Dependency Injection
DI Using Dagger

WIZELINE®

Introduction to Concepts

INTRODUCTION TO SOLID & OOP

SOLID

Set of rules and good practices (principles) to develop software in a more stable and well done way.

S -> Single Responsibility
O -> Open/Closed
L -> Liskov Substitution
I -> Interface Segregation
D -> Dependency INVERSION

OOP

Try to transform/recreate objects in the real world into code

INTRODUCTION TO SOLID & OOP

SOLID

Set of rules and good practices (principles) to develop software in a more stable and well done way.

S -> Single Responsibility
O -> Open/Closed
L -> Liskov Substitution
I -> Interface Segregation
D -> Dependency INVERSION

OOP

Try to transform/recreate objects in the real world into code

S

Single Responsability

L

Liskov's Principle

Liskov's Principle

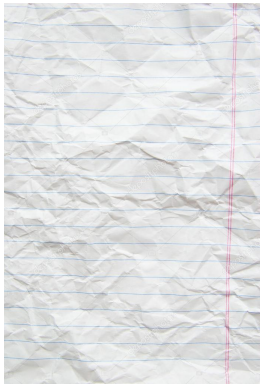
```
List<Data> mList = new LinkedList<>( );
workWithMyData(mList);
//...
ArrayList<Data> aList = new ArrayList<>( );
workWithMyData(aList);

// ...
public void workWithMyData(List<Data> list) {
// ...
}
```

D

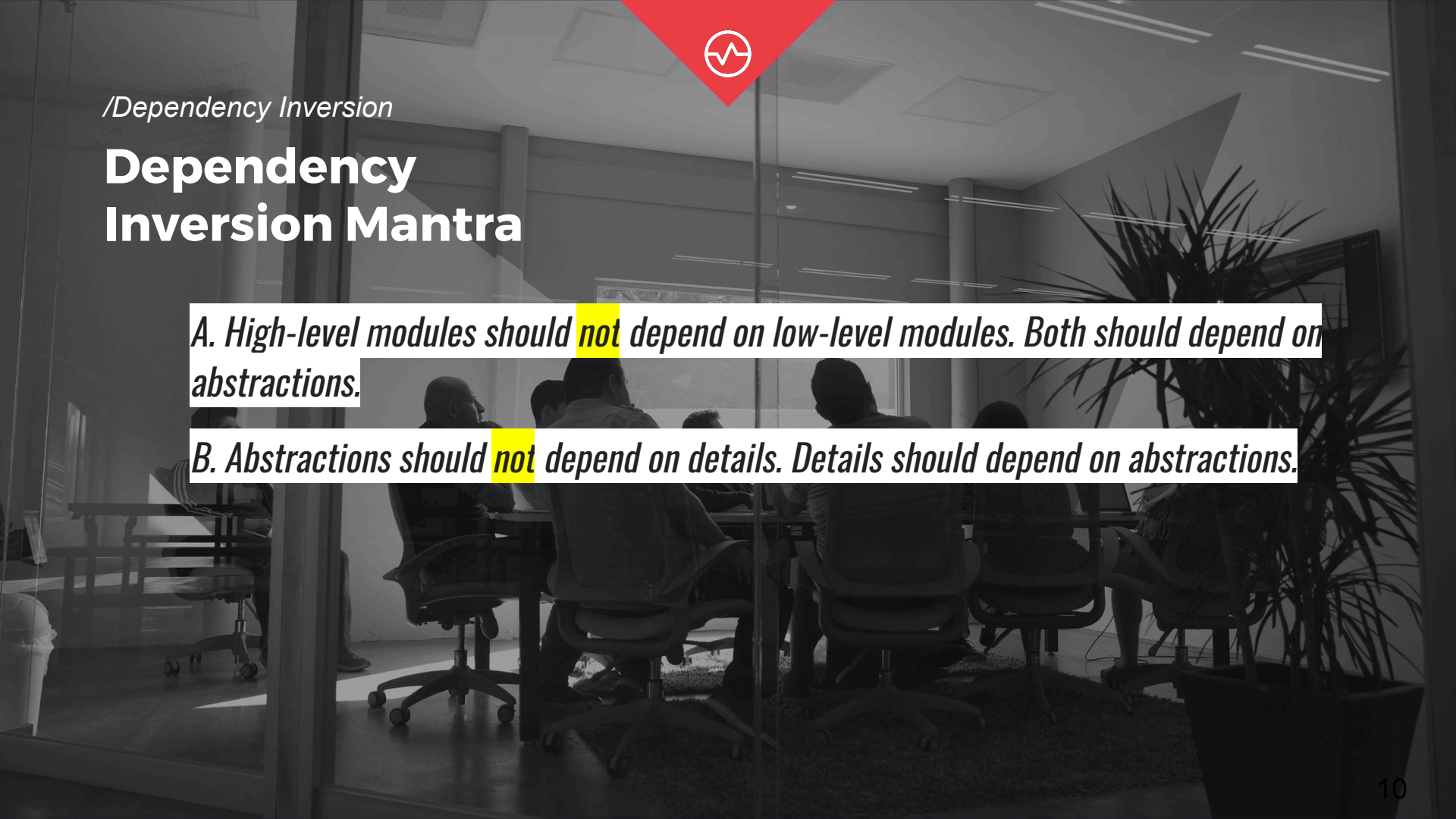
Dependency Inversion

(Still not Dependency Injection)

What's Dependency Inversion?



Writer → Pen → Ink → Paper



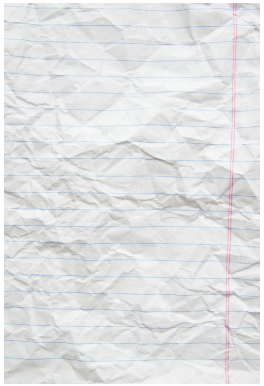
/Dependency Inversion

Dependency Inversion Mantra

A. High-level modules should not depend on low-level modules. Both should depend on abstractions.

B. Abstractions should not depend on details. Details should depend on abstractions.

What's Dependency Inversion?



Writer ← Pen ← Ink

Paper ←

Finally!!!!

Dependency Injection

What's Dependency Injection?

- An special kind of Dependency INVERSION
- A solution for some kind of design problems
- One way to avoid some problems when making changes
- A fashion way to call something that you maybe are already doing

What is NOT Dependency Injection?

- A solution to ALL your problems
- A library
- A framework
- A person

Achieve Dependency Injection

- **Method**

Achieve Dependency Injection

- Method
- Constructor

Achieve Dependency Injection

- Method
- Constructor
- Annotations (Java EE)

Achieve Dependency Injection

- **Method**
- **Constructor**
- **Annotations (Java EE)**
- **Other ways? Maybe some design Patterns that allow the injection of dependencies?**

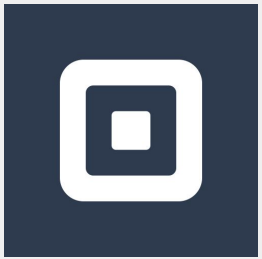
Achieve Dependency Injection

- **Method**
- **Constructor**
- **Annotations (Java EE)**
- **Other ways? Maybe some design Patterns that allow the injection of dependencies?**
- **Third Parties/Libraries /Frameworks**

Dagger

A Brief into history

Square Inc



Dagger 1

- Works by Reflection

A Brief into history

Google

Dagger 2

- Works by Code Generation
- Uses annotation processor to create classes

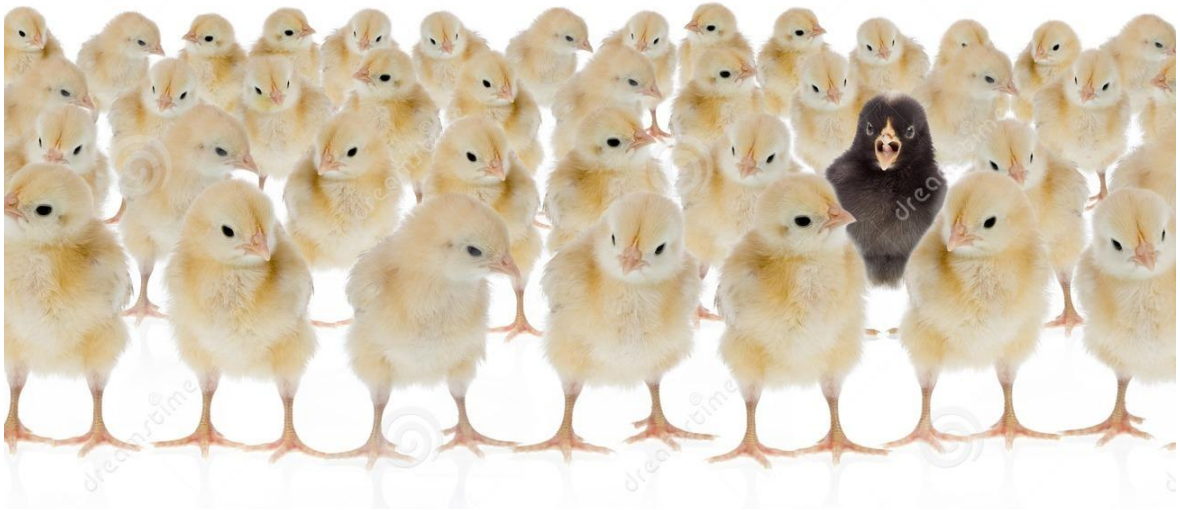
Initial Concepts

- **Parts/Elements of Dagger**

Module

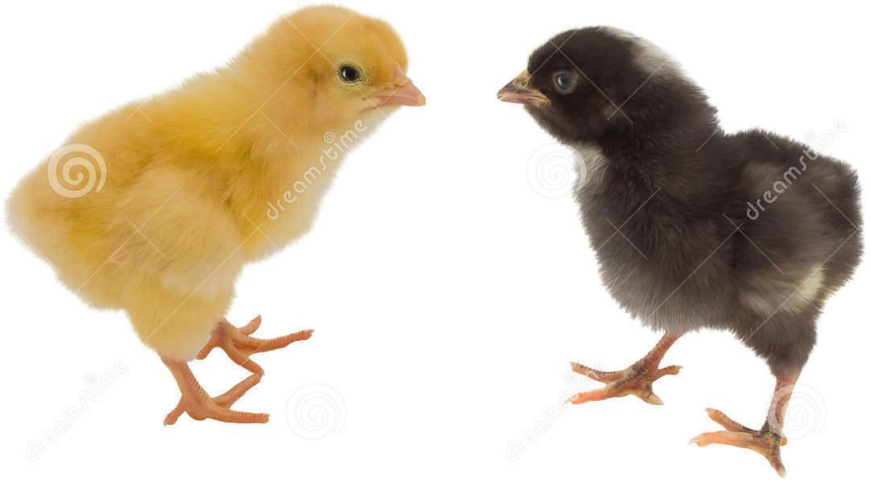
Component

Singleton



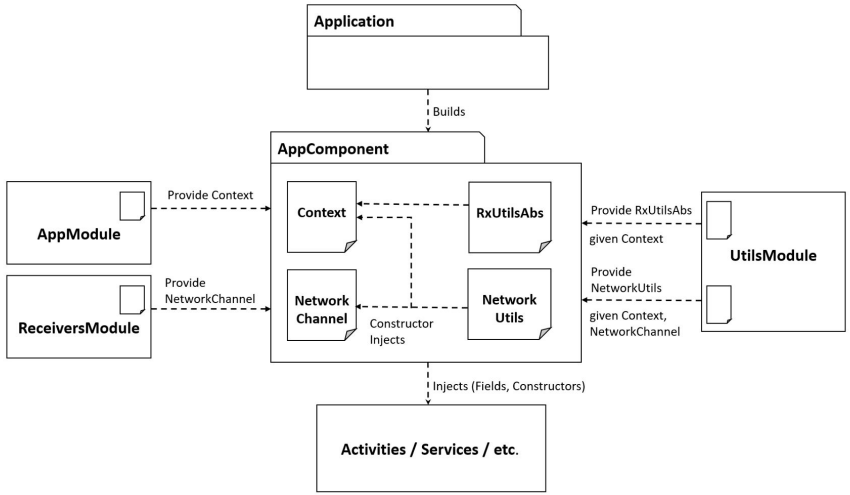
Inject

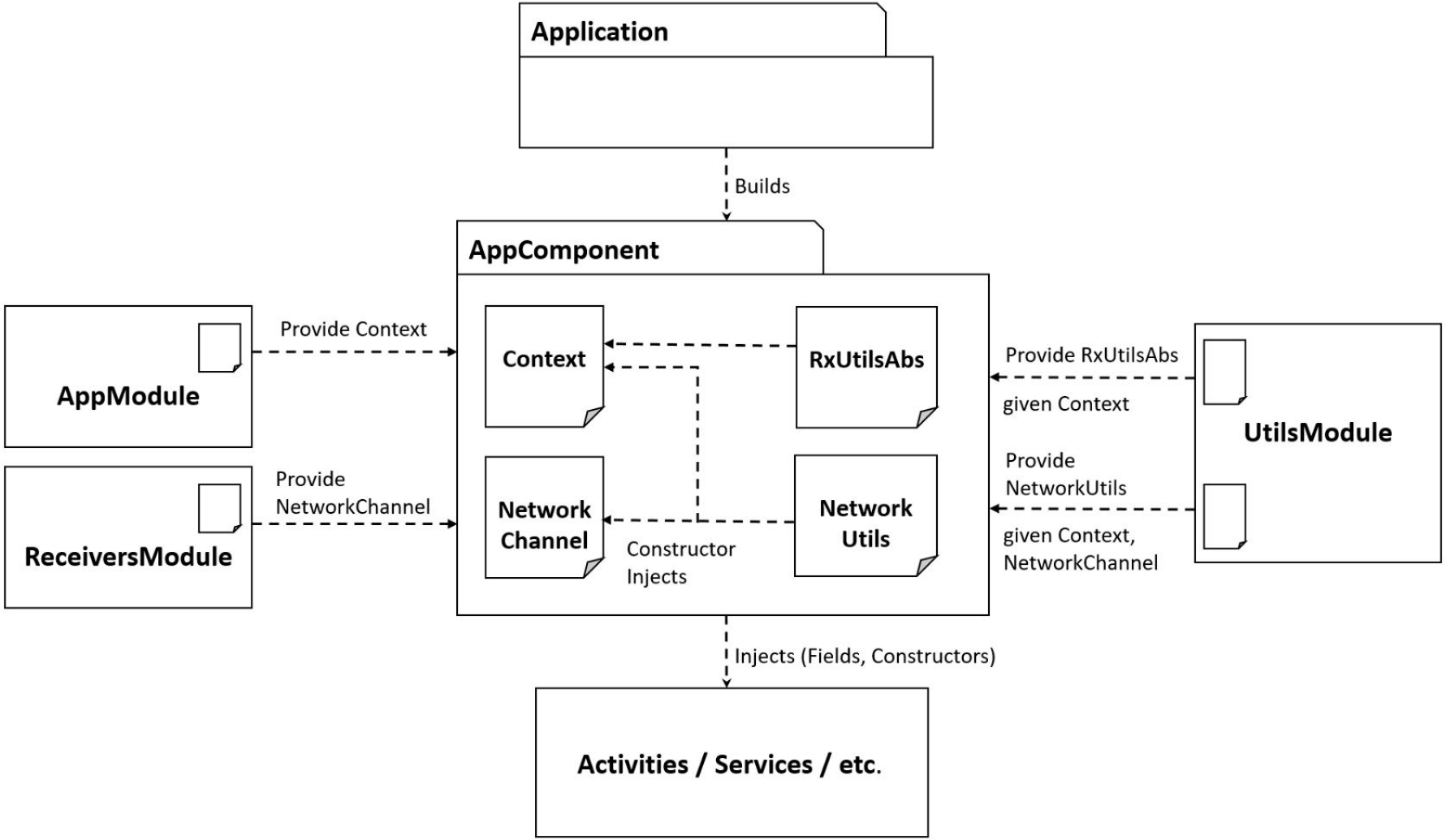
Scope



Initial Concepts

- **Build graph/Dependency Graph**




Application
Builds
AppComponent
AppModule
Provide Context
Context
RxUtilsAbs
Provide RxUtilsAbs
given Context
UtilsModule
ReceiversModule
Provide
NetworkChannel
Network
Channel
Network
Utils
Provide
NetworkUtils
given Context,
NetworkChannel
Constructor
Injects
Injects (Fields, Constructors)
Activities / Services / etc.

Initial Concepts

- **Annotations**

```
@Provides
```

Basic Dagger Elements

- **@Inject** -> Base annotation, represents "*dependency is requested*"
- **@Module** -> Classes which methods "*provide dependencies*"
- **@Provide** -> Methods inside **@Module**, which "*tell Dagger how we want to build and present a dependency*"
- **@Component** -> Bridge between **@Inject** and **@Module**
- **@Scope** -> Enables to create "*global*" and "*local*" singletons
- **@Qualifier** -> If different objects of the same type are necessary

Deeper into Dagger Elements

- **@Singleton** -> Used to indicate that no matter who asks for this dependency... you always get the same instance.
- **@Subcomponent** -> Used to indicate when a **@Component** is child of another. A **@Component** can be both, father and child of other components, but as the same in java, you have to keep this in a straight line.


DEMO

WIZELINE®

sales@wizeline.com