**PHP中对象的继承**

**★继承的基础内容**

**定义:继承extends,是指子类通过继承可以访问父类的成员。**

* **基础语法**

PHP语法: class 子类 extends 父类{}

中文字义: 创建类 子类对象 继承 父类对象{子类内容}

//父类(基类)

class Human{}

//子类(派生类)

class Man extends Human{}

* **继承效果**

继承效果: 父类非私立(private)方法和属性会继承给子类,可以在子类直接使用;

//父类(基类)

class Human{

public function showName( ){

echo \_ CLASS\_\_\_ ;

}

//子类(派生类)

class Man extends Human{ }

//实例化子类对象:如果实例化父类对象Human,那么与子类和继承毫无关系

$m = new Man();

$m- >showName();

//访问继承自父类的方法

* **小结**

1.继承是利用extends进行子类和父类的关联

2.继承利用extends关键字实现

3.继承是指实现继承关系后，子类对象可以访问父类被继承的成员。而父类对 象不可以访问子类成员

☆注意:构造方法和析构方法也可以被子类继承，此时需要注意子类对象实例化时对应的父类构造方法的参数

//父类 //子类继承

class Human{ class Man extends Human{]}

private $money; //子类实例化:自身是空类，没有指定构造方法

public function construct( $money){ $m = new Man();//错误:缺少参数，因为会自动调用父类构造方法

$this->money = $money; $m = new Man(100);//正确

}

}

**★有限继承**

**定义:** 有限继承，指子类在继承父类的成员的时候，并非继承所有内容，而是继承并使用父 类部分内容。

1.继承内容: PHP中继承是子类继承父类所有的公有成员、受保护成员和私有属性,不能继承父类的私有方法

* **不同定义的继承效果**

class Human{ //定义父类 // 不能继承私有方法和变量!!

public $公有可以继承;

protected $受保护可以继承; 如果你想要让子类访问父类私有属性或对象

private $私有不可以继承; ↓↓↓↓↓↓↓↓↓↓↓↓

1.在父类创建一个访问自己私有变量的方法(方法必须是共有或受保护这样才可以给子类使用)

}

//定义子类 extends表示继承父类内容

class Man extends Human{}

//实例化 子类 能调用父类非私有内容

1. 在子类创建一个方法,该方法内容需要有：

子类方法→访问→父类方法→访问→父类私有属性

注意:虽然子类可以通过以上方式来实现访问父类的私有成员(包括私有方法)，但是从设计的意 义上讲，私有就是不允许外部访问，所以父类通常不会提供对外的访问接口，以上案例 只是为了知识说明。

4.静态成员(类常量)也遵循继承规则(PHP继承本质是对象) ,只是访问方式是由类进行访问

* **小结**

1. 继承是有限继承，理论上是用来继承父类允许被继承的部分，即使用public或者protected修饰的成员
2. 因为对象的属性是保存在对象内存空间，所以父类的私有属性也会继承
3. 父类私有成员本质不允许被子类访问，但是可以通过父类开放接口实现(一般不会这么操作)
4. 静态成员也可以遵循继承规则
5. 构造方法也可以被继承，因此在实例化子类对象的时候，要考虑到父类构造方法所使用到的参数问题

**思考:**子类可以从父类进行继承，那如果子类与父类的成员重名了怎么办?

**引入:**首先，类中定义的成员是允许出现与其他类同名的，但是这个过程如果出现了继承，那么重名就存在(

重写的问题。

**继承中的Ovrride重写**

**☆重写的定义**

* 子类中含有父类相同函数,会使用子类的函数和变量而不会继承!
* 重写之后子类只能访问子类的属性和方法（因为重新定义的子类把父类同名的属性或方法覆盖了）

★ 注意！重写只会覆盖继承的属性和方法(私有的属性因为没有继承不会重写,可以用父类的方法访问)

**★重写要求**

* 重写要求1:重写的属性必须同级以以下,但不能比继承属性还高(私有>受保护>共有)
* 重写要求2:重写方法的时候,行参需要与父类保持一致(形参数量,形参强类型时的类型)

**★访问被重写方法**

* 重写是指子类拥有特殊的情况，一般是需要在父类的基础上进行扩展，此时如果想要继续保证父类被重写的方法继续执行(默认永远只访问子类重写的新方法)，需要在子类重写方法的时候使用 parent关键字

//子类继承

class Man extends Human{

//重写

public function

show(){

//强制调用父类被重写方法

parent: :show();

//扩展业务逻辑

echo \_\_ CLASS\_ ;

}

}

//父类

class Human{

protected function show(){

echo \_\_ CLASS\_\_ ;

}

}

**★** 注意: parent不能访问父类的属性，可以访问静态属性、静态方法、类常量和普通方法

* **小结**

1. 重写override是一种在子类中定义父类同名成员的操作
2. 公有、受保护的属性重写是直接覆盖父类成员，私有属性不会被覆盖;公有、收保护的方法会被重写，但是私有方法不会被重写(私有方法本质没有被继承)
3. 重写的要求

* 子类控制权不能高于父类控制权
* PHP7中要求被重写的方法必须与父类保持参数-致(数量和类型)

1. 方法被重写后，访问调用的都是子类方法，如果想要访问父类方法，可以通过在子类法中使用parent关键字来强制访问父类方法
2. parent不能用于访问父类的属性(静态属性可以)

**思考:** PHP继承与其他面向对象语言的继承有不同吗?

**引入:** PHP因为是自面向过程发展而来，所以并非完全使用面向对象的思想，因此PHP的继承有一些自己的特 点。

**PHP继承特点**

**▷缘由**

定义: PHP继承与其他*纯面向对象*(从设计之初就完全由面向对象思维支配)编程语言是有一些不一样的。

**☆PHP特点**

* PHP中继承只能单继承:即子类只有一个父类(有些语言支持多继承)

class Man{}

class Woman{}

class Ladyboy extends Man, Woman{} //PHP中错误， 不允许继承多个父类

* PHP若想继承多个类，可以使用链式继承

**思考:** self关键字是用于在类内部代替类的，代替的是当前方法所在的类本身，随着继承的实现，如果子类在访问父类的方法的时候，self到底代替的是谁呢?

**引入:** self是一 种静态绑定，换言之就是当类进行编译的时候self已经明确绑定了类名，因此不论多少继承，也不管是子类还是父类自己来进行访问，self代表都是当前类。如果想要选择性的来支持来访者，就需要使用静态延迟绑定

//Ladyboy包含了

Man和Woman类中所有可继承的成员

class Man{}

class Woman extends Man{}

class Ladyboy extends Woman{}

* PHP中继承只有私有方法不能被继承
* PHP允许子类继承父类的构造方法和析构方法

**★静态延迟绑定(父类子类的静态属性访问)**

**关键字** static(静态) ≈ this(普通) //谁调用就指向谁

**关键字** self却是指向原来的方法 //谁创建就指向谁

▷继承的目的是为了使用父类的公共代码

**▶最终类Final(不能再继承)**

关键字 final class 类名 //表示不能继承该父类

关键字 public final function Name(){} // 表示不能重写该方法

* 实例：

//父类

class Human{

public function show(){} //普通方法

public final function walk(){} //最终方法

}

//子类

class Man extends Human{

//重写

public function show(){} //没问题 可以重写

public function walk(){} //致命错误:不能重写父类中的最终方法

}

**★小结**

1. final关键字修饰的类表示无法被继承

2. final关键字还可以修饰方法，表示方法不能子类重写(通常类不会使用final关键字)

3. final修饰类表示不希望类再出现子类，可以很好保护类的内部结构不被暴露

4. final修饰方法表示不希望方法被修改，可以在一个更高的维度来保证同类事务的共同表现

**思考:**在一个项目中，如果架构师在前面给定义了-些父类，而且希望所有的子类都遵循父类的 设置，有没有什么办法可以限制呢?

**引入:**理论上来讲没有系统性的办法能够严格去规范程序员必须怎么实现，但是大家如果都遵循 规则的话，那么可以利用父类来强制规范子类必须完成要做的事情，这个时候需要用到抽 象类。

★**抽象类**Abstract

**定义:**抽象类，使用abstract关键字修饰的类，表示该类只能被继承，不能被实例化

注意！抽象方法不能私有化

注意：没有方法体

* 使用方法：

关键字 abstract class 类名{} //只能被继承

关键字 abstract public function Name() //没有内容,定义必须重写

1. 抽象类无法被实例化
2. 抽象类只能被继承
3. abstract关键字还可以用来修饰方法(抽象方法) , abstract修饰的方法不能有方法体，而且有抽象方法的类必 须声明为抽象类
4. 抽象方法因为要被子类继承实现，所以不能使用private修饰(私有方法不会被继承)
5. 子类继承抽象类后，如果抽象类中有抽象方法，那么子类必须选择自己成为抽象类或者实现抽象方法(所有 抽象方法)

**★小结**

1.使用abstract修饰的类叫做抽象类

2.抽象类不可以被实例化，只能被继承

3.因为抽象类无法被实例化，因此私有成员在类中没有实质意义(还需要额外提供受 保护或者公有方法来实现访问)

4.抽象类的目的是用来规范子类(通常必配抽象方法)

5. abstract还可以修饰方法,称之为抽象方法:抽象方法所在的类必须是抽象类，抽 象方法不能有方法体

6.有抽象方法的抽象类被继承时子类要么自身是抽象类，要么实现所有抽象方法

7.抽象类这种结构管理,需要耗费较多的架构和初始代码，通常在比较大型或者规 范的项目中才会使用

**思考:**在大型项目中，如果想要规定某些类必须有某些方法，我们可以使用抽象类来进行视范吗?

**引入:**抽象类的目的就是用来规范一些共性子类的结构，必须实现某些指定的方法的。在PHP中， 还有一种专门用来规范类内部实现方法的结构，叫做接口(interface)。

**▶接口**interface

**定义:**接口,使用interface关键字定义,与类类似,专门用来规范一些共性类必须实现的方法

* 使用方法

new Human{}

致命错误，接口不能实例化

interface 类名(接口名){} //创建接口

接口实现:接口是用来规范类必须完成的事情，所以接口只能被类实现: implements

class 类名 implements 类名(接口名){}

**★接口成员**

接口中只能定义公有抽象方法和接口常量

* 实例(创建接口):

interface Human{

//接口常量

const name =‘我是常量’; //通常来说常量用的很少

//接口抽象方法

public function eat(); //注意！ 在接口中,方法默认就是自带abstract关键字

//错误示例

public function go(){} //错误: 接口中的方法必须为抽象

public $age; //错误:接口中不能有属性

public static $count = 0; //错误:接口中不能有静态属性(成员属性)

protected function walk(); //错误:接口方法必须为公有抽象方法

}

* 实例(接口实现):

Class Man inplements Human{

Const name = “我想重写” //错误:不能重写接口的常量

//实现接口方法

public function eat(){}; //接口有抽象方法必须继承

}

▶接口的继承

* 接口可以继承接口: extends, 而且接口可以多继承接口

Interface Animal{

public function eat();

}

interface Human{

public function walk();

}

//单继承

interface Man extends Human{ }

//多继承

interface Ape extends Human, Animal{ }

★小结

1.接口是使用interface关键字定义，与类结构类似

2.接口只能被类实现

3.接口成员只有两种:接口常量和公有抽象方法(没有方法体:不需要abstract关键字)

4.类实现接口的时候，必须实现接口中所有的抽象方法(或者抽象类实现)

5.接口中的成员不能被实现接口的类进行重写(接口常量)和权限更高(受保护或者私有化接口方 法)

6.接口可以继承接口，而且可以多继承

7.接口是在更大型的项目中，为了保证底层的实现而设定的规范，通常是抽象类实现接口，增加 必要成员，然后让实际业务类去继承抽象类。

**思考:** PHP中继承是单继承，如果某个类有成员要被其他类使用，就需要成为其他类的父类才行，这样可能

会导致继承链很长，合适吗?

**引入:**从继承的角度出发,继承链可以解决问题，但是的确效率会打折扣。同时，如果某些功能是共性使

用，但是并不符合继承条件(不属于同一类事务)，那么使用继承也有所违背面向对象规则，此时可以使用

PHP提供的另外一种代码复用技术trait

**★**trait**代码复用**

定义:trait是为类似PHP的单继承语言而准备的一种代码复用机制。trait可以使得单继承语言拜托为了复用而不得不继承的尴尬，让面向对象变得更加纯粹。

* 实例创建： trait是类似class的关键字

trait Eat{ //trait内部可以拥有一个类能拥有成员属性(包含静态),成员方法(包含静态),但不能有类常量

public $time;

protected Show; //允许定义，但是实际不用

private $info;

public function showTime(){

echo $this->time;

}

protected function showHow(){ //允许定义， 但是实际不用

echo $this->how;

}

const PI = 3.14; //错误: trait中不能有常量

}

* 实例使用: use A,B,C

**★注意！**trait是用来实现代码的复用的，不可以被实例化也不可以被继承(不是类)

class Man{ //使用

//引入trait公共代码

use Eat; //可以使用多个

}

trait Eat{ //创建

public function show(){

echo “eat”；

}

}

**思考:**如果同一个Trait有相同名字的属性或方法会发生什么？怎么解决这个问题。

**引入:**trait有相同名字的方法或属性就要明确表明使用谁,谁代替谁,这时候就要用到insteadof关键字

★insteadof**实例**

trait t1{

public function eat(){

echo 't1,eat';

}

}

首先创建两个trait

里面有相同的eat方法

trait t2{

public function eat(){

echo 't1,eat';

}

}

//解决方案:明确替代(确定使用哪一个)

class Person{

use t1,t2{ //花括号

//t2的eat代替t1的

t2: :eat insteadof t1;

}

}

$p = new Person();

$p->eat(); //使用的是t2的方法 t2,eat

//解决方案:先替换后别名

class Animal{

use t1,t2{

t2::eat insteadof t1; / /明确使用t2中的eat方法

t1: :eat as eat1; //t1中的eat方法改名叫eat\_t1

}

}

$p = new Animal();

$p->eat(); // t2,eat $p->eat\_t1(); //t1,eat

★trait同名覆盖问题:

如果类中有与引入的trait同名成员，会有不同处理

* 属性:不允许重名，即类中不允许定义与trait中同名的成员属性(静态属性也-样)
* 方法:类方法覆盖trait方法

★trait继承覆盖问题:

* 如果类中在使用trait的同时，也是继承自父类,而trait中与父类中有同名方法， 那么trait中将覆盖父类同名方法;如果要访问父类方法，可以在trait同名方法中使用parent关键字访问父类同名方法
* 简单来说：就是如果父类和trait和自己都有相同方法那么: 自己>trait>父类

★注意！

* trait自己不能访问，只是用来给其他类提供代码复用的，因此允许类在使用trait时更高里面方法的访问控制权:在as之后，使用目标访问修饰限定符（在别名时重新定义访问权限 可升降）

use eat{

show as protected p\_show;

}

重定义访问权限(访问修饰符)必须别名↓↓

虽然没有同名show,但是系统认为show已经存在，所以必须别名

▷trait中的抽象方法

* + - 知道有就行..基本不用

★小结

1. trait是一种类似class结构关键字, trait不能被实例化,可以拥有所有类成员结构(类常是不行)

2. trait是用来实现代码复用的，为其他类提供公共代码(方法) , 其他类如果使用trait用use关 键字引入

3.在类中use具体trait就相当于将trait内的所有代码在类中写了- -遍

4. -个类可以使用多个trait,但是要注意同名问题

* 同名方法可以使用insteadof来实现替代:一个trait中的同名方法替代另外-一个,类就访问替代的那个
* 同名方法可以在被替代之后使用as制作方法别名:类就可以拥有两个方法

5.类中在引入trait后，要注意与trait中的同名成员问题

* 同名属性:不予许(类中不允许)
* 同名方法:允许，类中的方法会覆盖trait中的方法

6.如果类在使用trait的同时也继承了其他类，那么trait中出现的同名方法会覆盖基类的同名方法 自己>trait>父类

7.类在使用trait时可以修改trait方法的控制级别:更严或者更宽松都可以，注意修改控制级别时 使用的是别名机制，一定需要改成别名: [trait名]方法名 as访问修饰限定符别名(原来trait中的 方法依然可以正常使用);

8. trait中可以使用抽象方法，那么使用该trait的类就必须本身 为抽象类或者将抽象方法实现

9. trait使用机制

* 有公共代码要实现(方法) ,而这些方法可能在很多类中会用到
* 公共代码不是属于某- 类事务特有，而是很多事务都有(不符合继承)

思考: PHP对象是复合数据类型，如果直接echo输出对象会报错的，这样的报错会给用户体验不好，有没有

办法能够解决这类问题呢?

引入:虽然上述问题出现是因为程序员的误操作，但是的确有可能会出现这样的问题，尤其是当我们的程序

是给一些外部调用时。所以，面向对象关于很多对象可能出现的错误都提供了-种容错机制，这种机制 叫做**重载**