面向对象三大基本特征，六大原则

面向对象三大基本特征：封装，继承，多态。

封装即指将某事物的属性和行为包装到对象中，这个对象只对外公布需要公开的属性和行为，而这个公布也是可以有选择的公布给其他对象。

继承是子对象可以继承父对象的属性和行为，亦即父对象拥有的属性和行为，其子对象也拥有这些属性和行为。类似自然界的遗传。

多态指允许不同类的对象对同一消息作出相应。比如同样的加法，把两个时间相加和把两个整数相加肯定完全不同，很好的解决了方法同名的问题。

１. 三大特征

a). 封装

设计角度：分而治之 封装变化 高内聚 低耦合

b). 继承

重用现有类的功能与概念，并在此基础上扩展

优点：代码复用

统一概念(人使用交通工具，而交通工具隔离了火车/汽车等具体类的变化)

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，且需要统一处(统一调用他们的某个成员)。

　 备注：＊＊建议不要将代码复用作为继承的适用性＊＊

c). 多态

同一个方法，在不同的子类中，有不同的表现。

继承的存在(继承是多态的基础,没有继承就没有多态).

子类重写父类的方法(多态下调用子类重写的方法).

作用：增强程序扩展性(子类增加，不影响调用者)

重写：子类具有与父类相同名称的方法，在调用父类时，执行子类的方法

运算符重载(重写)

面向对象设计原则：

六大原则之间并不是相互孤立的，彼此间存在着一定关联，一个可以是另一个原则的加强或是基础。违反其中的某一个，可能同时违反了其余的原则。

开闭原则是面向对象的可复用设计的基石。其他设计原则是实现开闭原则的手段和工具。

**开闭原则（目标）**：对扩展开放，对修改关闭。

稳定性。开闭原则要求扩展功能不修改原来的代码，这可以让软件系统在变化中保持稳定。

扩展性。开闭原则要求对扩展开放，通过扩展提供新的或改变原有的功能，让软件系统具有灵活的可扩展性。  
遵循开闭原则的系统设计，可以让软件系统可复用，并且易于维护。

提高系统的可复用性和可维护性。

**类的单一职责（类的定义）**：永远不要让一个类存在多个改变他的理由。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。体现了内聚性，当需求变化时，该变化会反映为类的职责的变化；如果一个类承担了多于一个的职责，那么引起它变化的原因会就有多个。

**依赖倒置（依赖抽象）**：客户端代码，尽量使用抽象的组件。抽象的是稳定的，实现是多变的。

依赖倒置原则的优点：可以减少类间的耦合性、提高系统稳定性，提高代码可读性和可维护性，可降低修改程序所造成的风险。

**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。

使用继承关系的条件：

组合是指子类是父类的一个特殊种类，而不是父类的一个角色，也就是区分"Has-A"和"Is-A"。只有"Is-A"关系才符合继承关系，"Has-A"关系应当用组合来描述。

永远不会出现需要将子类换成另外一个类的子类的情况。如果不能肯定将来是否会变成另外一个子类的话，就不要使用继承。

子类具有扩展父类的责任，而不是具有置换掉父类的责任。如果一个子类有需要大量的置换掉父类的行为，那么这个类就不应该是这个父类的子类。

**里氏替换原则（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，子类在重写父类方法时，尽量选择扩展重写，不改变原有功能

里式替换原则为我们是否应该使用继承提供了判断的依据，不再是简单地根据两者之间是否有相同之处来说使用继承。如果一个字类的对象可能会在父类出现的地方出现运行错误，则该子类不应该从该父类继承，或者说，应该重新设计它们之间的关系。

（PS: 子类实现了父类中相同的方法，在调用父类的方法时，实际执行子类的方法）

**迪米特法则：**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（低耦合）

（原话：不要和陌生人说话）

迪米特原则在于降低类之间的耦合。由于每个类尽量减少对其他类的依赖，因此，很容易使得系统的功能模块功能独立，相互之间不存在（或很少有）依赖关系。