SOLID Principles

S — Single Responsibility

A class should have only one reason to change.

O — Open/Closed Principle

- Open for extension
- Closed for modification

Android Ex:

✔ Relying on abstractions like `View`, `TextView`, `ViewGroup` to create custom views.

L — Liskov Substitution Principle

"Objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program."

Android Ex:

✔ `when` inject a `MockCustomerRepository`, The calling code just knows that it's working with an instance of `CustomerRepository`,

✔ `because it` implements the `CustomerRepository (a subtype of it)`

I — Interface Segregation Principle

"Make fine grained interfaces that are client-specific."

Android Ex:

```
public interface OnClickListener {
    void onClick(View v);
    void onLongClick(View v);
    void onTouch(View v, MotionEvent event);
}
```

X

✔ Avoid generic polluted interfaces

Client needs one interface and doesn't need the others

D — Dependency Inversion Principle

- High-level modules should not depend on low-level modules. Both should depend on abstractions.
- Abstractions should not depend on details. Details should depend on abstractions.

Android Ex:

To create a BusinessLayer object,

↓

create an object that implements IDataLayer

↓

It does not care who implements it, it just needs an object that implements that interface.

Dependency Injection

Dalia