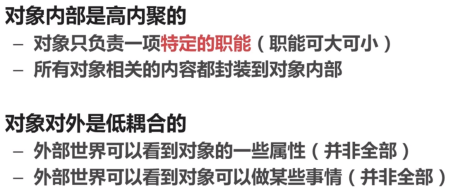
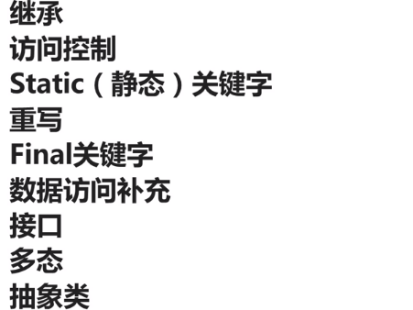
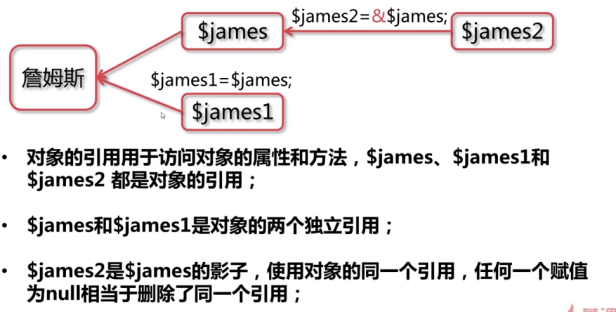
# 目标



# \*&



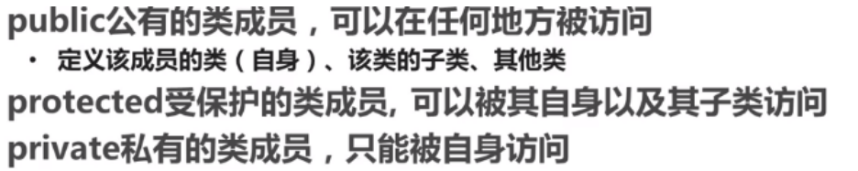
# 继承

子类对象可以访问父类

父类对象不可以访问子类

一次只可以拼接一个类，但是可以在父类中再继承加上类

# 权限



**//外面里面只能使用public方法  
//protected与private都是只能在类的里面进行访问  
//protected是在自身与子类中可以进行访问  
//private是在自身类中可以访问**

# static

例子是如果一个对象进行修改了属性值，另外一个对象不会被修改，也就是属性值没有被保存，所以应该使用static，静态类对象设置被保存之后，

这个类多有对应的对象都发生了变化。

只要使用static定义了，

所有依据此类新建立的对象就是引用传值了，

变化是一致的齐动的。

**静态属性与方法相当于引用传值**

**普通的就是值传递**

**静态属性就是地址 对象就相当于与变量**

**// \* 1. 静态属性用于保存类的公有数据,使得类的对象static属性一致  
// \* 2. 静态方法里面只能访问静态属性  
// \* 3. 静态成员不需要实例化对象就可以访问  
// \* 4. 类内部，可以通过self或者static关键字访问自身的静态成员  
// \* 5. 可以通过parent关键字访问父类的静态成员  
// \* 6. 可以通过类名称在外部访问类的静态成员**

# 重写

在类继承的时候，子类中进行命名相同的方法，就会导致方法的重写

# Final

**类进行final 会导致不能被继承**

**方法进行final 会导致不能被重写 但是可以被使用**

**\* 重写和Final  
\* 1. 子类中编写跟父类完全一致的方法可以完成对父类方法的重写  
\* 2. 对于不想被任何类继承的类可以在class之前添加final关键字  
\* 3. 对于不想被子类重写（overwrite, 修改）的方法，可以在方法定义前面添加final关键字**

# 数据访问

**\* 1. parent关键字可以用于调用父类被重写的类成员  
\* 2. self关键字可以用于访问类自身的成员方法，也可以用于访问自身的静态成员和类常量；不能用于访问类自身的属性；访问类常量时不用在常量名称前面加$符号  
\* 3. static关键字用于访问类自身定义的静态成员，访问静态属性时需要在属性名前面添加$符号**

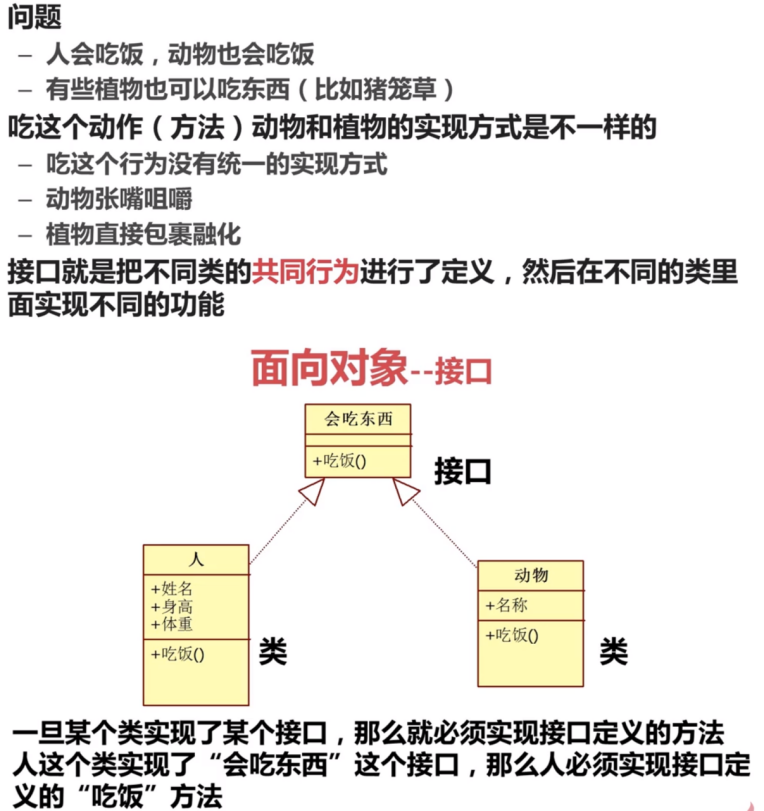
# 接口

## 说明

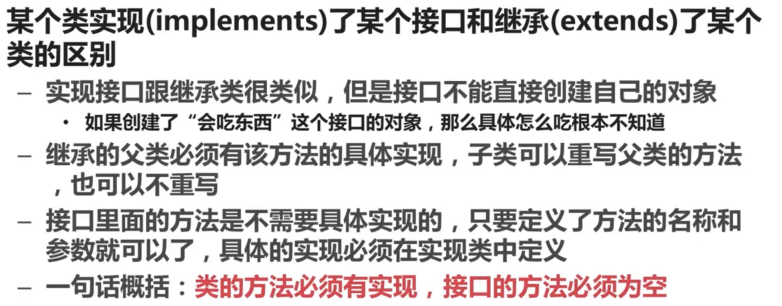
因为不同的类生成的对象实现接口定义的方法都有自己的一套规则

所以就有了接口，

接口定义了类要去实现的方法。



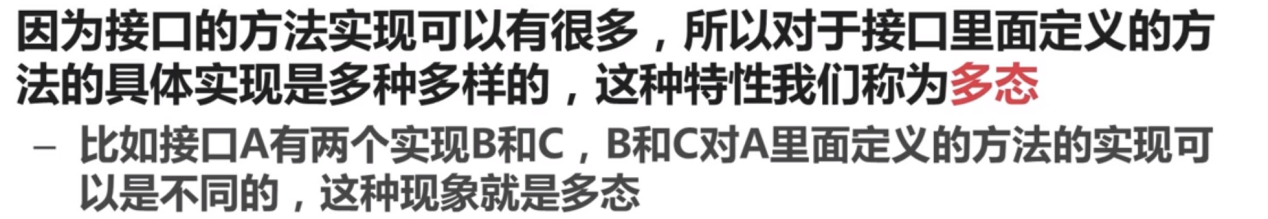
## 接口继承区别



# 多态

使用同一个接口，不同的类实现的接口的方法名字一样，

但是内容不一样，就是多态

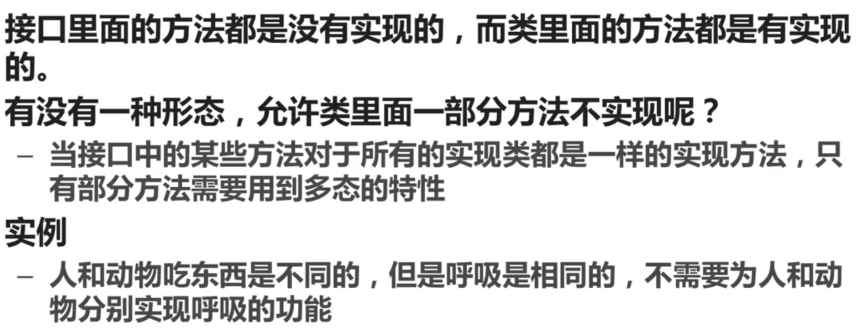


# 抽象类

继承是全部不用实现

接口是全部要实现

抽象类是abstract的要实现 其他的不用实现



**\* 抽象类  
\* 1. 抽象类允许类里面的部分方法暂时没有具体实现，这些方法我们成为抽象方法  
\* 2. 一旦类里面有抽象方法，这个类就必须是抽象类  
\* 3. 抽象类跟接口一样，不能直接实例化为对象**

# construct destruct

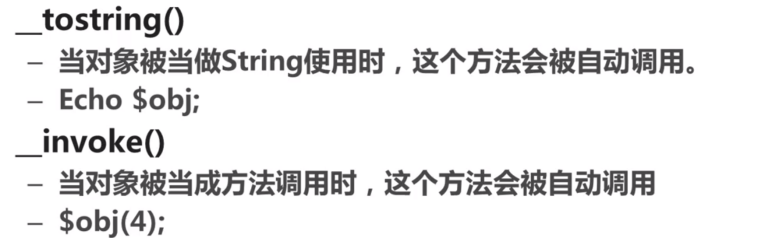
实例化对象的时候会调用construct

对象设置为空的时候或者程序结束的时候会调用destruct

# tostring invoke

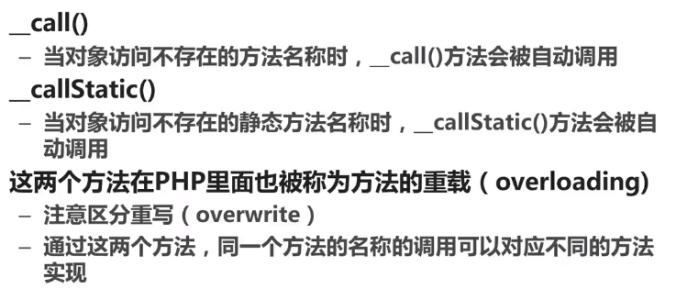
**// 把对象作为字符串输出时调用的方法tostring**

**// 把对象作为方法输出时调用的方法invoke**



# call callStatic

**对象调用方法时候，找不到要找的方法的时候，会进行\_\_call \_\_callStatic**



# get set isset unset

**当访问属性不存在或者不可访问的时候 进行获取调用\_\_get()**

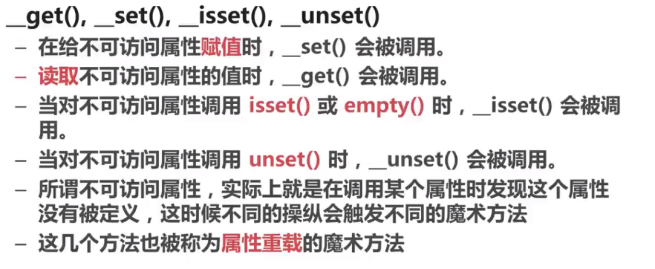
**当设置属性不存在或者不可访问的时候 进行获取调用\_\_set()**

**当使用isset()或者empty()访问属性不存在或者不可访问的时候**

**进行获取调用\_\_isset()**

**当使用unset()设置属性不存在或者不可访问的时候**

**进行获取调用\_\_unset()**



# Clone

**\_\_clone在clone的时候会被调用 是用来复制对象的**

