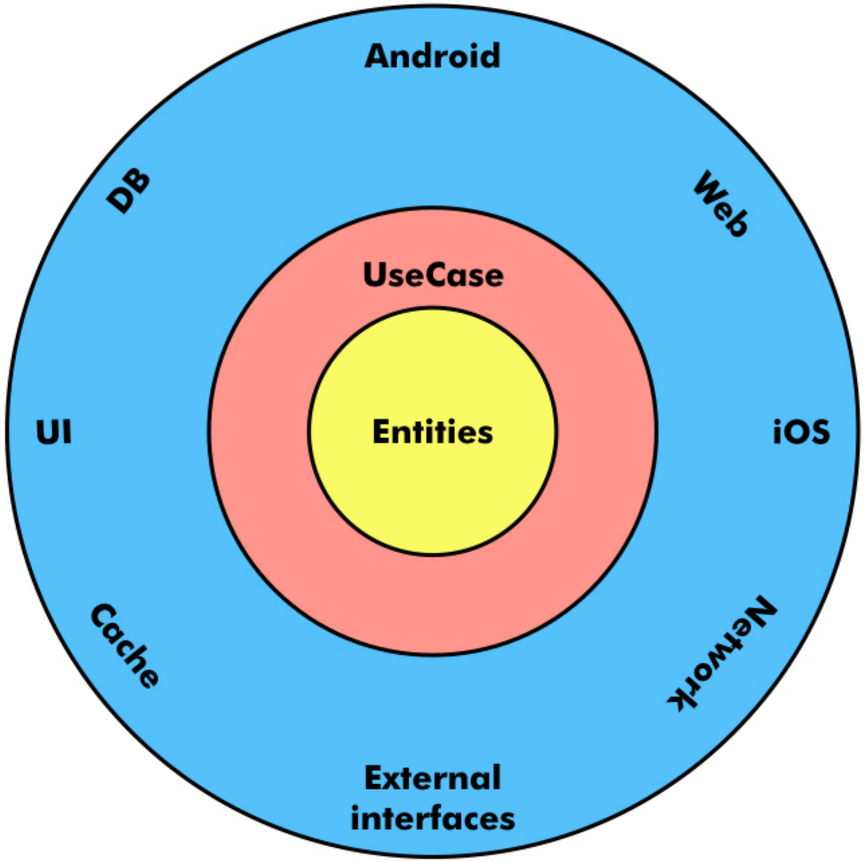


Use Case



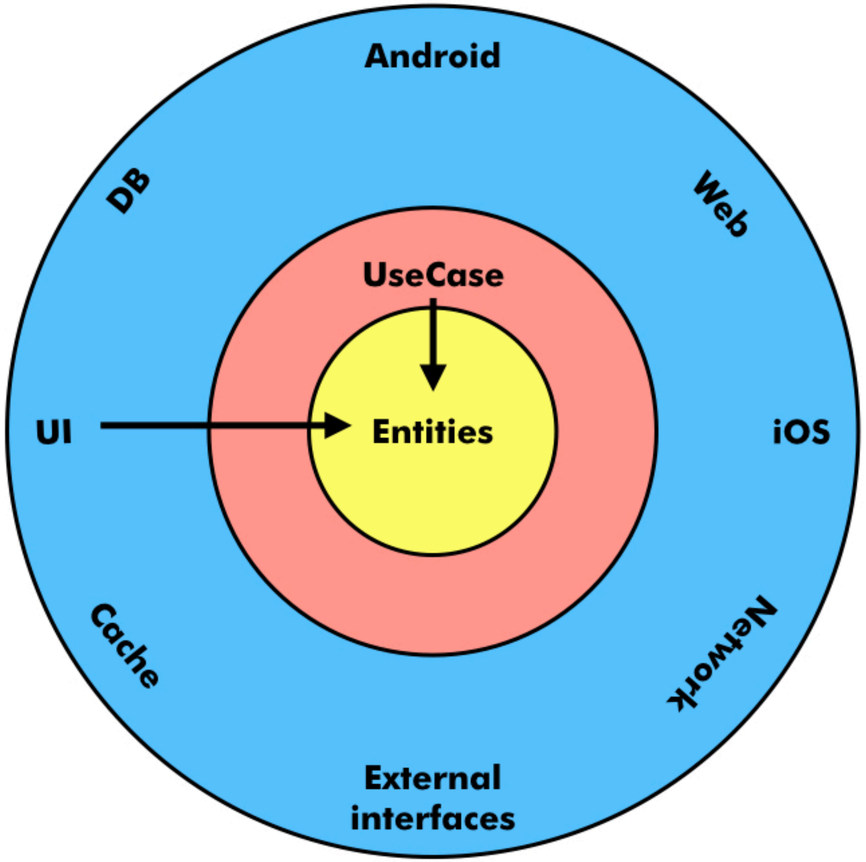
Entity

Business Object



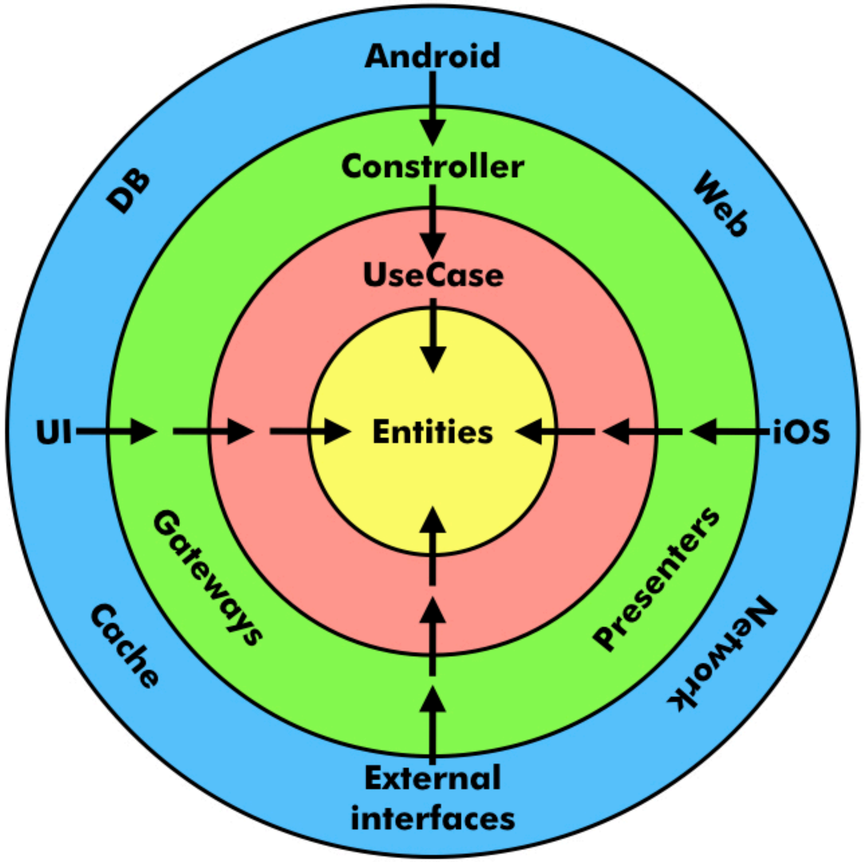
Entity

{DB, API, … }



Entity

{DB, API, … }



The Dependency Rule

Done.

So what?

Next!

Why?

#탭의 구조 및 문제점

MVVM


MVVM

VM

Presenter / Domain / Data

Presenter / Domain / Data

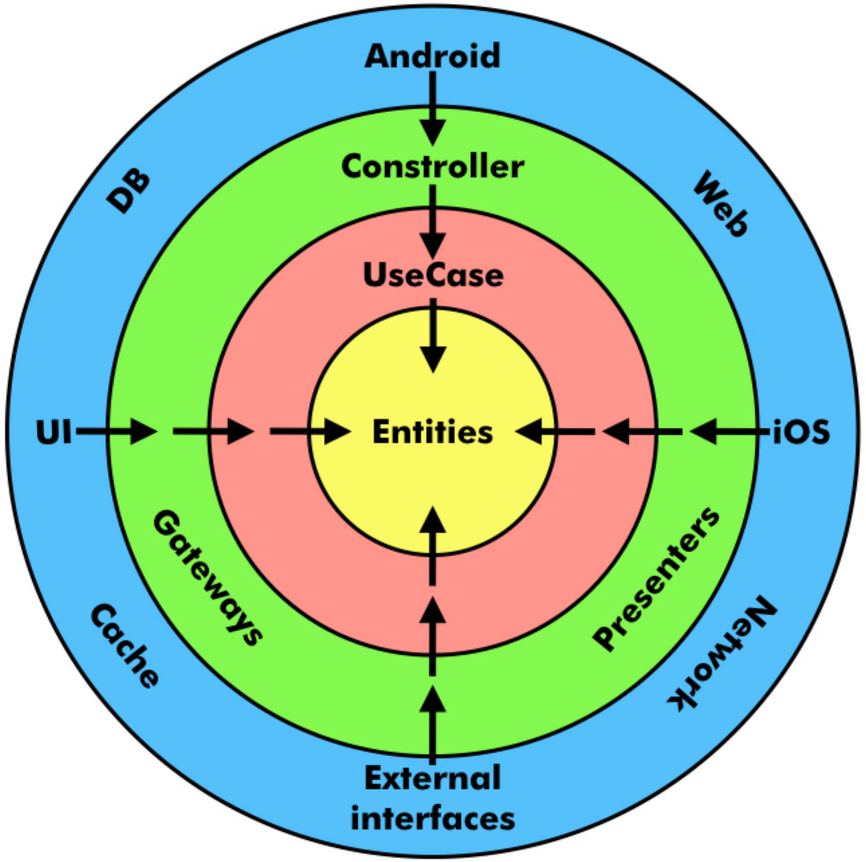
View + ViewModel

Presenter / Domain / Data

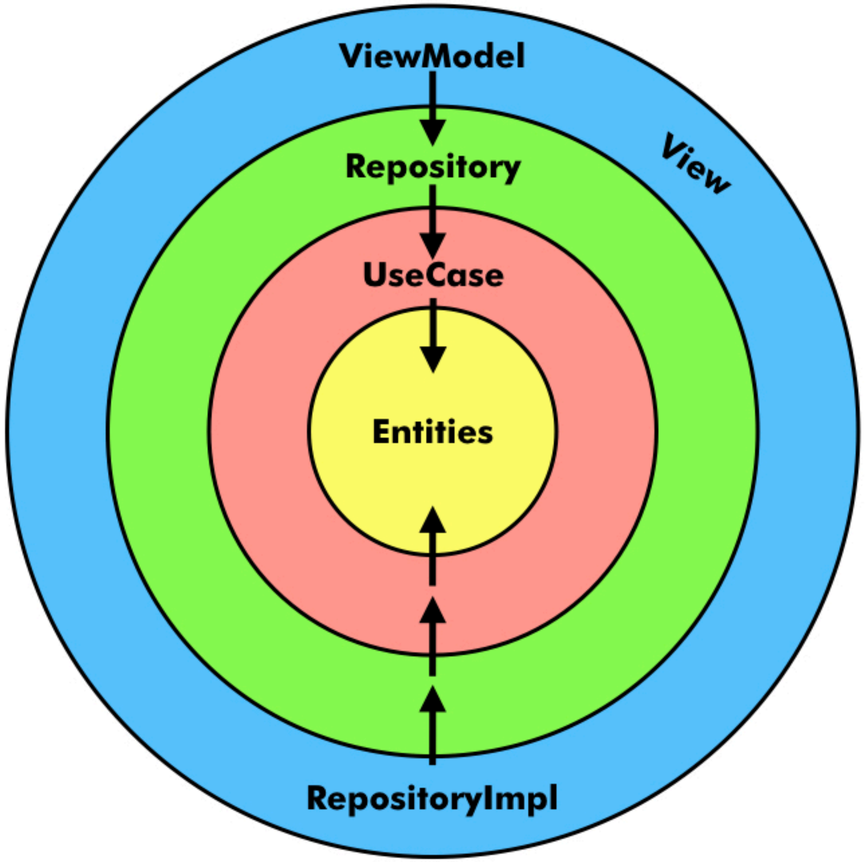
Repository + UseCase

Presenter / Domain / Data

RepositoryImpl
+ DataSource



The Dependency Rule

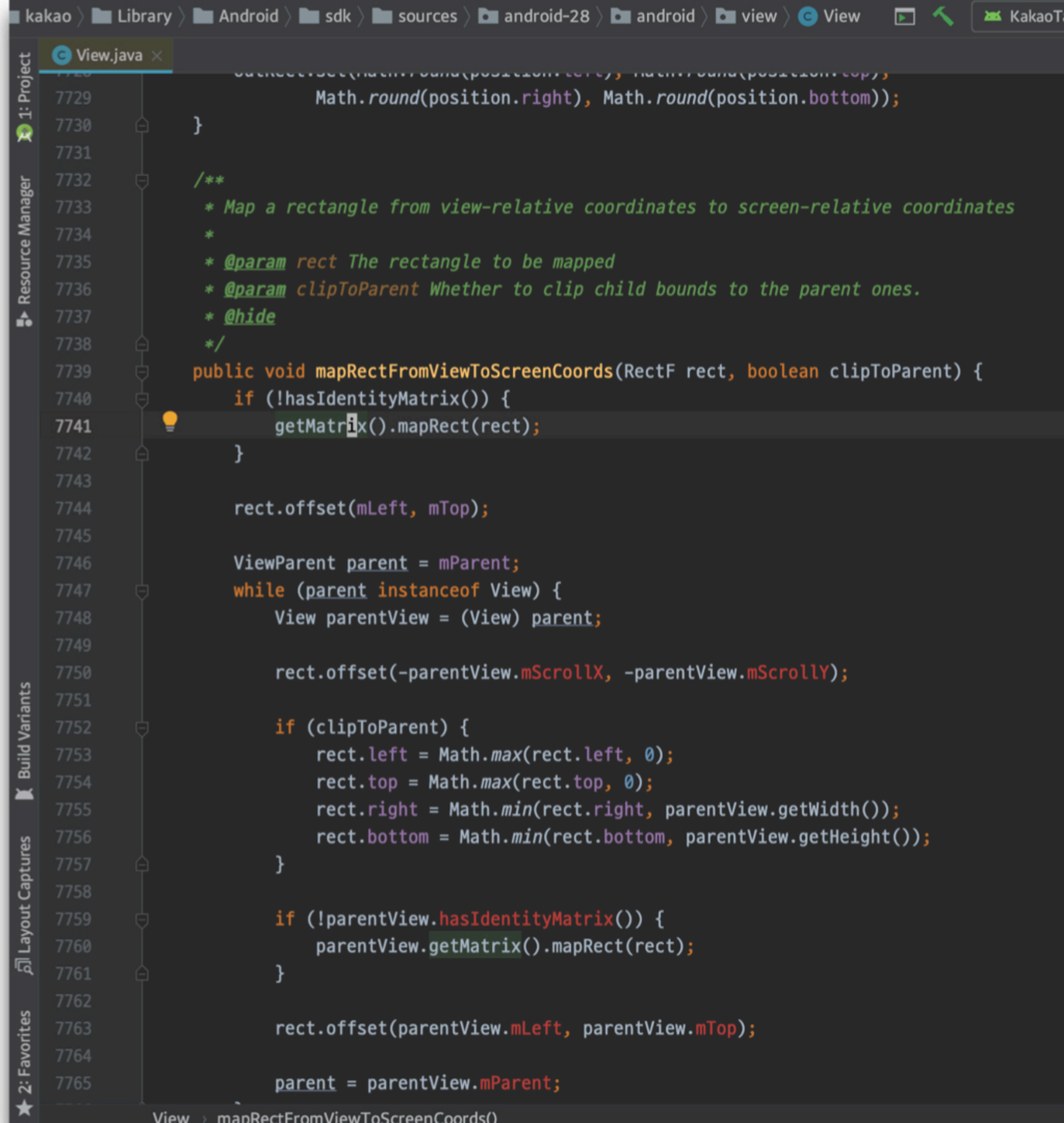


The Dependency Rule

Show me the code!

~~*Cheat Enabled*~~

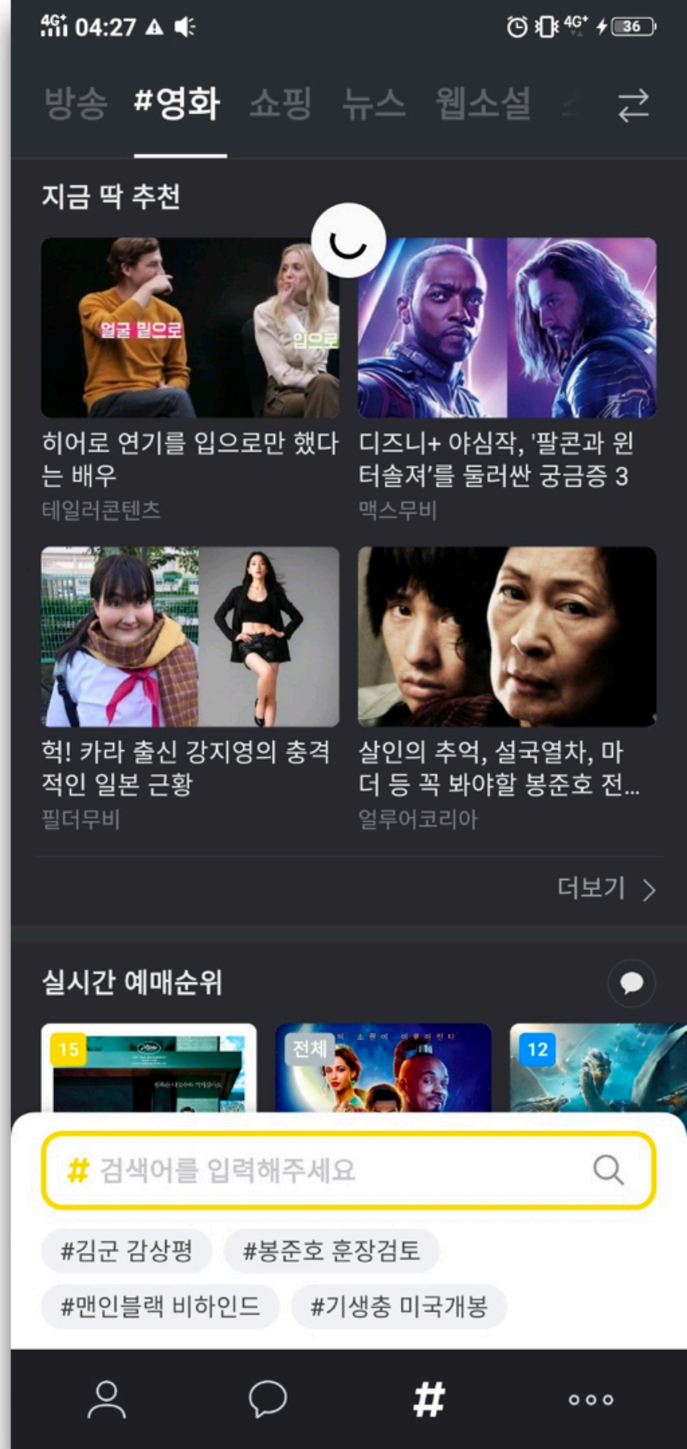
카카오톡
#탭 코드


kakao
Library
Android
sdk
sources
android-28
android
view
View
View.java
Math.round(position.right), Math.round(position.bottom));
}
/**
* Map a rectangle from view-relative coordinates to screen-relative coordinates
*
* @param rect The rectangle to be mapped
* @param clipToParent Whether to clip child bounds to the parent ones.
* @hide
*/
public void mapRectFromViewToScreenCoords(RectF rect, boolean clipToParent) {
if (!hasIdentityMatrix()) {
getMatrix().mapRect(rect);
}
rect.offset(mLeft, mTop);
ViewParent parent = mParent;
while (parent instanceof View) {
View parentView = (View) parent;
rect.offset(-parentView.mScrollX, -parentView.mScrollY);
if (clipToParent) {
rect.left = Math.max(rect.left, 0);
rect.top = Math.max(rect.top, 0);
rect.right = Math.min(rect.right, parentView.getWidth());
rect.bottom = Math.min(rect.bottom, parentView.getHeight());
}
if (!parentView.hasIdentityMatrix()) {
parentView.getMatrix().mapRect(rect);
}
rect.offset(parentView.mLeft, parentView.mTop);
parent = parentView.mParent;
View › mapRectFromViewToScreenCoords()
1: Project
Resource Manager
Build Variants
Layout Captures
2: Favorites

~~카카오톡~~

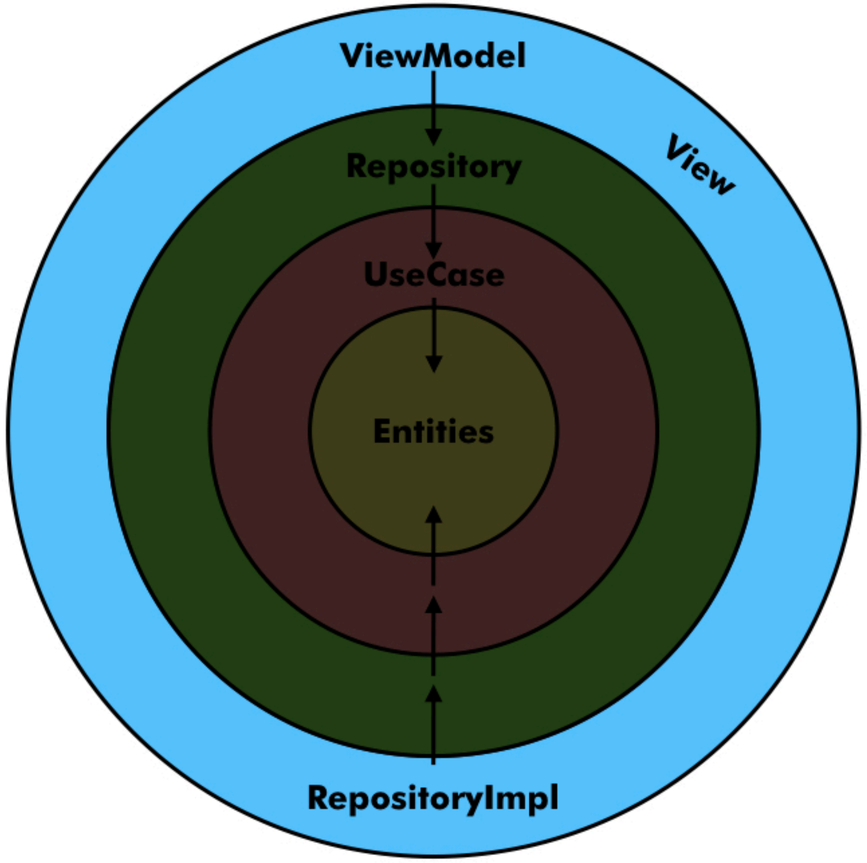
~~#탭 코드~~

예제 코드



예제 코드

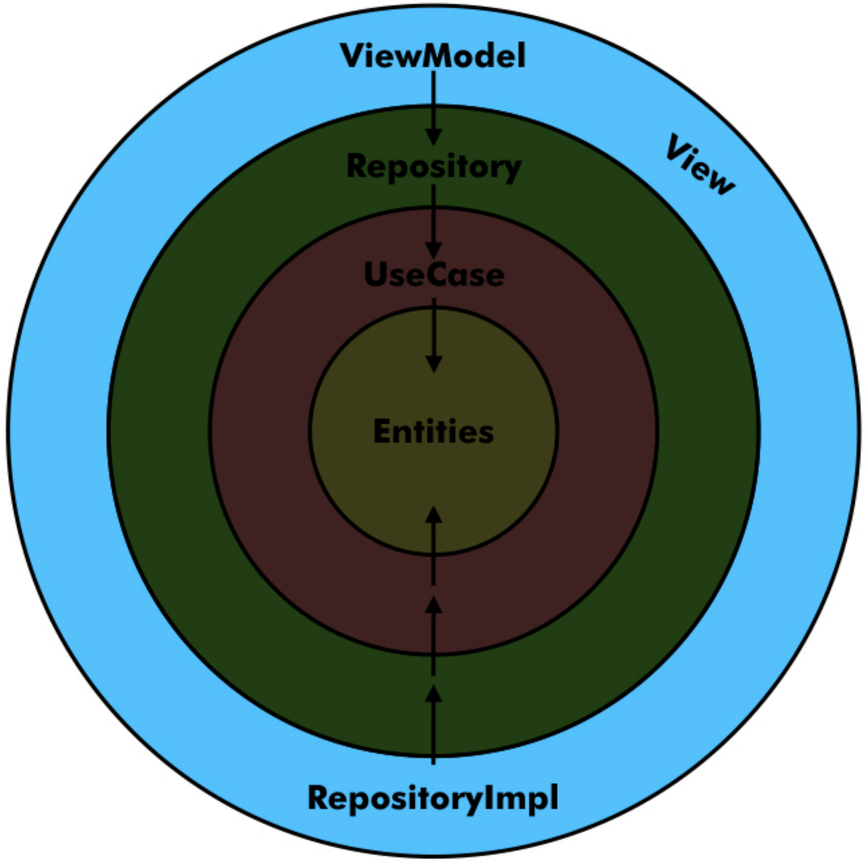
With Uncle Bob




```
swipeRefreshLayout = view.findViewById<SwipeRefreshLayout>(
    setOnRefreshListener {
        viewModel?.onSwipeRefresh()
    }
}
```

예제 코드

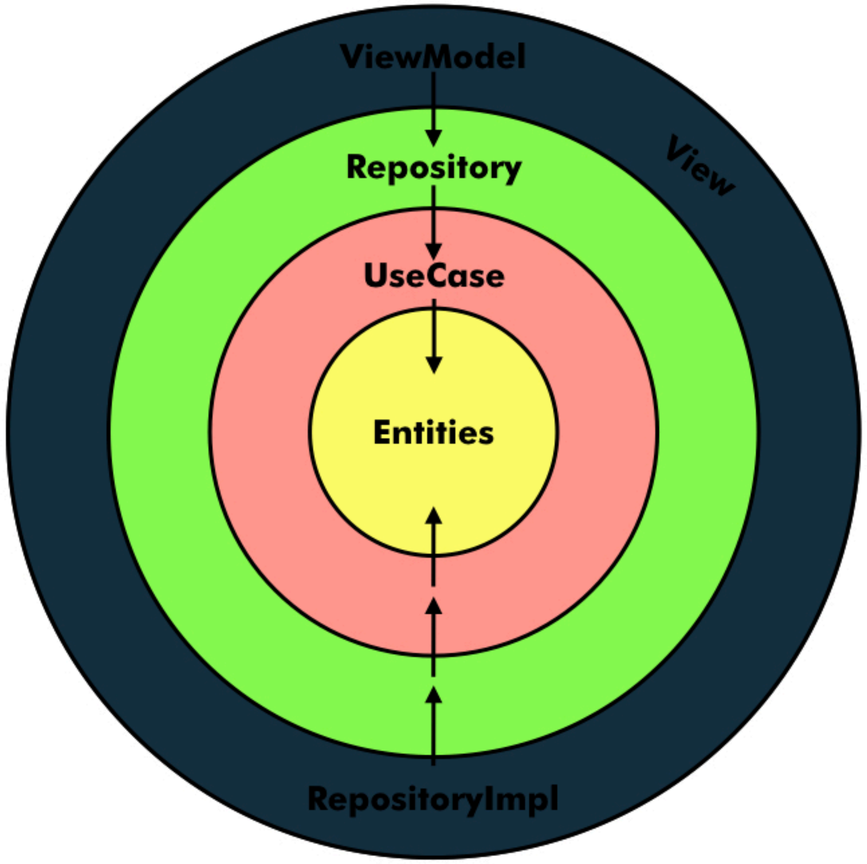
With Uncle Bob




```
fun onSwipeRefresh() {
    refreshUseCase(position, coll, onSuccess,onFail)
}
```

예제 코드

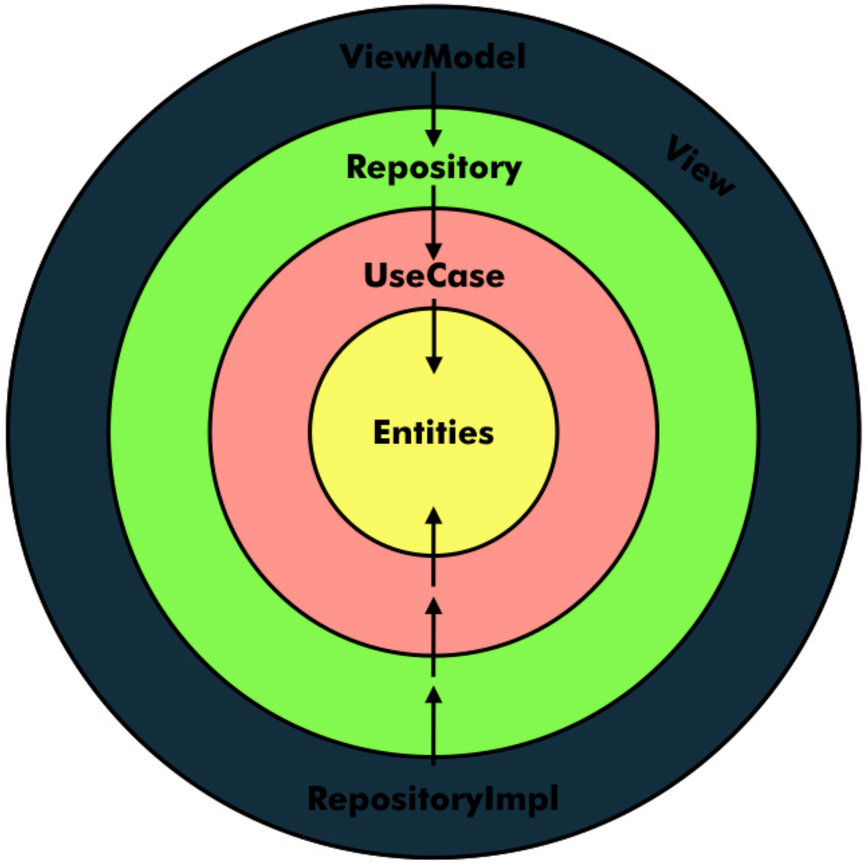
With Uncle Bob




```
class RefreshUseCase(private val repository: Repository) : UseCase() {
    operator fun invoke(position: Int,
                        coll: Item,
                        scheduler: Scheduler,
                        success: (coll: Item) -> Unit,
                        error: (throwable: Throwable) -> Unit) {
        disposable?.dispose()
        disposable = repository.refresh(position, coll)
                .observeOn(scheduler)
                .subscribe(success, error)
    }
}
```

예제 코드

With Uncle Bob




```
override fun refresh(position: Int, item: Item): Single<Item> {
    return networkDataSource.getData(headers, parameters)
            .observeOn(Schedulers.computation())
}
```

Strengths

Strengths

1. **Independent of X**
2. **Testable**
3. **Separation of Concerns**

Weakness

Weakness

1. **learning curve**
2. **Increase complexity**

"A good architecture is an architecture that allows major decisions to be delayed!"

"To delay those decisions as long as possible so that you have the most information with which to make them."

Uncle Bob

Thank you

Q&A