មជ្ឈមណ្ឌលកូរ៉ែ សហ្វវែរ អេច អរ ឌី

Korea Software HRD Center

Dependency Injection

Webcash

KOICA
Korea International Cooperation Agency

What is Dependency Injection?

- Dependency Inject is a way to push the dependency from outside into the class
- Ask third party to create object, instead of create dependencies yourself
- Decouple classes construction from construction of your dependency

This is Dependency

```
class Car {
    Engine engine = new Engine();
}

class Engine {

}
```

This is Dependency Injection

```
class Car {
    Engine engine;
    Car (Engine engine) {
        this.engine = engine;
    }
}
```

Why Dependency Injection?

- Easy to reuse of components
- Easy to Test

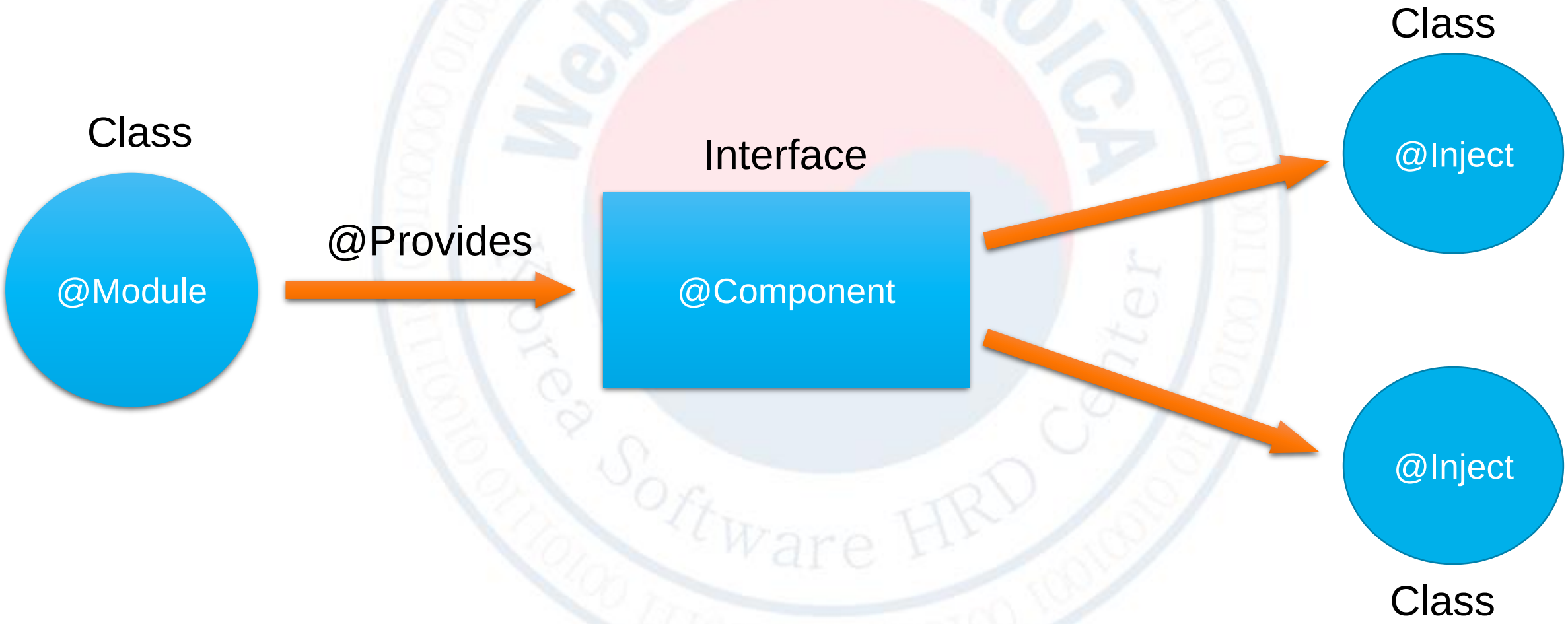
DI Framework

- Google Guice
- Spring DI
- Dagger

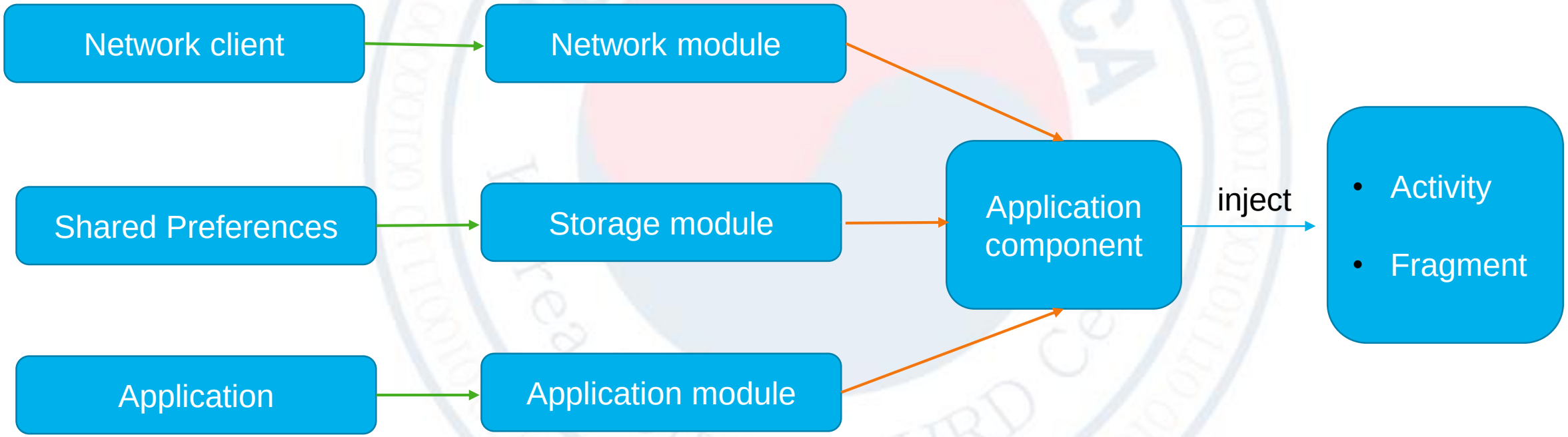
Dagger 2

Dagger 2 Usage

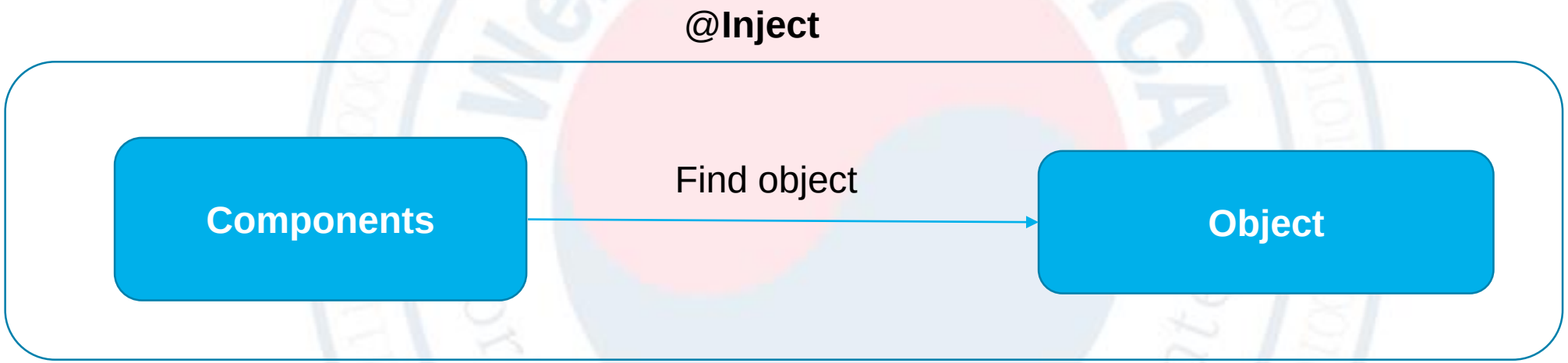
- @Module : Contain the dependencies (Class Level)
- @Provides : To create dependencies (Method Level)
- @Component: A bridge between Module & Injection
- @Named : To name a dependency in case there are the same types by using string as a name
- @Qualifier : To name a dependency in case there are the same types by using custom annotation
- @Scope : To define the scope of dependency
- @Inject : To request dependency


Class
@Module
@Provides
Interface
@Component
Class
@Inject
@Inject
Class

Modules & Components



@Inject



Take all fields with annotation,
look in the component and find the object by type, and set the field.

Type of Injection

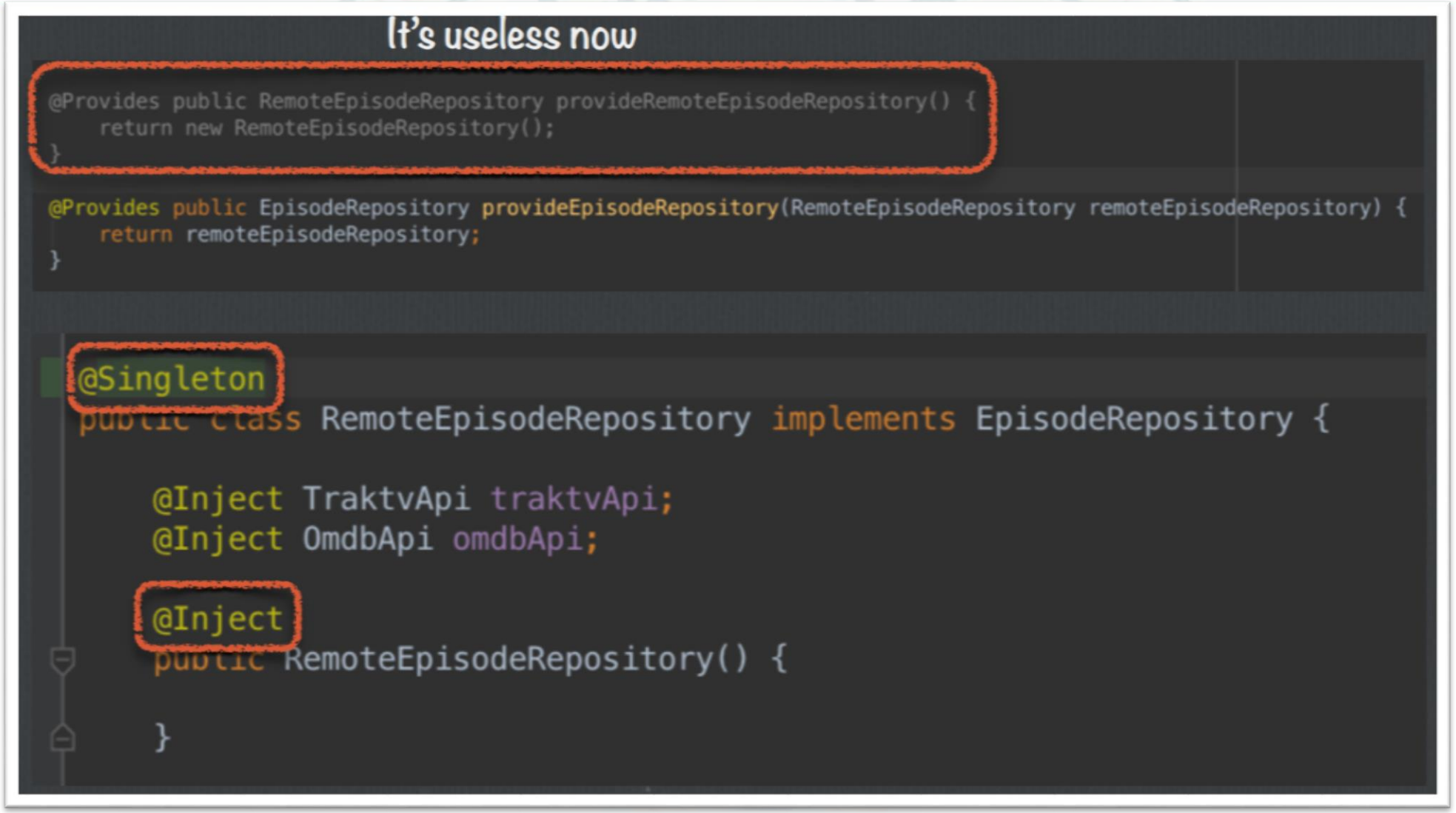
- Field (Can not be private or final)
- Constructor
  - Constructor parameter are dependencies
- Method

Constructor Injection

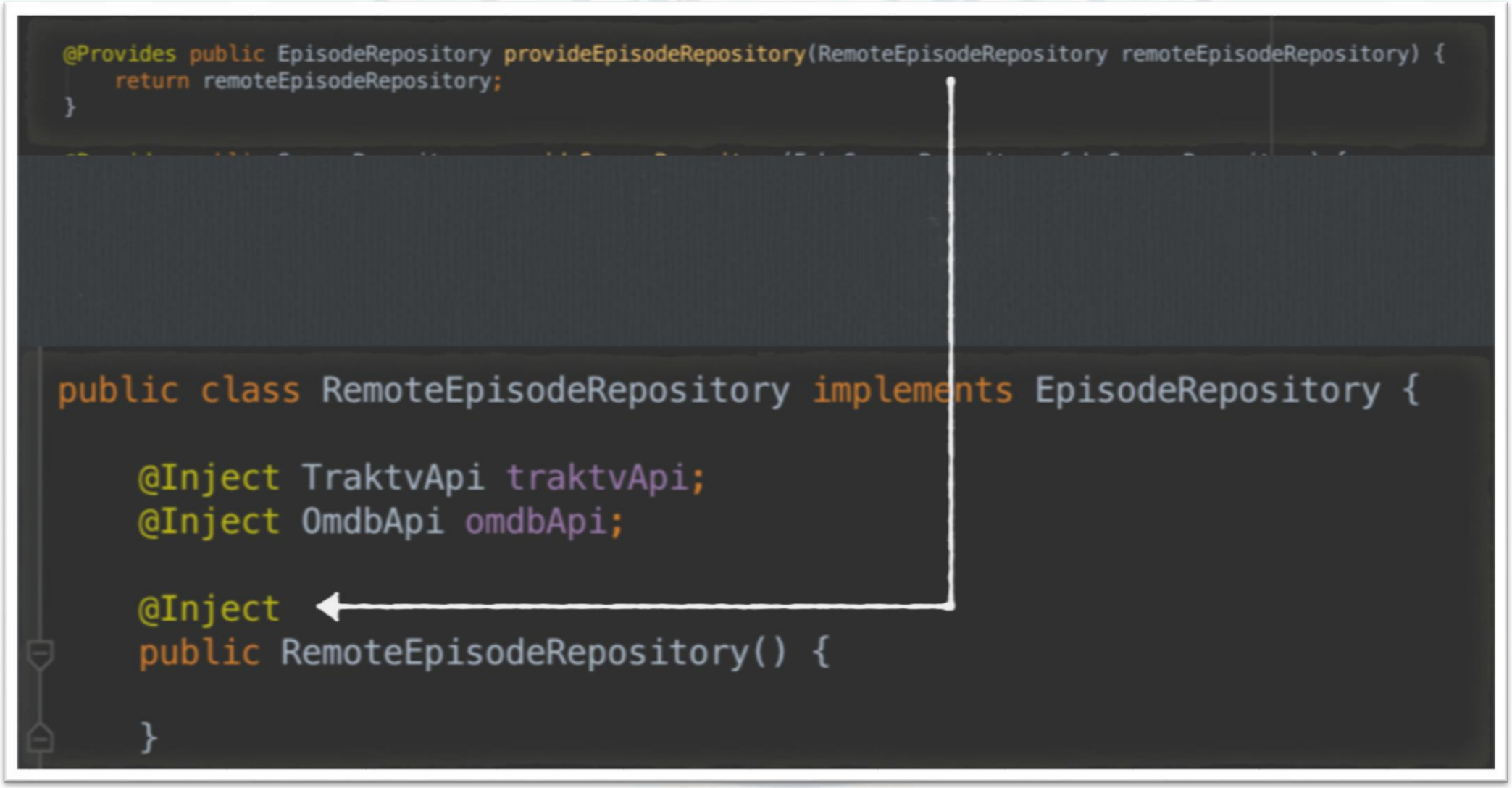
```
    @Provides @Singleton public RemoteEpisodeRepository provideRemoteEpisodeRepository() {
        return new RemoteEpisodeRepository();
    }

    @Provides public EpisodeRepository provideEpisodeRepository(RemoteEpisodeRepository remoteEpisodeRepository) {
        return remoteEpisodeRepository;
    }
}
```

```
public class RemoteEpisodeRepository implements EpisodeRepository {

    @Inject TraktvApi traktvApi;
    @Inject OmdbApi omdbApi;

    public RemoteEpisodeRepository() {

    }
```

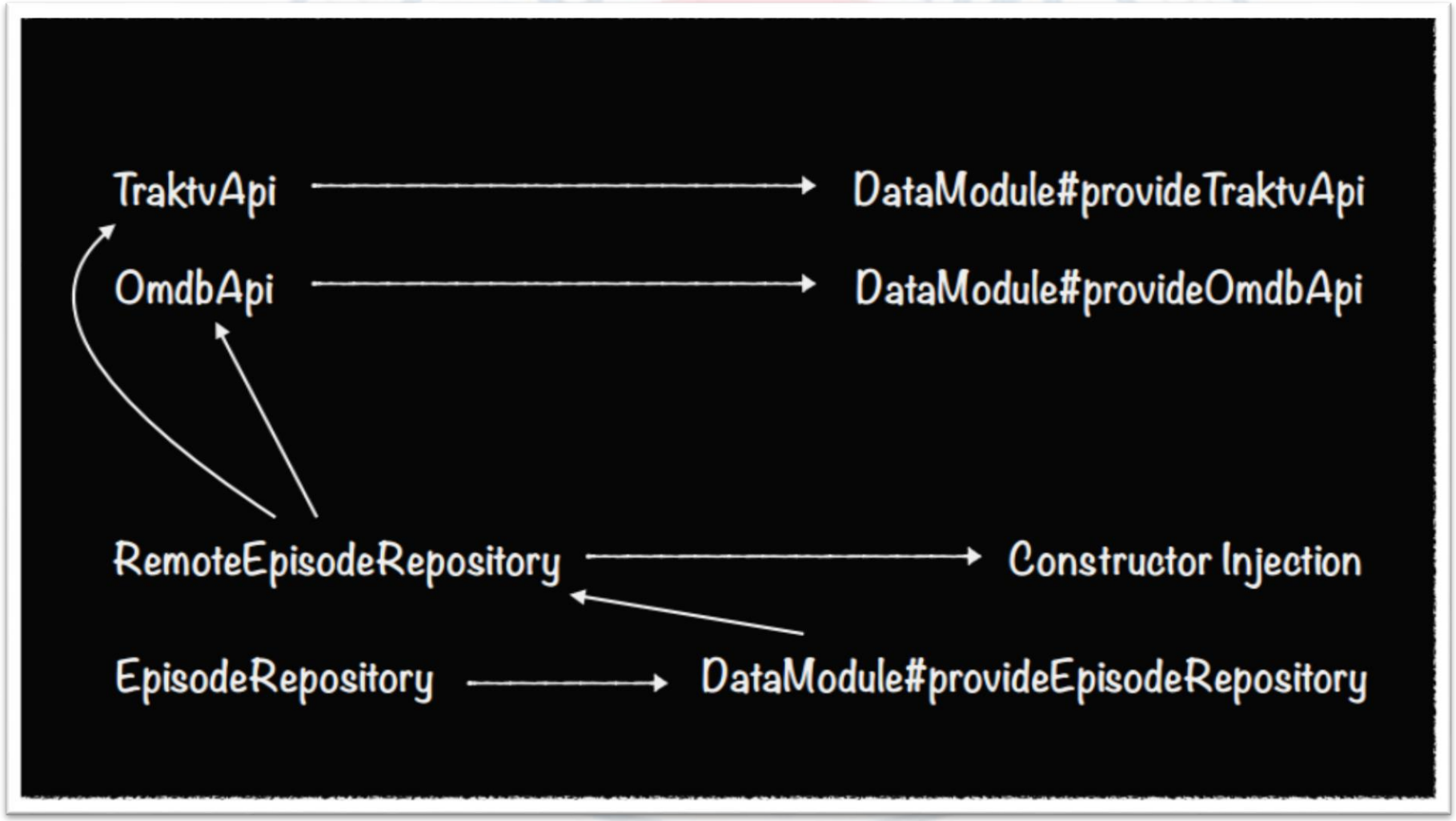
Constructor Injection



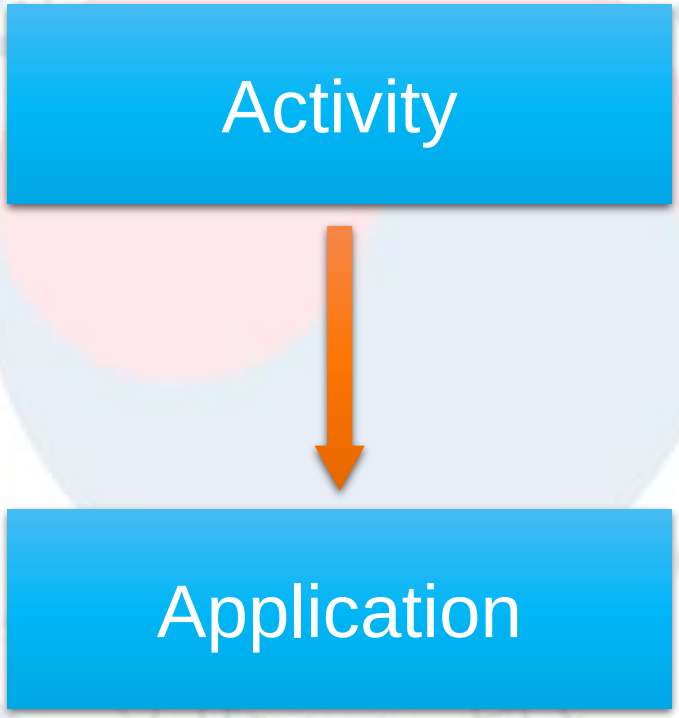
Constructor Injection



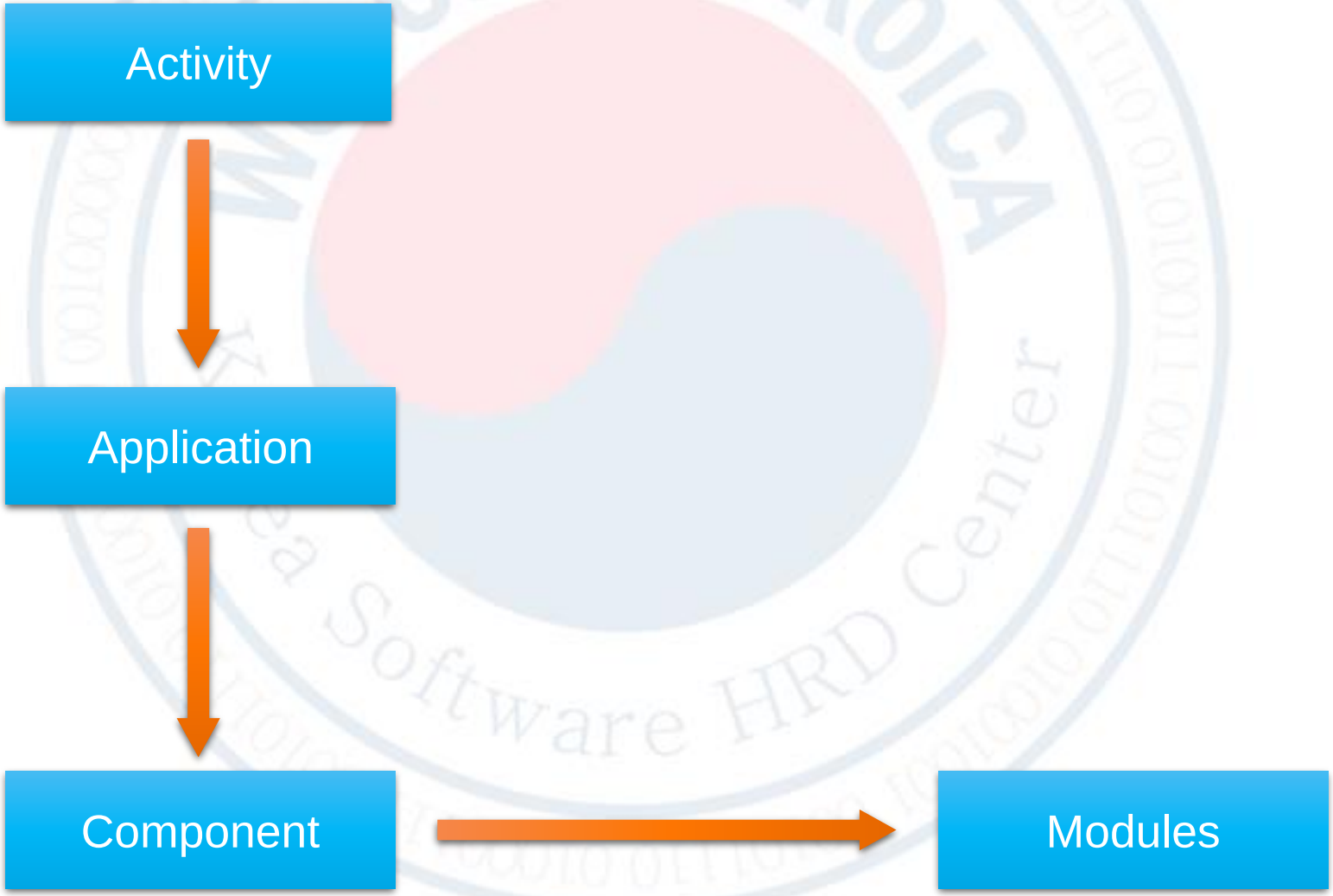
Constructor Injection



Activity


Activity
Application


Activity
Application
Component


Activity
Application
Component
Modules


Activity
Application
Component
Modules
[Modules 1]
[Modules 2]

Module Initialization

```
public class AndroidApplication extends Application {
    private ApplicationComponent applicationComponent;

    @Override public void onCreate() {
        super.onCreate();
        this.initializeInjector();
    }

    private void initializeInjector() {
        this.applicationComponent = DaggerApplicationComponent.builder()
                .presenterModule(new PresenterModule(this))
                .dataModule(new DataModule())
                .build();
    }

    public ApplicationComponent getApplicationComponent() { return this.applicationComponent; }
}
```

this notation is optional