●案例演示：子类只能继承父类所有非私有的成员(成员方法和成员变量)

雍正

属性：姓名，长相，玉玺

行为：江山，美人

乾隆

属性：姓名，长相

行为：江山，美人

```
public class Demo4 {
    public static void main(String[] args) {
        YongZheng y = new YongZheng();
        y.yuXi = "玉玺";
        y.name = "爱新觉罗.胤禛";
        y.shape ="瘦瘦的";
        y.jiangShan();
        System.out.println("--------");
        QiangLong q = new QiangLong();
        q.name = "爱新觉罗.弘历";
        q.shape = "胖胖的";
        q.yuXi = "玉玺";
        q.jiangShan();//在这里留下一个问题？皇帝私有的妃子被取到了，怎么解决？
    }
}
/*
  雍正
    属性：姓名，长相，玉玺
    行为：江山，美人
乾隆
    属性：姓名，长相
    行为：江山，美人
 */
class Person{
    String name;
    String shape;//外形
}
class YongZheng extends Person{
    String yuXi;
    public void jiangShan(){
        System.out.println(name+"打下了江山"+shape+yuXi);
        feiZi();//私有的用公有的方法调用
    }
    private void feiZi(){
        System.out.println("妃子");
    }
}
class QiangLong extends YongZheng{
    private void feiZi(){
        System.out.println("你还记得大明湖畔的夏雨荷吗？");
```

```
    }
}
```

B:什么时候使用继承

继承其实体现的是一种关系："is a"。 Person Student Teacher 水果 苹果 香蕉 橘子

采用假设法。 如果有两个类 A,B。只有他们符合 A 是 B 的一种，或者 B 是 A 的一种，就可以考虑使用继承。

●继承中成员变量的关系(掌握)

A:案例演示

a:不同名的变量

b:同名的变量

```
public class Demo1 {
    public static void main(String[] args) {
        Zi z = new Zi();
        z.show();
    }
}
class Fu{
    int a = 10;
    int b = 20;
    int c = 30;
}
class Zi extends Fu{
    int b = 50; //如果子类中的变量和父类中同名,取就近原则
   int d= 40;
    public void show(){
        System.out.println("a"+a);
        System.out.println("b"+b);
        System.out.println("c"+c);
        System.out.println("d"+d);
    }
}
```

●this 和 super 的区别和应用(掌握)

A:this 和 super 都代表什么

this:代表当前对象的引用,谁来调用我,我就代表谁

super:代表当前对象父类的引用

B:this 和 super 的使用区别

a:调用成员变量

this.成员变量 调用本类的成员变量,也可以调用父类的成员变量

super.成员变量 调用父类的成员变量

```java
public class Demo1 {
    public static void main(String[] args) {
        Zi z = new Zi();
        z.show();
    }
}
class Fu{
    int a = 10;
    int b = 20;
    int c = 30;
}
class Zi extends Fu{
    int b = 50; //如果子类中的变量和父类中同名,取就近原则
    int d= 40;
    public void show(){
        System.out.println("a:"+this.a);//其实隐含了一个this,也可以省略
        System.out.println("b:"+this.b);//取的子类中的b 的值
       System.out.println("c:"+c);
        System.out.println("d:"+d);
        System.out.println("父类中 b 的值:"+super.b);//super 用来调用父类中同名的变量值,super 指向是父类的引用,不能省略
    }
}
```

调用父类中同名的成员方法:

```java
public class Demo1 {
    public static void main(String[] args) {
        Zi z = new Zi();
        z.show();
    }
}
class Fu{
    int a = 10;
    int b = 20;
    int c = 30;
    public void show(){
        System.out.println("Fu 类中 show 方法:a:"+a+"-b:"+b+"-:c"+c);
    }
}
class Zi extends Fu{
    int b = 50; //如果子类中的变量和父类中同名,取就近原则
   int d= 40;
    public void show(){
        System.out.println("子类中 show");
        System.out.println("a:"+this.a);
        System.out.println("b:"+this.b);//取的子类中的b 的值
      System.out.println("c:"+c);
        System.out.println("d:"+d);
        System.out.println("父类中 b 的值:"+super.b);//super 用来调用父类中同名的变量值,super 指向是父类的引用
```

```
        super.show();//调用父类中的 show 方法
    }
}
```

b:调用构造方法（待会讲）

this(...) 调用本类的构造方法

super(...) 调用父类的构造方法

●继承中构造方法的关系(掌握)

A:案例演示

子类中所有的构造方法默认都会访问父类中空参数的构造方法

```
16
17 class Father {
18     public Father() {
19         System.out.println("Father 的构造方法");
20     }
21 }
22
23 class Son extends Father {
24     public Son() {
25         System.out.println("Son 的构造方法");
26     }
27 }
```

```
public static void main(String[] args) {
    Son s = new Son();
}
```

B:为什么呢?

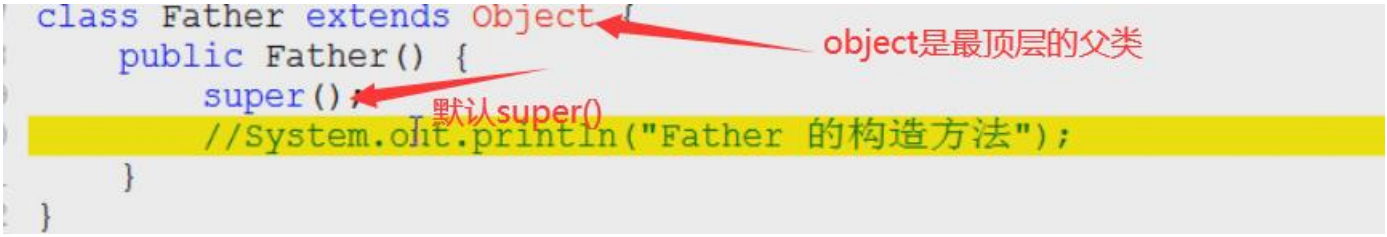
因为子类会继承父类中的数据，可能还会使用父类的数据。
所以，子类初始化之前，一定要先完成父类数据的初始化。

其实：

每一个构造方法的第一条语句默认都是：super() ,Object 类最顶层的父类。相当于默认继承 object 类

```
class Son extends Father {
    public Son() {
        super();                    //这是一条语句,如果不写,系统会默认加上
        System.out.println("Son 的构造方法");
    }
}
```

用来访问父类的空参构造




```java
public class Demo1 {
    public static void main(String[] args) {
        Zi z = new Zi();

    }
}
class Fu extends Object{ //Object 是 java 中所有类的超类(基类,祖宗)
    public Fu(){
        super();//所有的类的无参构造方法都隐含了一个super(),他们直接或者间接的继承了Object 类
      System.out.println("Fu 类的无参构造方法");
    }
    int a = 10;
    int b = 20;
    int c = 30;

}
class Zi extends Fu{
    int b = 50; //如果子类中的变量和父类中同名,取就近原则
   int d= 40;
    public Zi(){
        //子类创建对象会先调用父类的无参构造方法,子类创建对象有可能需要使用父类中的数据
      super();//在子类的构造方法第一行隐含这个方法,如果我们不加,系统会自动默认加上,用来调用父类的无参构造方法
      System.out.println("Zi 类中的无参构造方法");
    }

}
```

●继承中构造方法的注意事项(掌握)

A:案例演示

父类没有无参构造方法,子类怎么办?

super 解决

this 解决

B:注意事项

super(…)或者 this(….)必须出现在构造方法的第一条语句上

```java
class Father {
    private String name;                    //姓名
    private int age;                        //年龄

    public Father() {                       //空参构造
        System.out.println("Father 空参构造");
    }

    public Father(String name,int age) {    //有参构造
        this.name = name;
        this.age = age;
        System.out.println("Father 有参构造");
    }
```

```java
    public void setName(String name) {  //设置姓名
        this.name = name;
    }

    public String getName() {           //获取姓名
        return name;
    }

    public void setAge(int age) {       //设置年龄
        this.age = age;
    }

    public int getAge() {               //获取年龄
        return age;
    }
```

```java
class Son extends Father {
    public Son() {
        System.out.println("Son 空参构造");
    }

    public Son(String name,int age) {
        System.out.println("Son 有参构造");
    }
}
```

```java
public static void main(String[] args) {
    Son s1 = new Son();
    System.out.println("--------------------");
    Son s2 = new Son("张三",23);
}
```

父类没有无参构造方法,子类怎么办?

```
public Son(String name,int age) {    //有参构造
    super(name,age);
    System.out.println("Son 有参构造");
}
```

```
public Son() {                       //空参构造
    super("李四",24);
    System.out.println("Son 空参构造");
}
```

```
Son s1 = new Son();
System.out.println(s1.getName() + "..." + s1.getAge());
System.out.println("----------------------");
Son s2 = new Son("张三",23);
System.out.println(s2.getName() + "..." + s2.getAge());
```

一般用子类有参　以上是一种解决方案

```
public class Demo2 {
    public static void main(String[] args) {
        Student stu  = new Student("小明",19);
        stu.show();
        System.out.println("-----------------");
        Student stu2 = new Student("小红",19,"英语班");
        stu2.show();
        System.out.println("----------------");
        Student stu3 = new Student("张涵",20,"数学班",'男');
        stu3.show();
    }
}
class Person{
    private String name;
    private int age;
    public Person(String name,int age){
        super();//在父类中有参的构造方法第一行隐含了这个super()方法
       this.name = name;
        this.age = age;
    }
    public void show(){
        System.out.println(name+"-"+age);
    }
}
class Student extends Person{
    String classNo;
    public Student(String name,int age){
        super(name,age);//在这里调用父类的有参构造方法
   }
```

```
    public Student(String name,int age,String classNo){
        super(name,age);//调用父类两参数的的构造方法,要放在第一行
      this.classNo = classNo;
    }
    public void show(){
        super.show();
        System.out.println("classNO:"+classNo+"sex:"+sex);
    }
    char sex;
    public Student(String name,int age,String classNo,char sex){
        this(name,age,classNo);//开发中推荐这么写:调用本类中三个参数的构造方法,复合 java 中代码的复用
      this.sex= sex;
    }
}
```

●继承中的面试题(掌握)

A:给出程序结果：案例演示

```
class Fu{
    public int num = 10;
    public Fu(){
        System.out.println("fu");
    }
}
class Zi extends Fu{
    public int num = 20;
    public Zi(){
        System.out.println("zi");
    }
    public void show(){
        int num = 30;
        System.out.println(num);
        System.out.println(this.num);
        System.out.println(super.num);
    }
}
public class Test1_Extends {
    public static void main(String[] args) {
        Zi z = new Zi();
        z.show();
    }
}
```

输出了什么数据:fu zi 30 20 10

2.说出结果：

```
class Fu {
```

```
    static {
        System.out.println("静态代码块 Fu");//无论创建多少个对象只会调用一次
    }

    {
        System.out.println("构造代码块 Fu");
    }

    public Fu() {
        System.out.println("构造方法 Fu");
    }
}

class Zi extends Fu {
    static {
        System.out.println("静态代码块 Zi");
    }

    {
        System.out.println("构造代码块 Zi");
    }

    public Zi() {
        System.out.println("构造方法 Zi");
    }
}

public class Test1_Extends{
   public static void main(String[]args){
      Zi z = new Zi();
   }
}
```

结果:

```
静态代码块 Fu
静态代码块 Zi
构造代码块 Fu
构造方法 Fu
构造代码块 Zi
构造方法 Zi
```

子类在加载之前会先加载父类进内存

●继承中成员方法关系(掌握)

A:案例演示

a:不同名的方法

b:同名的方法

```
public class Demo3 {
    public static void main(String[] args) {
        Zi z = new Zi();
        z.show();
        z.display();
    }
}
class Fu{
    String name = "张三";
    public void show(){
        System.out.println("fu----"+name);
    }
}
class Zi extends Fu{
    String name = "李四";//子类特有的属性
    //要访问子类特有的属性,要使用重写父类的方法来调用
    public void show(){  //子类的方法定义的和父类一样,那么叫做方法的重写
       System.out.println("zi_____"+name);
    }
    //子类特有的方法
    public void display(){
        System.out.println("name:"+name);
    }

}
```

重写要注意的细节:

```
import org.w3c.dom.ls.LSOutput;

public class Demo3 {
    public static void main(String[] args) {
        Zi z = new Zi();
        z.show();
        z.display();
    }
}
class Fu{
    String name = "张三";
    public void show(){
        System.out.println("fu----"+name);
    }
        //私有的方法不能被继承
    private void getMsg(){
```

```
        System.out.println("fu----getMsg()方法");
    }
}
class Zi extends Fu{
    String name = "李四";//子类特有的属性
    //要访问子类特有的属性,要使用重写父类的方法来调用
    public void show(){  //子类的方法定义的和父类一样,那么叫做方法的重写

        System.out.println("zi_____"+name);
        super.show();//是在子类中调用父类的方法
    }
    //子类特有的方法
    public void display(){
        System.out.println("name:"+name);
    }
   //不叫重写,因为父类是私有的getMsg 方法,子类这个叫子类特有的方法与父类没有关系
    public void getMsg(){
        System.out.println("zi_____getMsg");
    }

}
```

●方法重写概述及其应用(掌握)

A:什么是方法重写

重写:子父类出现了一模一样的方法(注意:返回值类型可以是子父类,这个我们学完面向对象讲)

B:方法重写的应用:

当子类需要父类的功能，而功能主体子类有自己特有内容时，可以重写父类中的方法。这样，即沿袭了父类的功能，又定义了子类特有的内容。

C:案例演示

a:定义一个手机类。

```
public class Demo4 {
    public static void main(String[] args) {
        Phone4G p = new Phone4G();
        p.call();
        p.sendMessage();
        p.game3d();
    }
}
class Phone{
    public void call(){
        System.out.println("打电话");
    }

}
class Phone3G extends Phone{
```

```
    public void sendMessage(){
        System.out.println("可以发文字短信");
    }
}
class Phone4G extends Phone3G{
    public void sendMessage(){
        super.sendMessage();
        System.out.println("发语音短信");
    }

    //拥有自己特有的功能
    public void game3d(){
        System.out.println("可以玩大型 3D 游戏");
    }
}
```

●方法重写的注意事项(掌握)

A:方法重写注意事项

a:父类中私有方法不能被重写

因为父类私有方法子类根本就无法继承

b:子类重写父类方法时，访问权限不能更低

最好就一致

c:父类静态方法，子类也必须通过静态方法进行重写

其实这个算不上方法重写，但是现象确实如此，至于为什么算不上方法重写，多态中我会讲解(**静态只能覆盖静态**)

子类重写父类方法的时候，最好声明一模一样。

B:案例演示

方法重写注意事项

```
public class Demo3 {
    public static void main(String[] args) {
        Zi z = new Zi();
        z.show();
        Zi.print();
    }
}
class Fu{
    protected void show(){  //受保护的修饰符protected
        System.out.println("fu----");
    }

    public static void print(){
```

```
        System.out.println("Fu----print");
    }
}
class Zi extends Fu{
    public void show(){  //子类的方法定义的和父类一样,那么叫做方法的重写
       System.out.println("zi_____");
    }
    //静态的方法不叫重写
   public static void print(){
        System.out.println("Zi_______print");
    }
}
//子类重写父类发方法,修饰符只能和父类一样或者着比父类的权限大,不能比父类的权限小
```

推荐在开发中重写方法,写成与父类一模一样

●方法重写的面试题(掌握)

A:方法重写的面试题

Override(重写)和 Overload（重载）的区别?Overload 能改变返回值类型吗?
overload 可以改变返回值类型,只看参数列表

方法重写：前提是有继承关系的子父类,子类中出现了和父类中方法声明一模一样的方法。与返回值类型有关,返回值是一致(或者是子父类)的

方法重载：本类中出现的方法名一样，参数列表不同的方法。与返回值类型无关。

子类对象调用方法的时候：

先找子类本身，再找父类。(就近原则)

●使用继承前的学生和老师案例(掌握)

A:案例演示
使用继承前的学生和老师案例
属性:姓名,年龄
行为:吃饭
老师有特有的方法:讲课
学生有特有的方法:学习

●使用继承后的学生和老师案例(掌握)

A:案例演示
把学生和老师的共有特性抽取为一个新的类 Person
使用继承后的学生和老师案例

●猫狗案例分析,实现及测试(掌握)

A:猫狗案例分析

B:案例演示

猫狗案例继承版

属性:毛的颜色,腿的个数

行为:吃饭

猫特有行为:抓老鼠 catchMouse

狗特有行为:看家 lookHome

```
public class Demo5 {
    public static void main(String[] args) {
        Dog d = new Dog();
        d.color = "小黄";
        d.legs =4;
        d.eat();
        d.lookHome();

        Cat c = new Cat();
        c.color ="小白";
        c.legs = 4;
        c.eat();
        c.catchMouse();
    }
}
/*
猫狗案例继承版
属性:毛的颜色,腿的个数
行为:吃饭
猫特有行为:抓老鼠 catchMouse
狗特有行为:看家 lookHome
 */
class Animal{
    String color;
    int legs;
    public void eat(){
        System.out.println("吃东西,color:"+color+"legs:"+legs);
    }
}
class Dog extends Animal{
    public void lookHome(){
        System.out.println("看家");
    }
}
class Cat extends Animal{
    public void catchMouse(){
        System.out.println("抓老鼠");
    }

}
```

```
public class Demo6 {
    public static void main(String[] args) {
        Zi z= new Zi();
        z.house();
    }
}
class Fu{
    public void house(){
        System.out.println("父母的房子");
    }
}
class Zi extends Fu{
    public void house(){
        System.out.println("自己的房子");
        System.out.println("五一回家看父母");
        super.house();
    }
}
```

●final 关键字修饰类,方法以及变量的特点(掌握)

A:final 概述(最终的)

B:final 修饰特点

修饰类，类不能被继承

修饰变量，变量就变成了常量，只能被赋值一次

修饰方法，方法不能被重写

C:案例演示

final 修饰特点

```
final class ZhaoGao{    //被final 修饰的类为太监类,不能被继承
    public void work(){
        System.out.println("皇上的秘书");
    }
}
//如果继承一个final 修饰的类就会报错,编译通不过
class ZhaoXiaoGao extends ZhaoGao{

}
```