一：应用开发原则：

1. 遵循面向对象的Solid原则。
2. 视图，数据，逻辑分离。

（solid原则是什么？）：

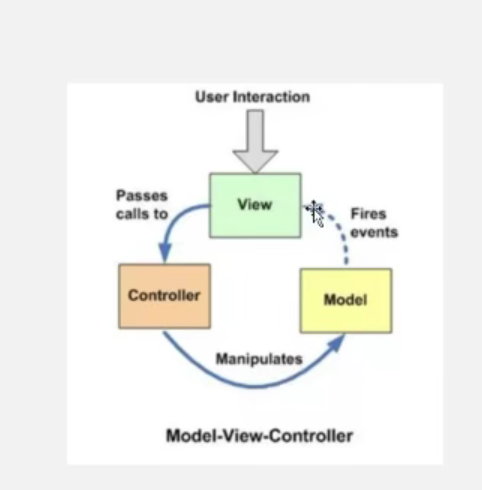
1. 单一职责原则（SRP）：表明一个类有且只有一个职责。一个类就像容器一样，它能添加任意数量的属性、方法等。
2. 开放封闭原则（OCP）：一个类应该对扩展开放，对修改关闭。这意味一旦创建了一个类并且应用程序的其他部分开始使用它，就不应该修改它。
3. 里氏替换原则（LSP）：派生的子类应该是可替换基类的，也就是说任何基类可以出现的地方，子类一定可以出现。值得注意的是，当通过继承实现多态行为时，如果派生类没有遵守LSP，可能会让系统引发异常。
4. 接口隔离原则（ISP）：表明类不应该被迫依赖他们不使用的方法，也就是说一个接口应该拥有尽可能少的行为，它是精简的，也是单一的。
5. 依赖倒置原则（DIP）：表明高层模块不应该依赖低层模块，相反，他们应该依赖抽象类或者接口。这意味着不应该在高层模块中使用具体的低层模块。

二：MVX开发模式的区别（见下图:）



用mvvm别用mvp了

MVC：View层要接收用户点击事件的请求然后view把事件给Controller（控制层）然后控制层经过处理之后给Model然后model做完业务逻辑操作之后再通知view层进行更新。（是单向通信方式）



MVC在android里面的问题：

MVC在Android里面出现了许多问题：Android中的Activity既要做UI还要做Model更加还要做Controller。



但是通常情况下，不可避免的会把Model层的职责交给了Activity；并且Activity也会不由自主的去承担View的职责。所以在传统开发模式MVC里面Activity这里代码会越来越多Activy这个类的职责也会不断的增加，以致于变得庞大臃肿。

三：MVC和MVP之间的对比

举个 例子：在MVC里面Activity既要做管理，还要做厨师还要去给客人点菜当服务员；但是MVP里Presneter只要做管理，model只做厨师，View（Activity）只给客户点菜。

MVP解决了MVC在android中的问题

在MVP里面Activity不做Controller（控制层）而是只做View V层，把Model变成了业务数据逻辑层 M层，P层去管理控制V层与M层的交互处理处理。

MVP的优点：1.View层完全不用关心Model层，相反Model层完全不用关心View层，而是用Presneter去管理他们两者。

2.如果界面UI出问题直接找View层，如果业务逻辑出问题直接找Model层。P层只做交互操作。

MVP：View收到用户的操作，View再把用户的操作交给Presneter于是Presneter做完校验操作之后把指令再给Model层，Model层把结果信息再给P层，P层再 解析完结果信息之后再给V层，V层再更新UI。（是双向通信方式）

