Making an Android app

The right way

Who am I?

Thomas Lindsjørn

Senior Android developer

Making apps for 10 years

TIDAL

Life as an app developer

- Teams / Scrum
- Job interviews
- Hobby projects

Lecture overview

Part 1: Overview - Mobile apps - Android Development - Good architecture, why and how?

Part 2: Live coding - refactor FilmAppen to use good architectural principles

Cross platform vs. Native

Cross platform / Hybrid

- Code sharing - “write once, run everywhere”
- Complicated testing - “write once, debug everywhere”
- Features not available on all platforms
- Different design guides and principles for each platform
- Smaller ecosystem: Fewer resources, more bugs
- Performance hits
- Popular tools:
    - Xamarin - C#
    - React Native - JS
    - JS + HTML: Progressive Web Apps, Cordova

Native

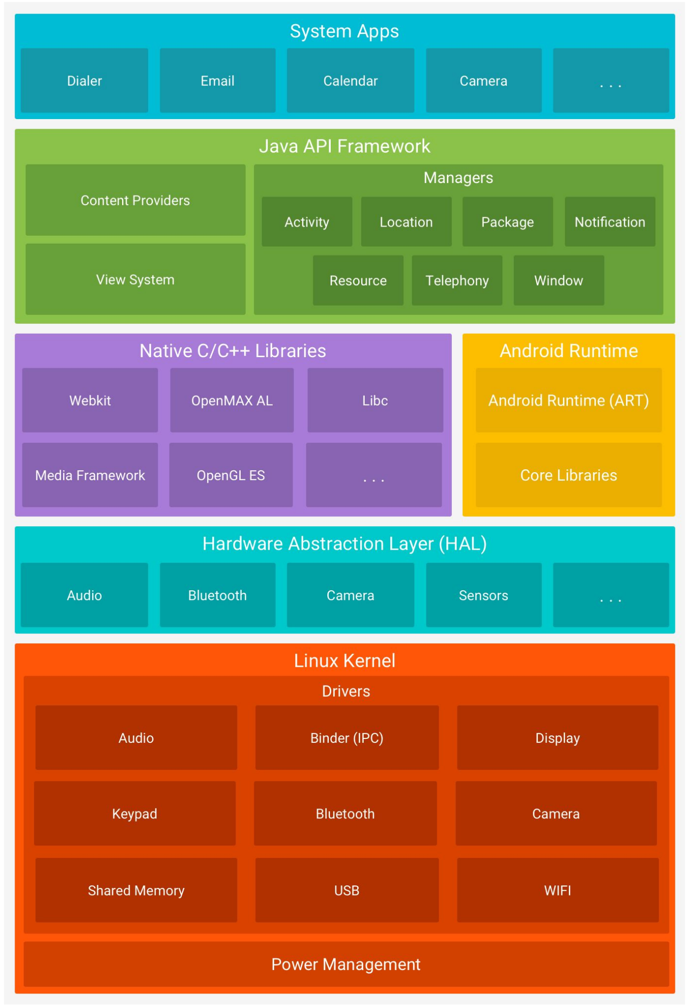
- Great tools
- Plenty of libraries
- Performance
- Design guides / principles
- More languages:
  - Java or Kotlin for Android
  - Obj-C or Swift for iOS
  - JS / other language for web
  - C# for Windows
  - Yet another language for backend?
- Costly / resource-heavy business wise in early / startup stages

Android framework

Language:

Java or Kotlin

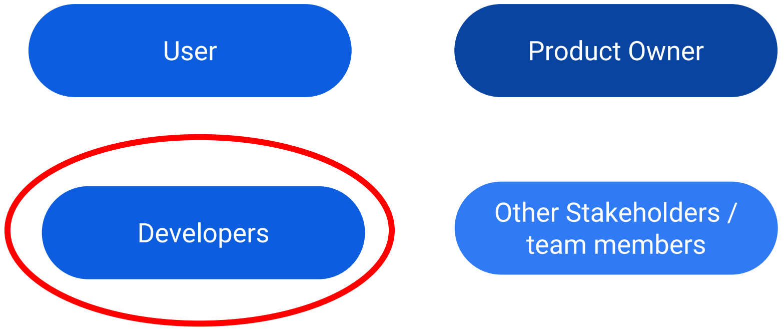
C / C++



What is a good code base?

- Robust
- Performant
- Good UX and design
- Expected behaviour

- Easy to understand and expand the code

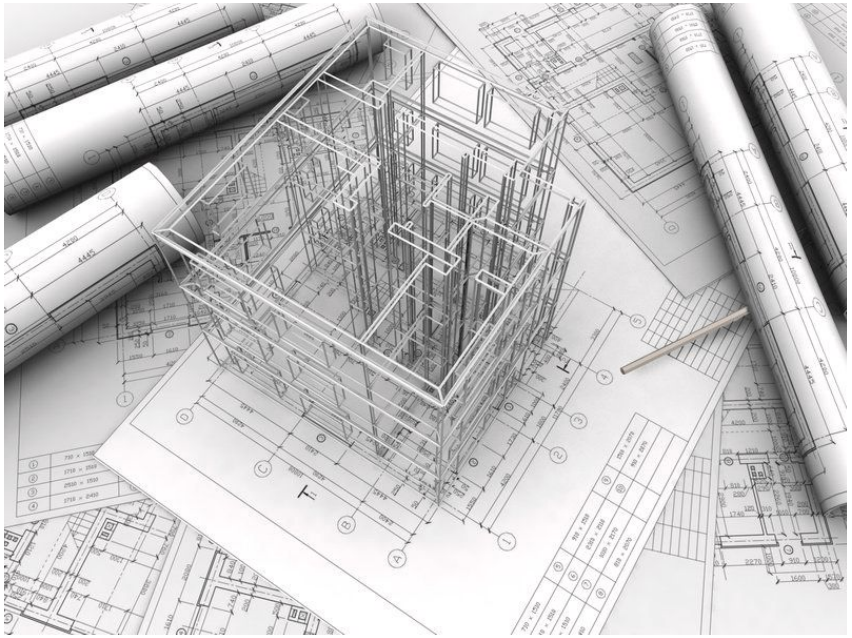


Principles of good architecture

- Maintainability
- Testability
- Performance

Separation of Concerns

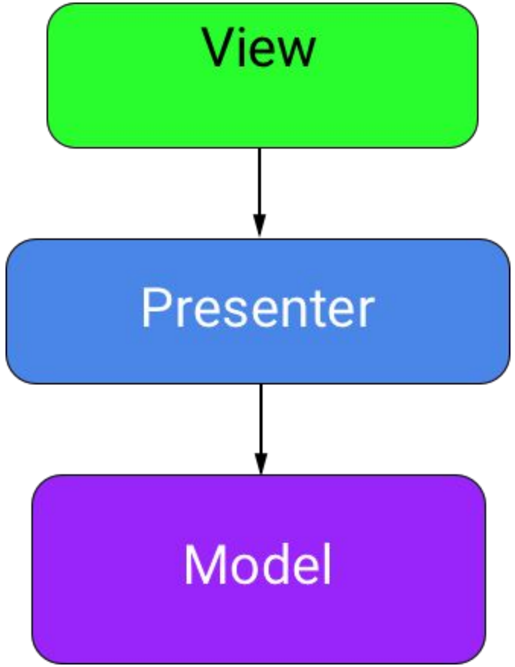
Single responsibility principle



A bad example

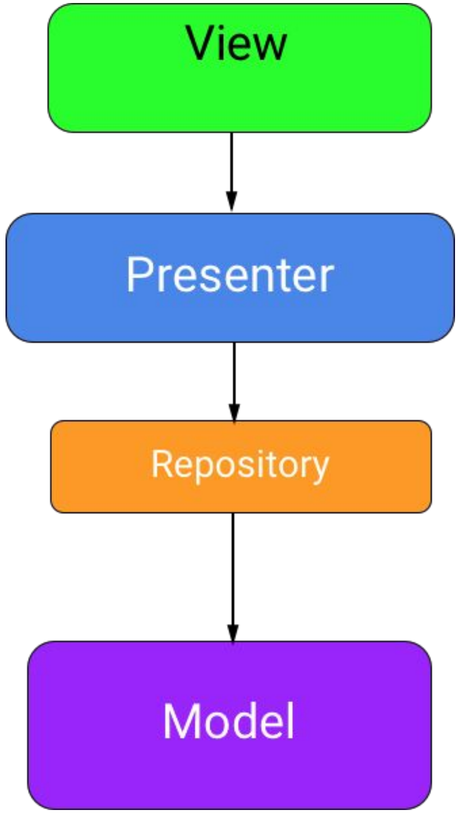
The MVP Pattern

- Separates concerns
- Provides testability

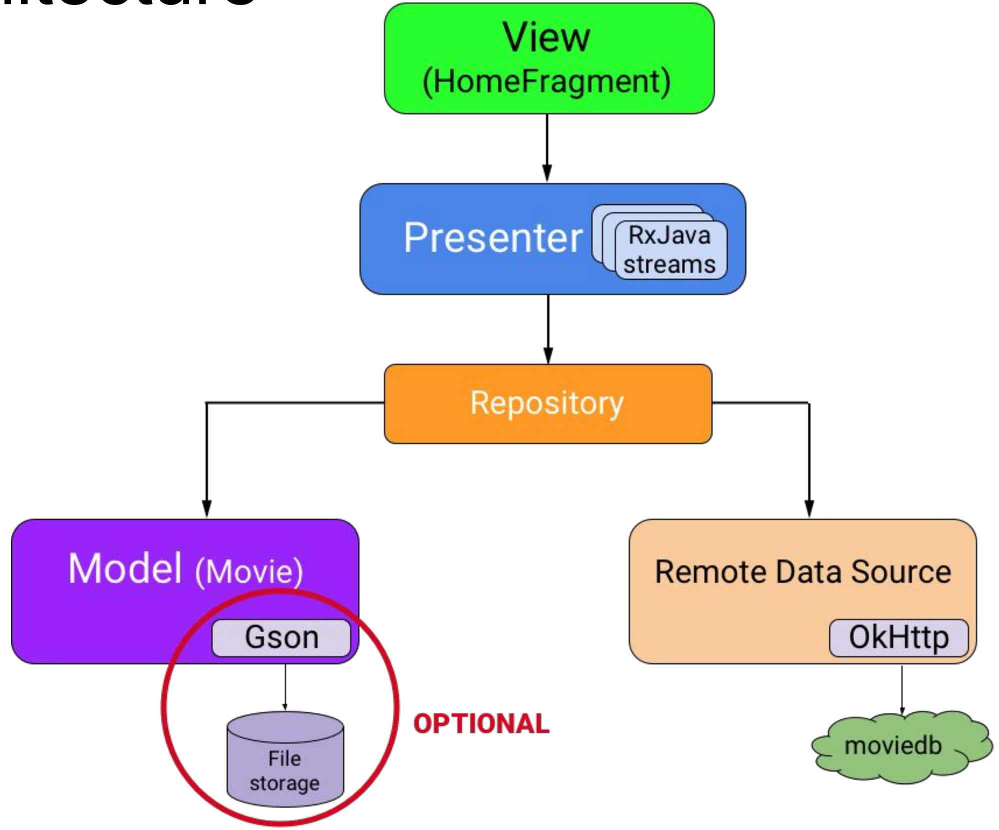


Modified MVP for Android

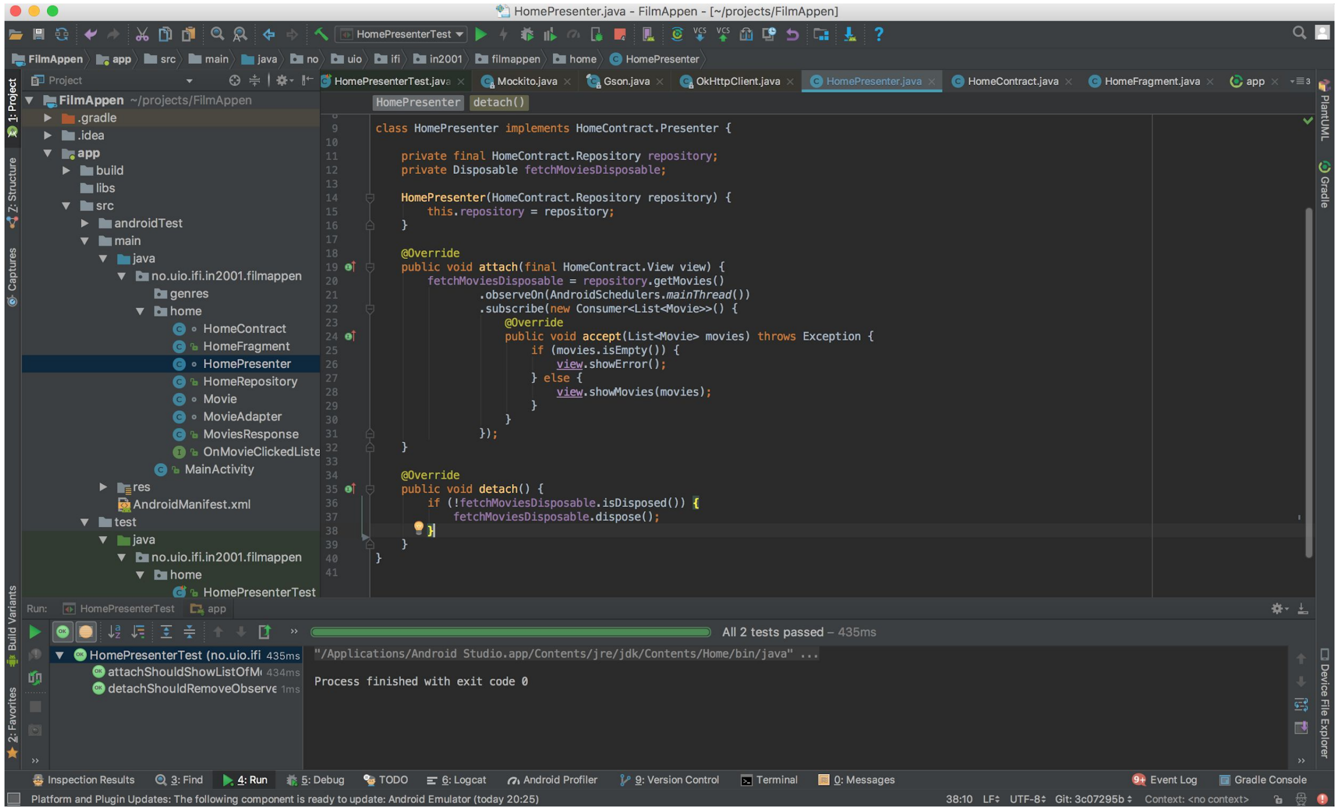
- View lifecycle out of our control
- Repository: Decide where to fetch Models from - backend or cache?



Final architecture



Live coding - Refactor FilmAppen



HomePresenter.java - FilmAppen - [~/projects/FilmAppen]

```
class HomePresenter implements HomeContract.Presenter {

    private final HomeContract.Repository repository;
    private Disposable fetchMoviesDisposable;

    HomePresenter(HomeContract.Repository repository) {
        this.repository = repository;
    }

    @Override
    public void attach(final HomeContract.View view) {
        fetchMoviesDisposable = repository.getMovies()
                .observeOn(AndroidSchedulers.mainThread())
                .subscribe(new Consumer<List<Movie>>() {
                    @Override
                    public void accept(List<Movie> movies) throws Exception {
                        if (movies.isEmpty()) {
                            view.showError();
                        } else {
                            view.showMovies(movies);
                        }
                    }
                });
    }

    @Override
    public void detach() {
        if (!fetchMoviesDisposable.isDisposed()) {
            fetchMoviesDisposable.dispose();
        }
    }
}
```

All 2 tests passed – 435ms

```
HomePresenterTest (no.uio.ifi 435ms
    attachShouldShowListOfM( 434ms
    detachShouldRemoveObserve 1ms
```

```
"/Applications/Android Studio.app/Contents/jre/jdk/Contents/Home/bin/java" ...

Process finished with exit code 0
```

Platform and Plugin Updates: The following component is ready to update: Android Emulator (today 20:25)