# 面向对象的三大特征：

# 封装

## 从数据角度讲：

一个类封装多个数据。

将一个基础变量复合为自定义类型。

不但可以准确的描述实物，还可以提现该事物的行为。

## 从行为角度讲：

一个类封装多个行为

向类外提供必要的功能，隐藏实现的细节（用双下划线），用属性把数据保护起来。

使用者不必操心实现的过程。

## 从设计角度讲：

1. 分而治之

将一个大的需求分解为许多类，让每个类处理一个独立的功能。

优点：便于分工，便于复用，可扩展性强。

1. 封装变化

需求可能会变化的功能单独封装，避免影响其他类。

1. 高内聚

类中各个方法都在完成一项任务

1. 低耦合

类和类的关联性与依赖度很低，让一个类变化，尽少影响其他类。

## 大白话

## 从数据角度讲：

用一个类包装多个变量（数据），还可以定义方法，操作数据。

## 从行为角度讲： 不同程序员开发不同的功能，使用时，只需要考虑调用，不需要考虑内部实现。

## 从设计角度讲：

分而治之 – 拿到需求后，分析解决问题的类，让每个类负责处理一件事。

封装变化 – 找出可能变化的功能，使用类单独定义。

高内聚 – 类中所有成员负责一件事

低耦合 – 类与类之间的关系尽可能少，一个类的变化不影响其他类。

# 继承

重用现有类的功能和概念，并在此基础上进行扩展。

优点：代码复用率高

统一概念(人使用交通工具，而交通工具隔离了火车/汽车等具体类的行为。

缺点：耦合度高

切换不灵活（员工转岗）

解决方案：使用关联关系代替继承关系。（员工有一个岗位）

适用性：多个类在概念上是一致的，且需要统一的处理（统计调用他们的某个行为）

备注：建议不要把代码复用作为继承的适用性。

# 多态

### 定义：

调用父类的一个方法，在不同的子类对象中，有不同的实现效果。

作用：增强程序性扩展性（子类增加，不影响调用者）。

重写：子类和父类有相同的名称方法，在调用父类时，执行子类。

### 作用：

1. 天线子类的个性化。
2. 增强程序的扩展性，提现开闭原则。

### 重写

子类实现父类名称中相同（名称、参数）的方法，在调用该方法时，实际执行的是子类方法。

# 设计原则：

## 开闭原则（目标）：

对扩展开放，对修改关闭。

允许增加新功能，不允许修改客户端（使用者）代码。

## 类的单一职责（类的定义）：

一个类有且只有一个改变的原因。

外界一个需求的变化，内部一个改变的类。

变更引起的风险降低，变更是必不可少的，如果类的单一职责做得好一个类修改只有相应的实现类有影响，对其他类没有影响，这对系统的可扩展性、维护性都有非常大的帮助。

## 依赖倒置（找爹）

客户端代码尽量依赖（使用）抽象（父）的组件。

抽象是稳定的，实现是多变的。

## 组合复用原则（少用继承）

如果仅仅为了代码复用优先选择组合关系，而非继承关系。

组合的耦合度低于继承，灵活度高于继承。

## 里氏替换（重写注意事项）

父类出现的地方可以被子类替换，在替换后依然保持原有功能。

--从内存角度解释：父类（成员少） 子类（成员多）

--子类在重写父类方法时，尽量选择扩展重写（先调用父类同名方法），不要改变原有功能。

## 迪米特法则（低耦合）

原话：不要和陌生人说话。

类与类交互时，在满足功能的基础上，传递的数据量越少越好。