抽象(abstract)  
忽略一个主题中与当前目标无关的那些方面，以便更充分地注意与当前目标有关的方面。抽象并不打算了解全部问题，而只是选择其中的一部分，暂时不用关注细节。

封装(Encapsulation)  
封装是面向对象的特征之一，是对象和类概念的主要特性。封装是把过程和数据包围起来，对数据的访问只能通过指定的方式。  
在定义一个对象的特性的时候，有必要决定这些特性的可见性，即哪些特性对外部是可见的，哪些特性用于表示内部状态。通常，应禁止直接访问一个对象中数据的实际表示，而应通过操作接口来访问，这称为信息隐藏。信息隐藏是用户对封装性的认识，封装则为信息隐藏提供支持。  
封装保证了模块具有较好的独立性，使得程序维护修改较为容易。对应用程序的修改仅限于类的内部，因而可以将应用程序修改带来的影响减少到最低限度。

继承(inheritance)  
继承是一种联结类的层次模型，并且允许和支持类的重用，它提供了一种明确表述共性的方法。  
新类继承了原始类后,新类就继承了原始类的特性，新类称为原始类的派生类(子类)，而原始类称为新类的基类(父类)。  
派生类(子类)可以从它的基类(父类)那里继承方法和实例变量，并且派生类(子类)中可以修改或增加新的方法使之更适合特殊的需要

多态(polymorphism)  
多态性是指允许不同类的对象对同一消息作出响应。  
多态性语言具有灵活、抽象、行为共享、代码共享的优势，很好的解决了应用程序函数同名问题。  
相同类域的不同对象,调用相同方法,表现出不同的结果

方法的调用:  
在类中定义了方法,这个方法中的代码并不会执行,当这个方法被调用的时候,方法中的代码才会被一行一行顺序执行。  
1)非静态方法  
没有使用static修饰符修饰的方法,就是非静态方法.  
调用这种方法的时候,是"一定"要使用对象的。因为非静态方法是属于对象的。(非静态属性也是一样的)  
例如:public class Student{  
 public void say(){}  
 }  
 main:  
 Student s = new Student();  
 s.say();  
2)静态方法  
使用static修饰符修饰的方法,就是静态方法.  
调用这种方法的时候,"可以"使用对象调用,也"可以"使用类来调用,但是推荐使用类进行调用,因为静态方法是属于类的。(静态属性也是一样的)  
例如:public class Student{  
 public static void say(){}  
 }