●面向对象(代码块的概述和分类)(了解)(面试的时候会问,开发中用的不多)

A:代码块概述

在 Java 中，使用{ }括起来的代码被称为代码块。

B:代码块分类

根据其位置和声明的不同，可以分为局部代码块，构造代码块，静态代码块，同步代码块(多线程讲解)。

C:常见代码块的应用

a:局部代码块

在方法中出现；限定变量生命周期，及早释放，提高内存利用

```
public class Demo10 {
    public static void main(String[] args) {
        show();
    }

    static int c = 30;
    public static void show(){
        System.out.println("这是 show 方法");
        //在方法中的代码块为局部代码块
       int b = 20;
        {
            int a = 10;  //有效范围在局部代码块之内
          System.out.println(a);
            System.out.println(b);//局部代码块中可以使用方法中的局部变量
          System.out.println(c);//局部代码块中可以使用类中的成员变量
       }
//          System.out.println(a);//局部代码块中的变量出了局部代码块就被释放无效
    }
}
    /*
    局部代码块中的数据在运行之后会立马被从内存释放掉，可以节约内存
     */
```

b:构造代码块 (初始化块)

创建一个学生类，提供 getset 方法 提供有参无参构造 添加构造代码块

```
{                                                    //构造代码块:
    System.out.println("构造代码块");
}
```

Main() 初始化对象

```
//构造代码块:每创建一次对象就会执行一次,优先于构造函数执行
```

学生要学习 创建一个方法

```
{                                                    //
    //System.out.println("构造代码块");
    study
}

public void study() {
    System.out.println("学生学习");
}
```

在开发中用的不多

在类中方法外出现；多个构造方法方法中相同的代码存放到一起，每次调用构造都执行，并且在构造方法前执行

```
public class Demo10 {

    public static void main(String[] args) {
        Doctor d = new Doctor();
    }
}

class Doctor{
    private String name;
    private int no;
    public Doctor(){
        System.out.println("Doctor 的构造方法");
    }
    //类中方法外的代码块称为构造代码块
    {
        System.out.println("我是构造代码块");
    }
    /*
        构造代码块的特点：
       优先于构造方法运行，使用场景：在使用数据库时，用来加载数据库的驱动
     */


}
```

c:静态代码块

在类中方法外出现，并加上 static 修饰；用于给类进行初始化，在加载的时候就执行，并且只执行一次。

开发中一般用于加载驱动

```
public class Demo10 {

    public static void main(String[] args) {
        Doctor d = new Doctor();
    }
}

class Doctor{
    private String name;
    private int no;
    public Doctor(){
        System.out.println("Doctor 的构造方法");
    }
    //类中方法外的代码块称为构造代码块
   {
        System.out.println("我是构造代码块");
    }
    /*
        构造代码块的特点：
       优先于构造方法运行，使用场景：在使用数据库时，用来加载数据库的驱动
     */

    //在构造代码块前用 static 修饰就是静态代码块
   static{
        System.out.println("我是 static 代码块");
    }
    /*
        构造代码块的特点：
       优先于构造方法运行，使用场景：在使用数据库时，也可以用来加载数据库的驱动，或者初始化（提前准备好），程序运行后需要的数据
     */

}
```

运行的先后顺序：

静态代码块> 构造代码块>构造方法

●继承案例演示(掌握)

A:继承(extends)

让类与类之间产生关系,子父类关系

B:继承案例演示：

动物类,猫类,狗类

定义两个属性(颜色,腿的个数)两个功能(吃饭，睡觉)

C:案例演示

使用继承前

```
public class Demo1 {
    public static void main(String[] args) {

    }
}
class Dog{
    String color;//花色
   int legs;//腿的个数
   public void cry(){
        System.out.println("旺旺。。。");
    }
}
class Cat{
    String color;
    int legs;
    public void cry(){
        System.out.println("喵喵");
    }
}
class Bird{
    String color;
    int legs;
    public void cry(){
        System.out.println("咕咕。。。");
    }
}
```

我们发现，如果再去写更多的动物的时候，重复的代码特别多

需求：能不能把重复的代码复用起来？

解决？使用继承

D:案例演示

使用继承后

```
public class Demo1 {
    public static void main(String[] args) {
        Dog d = new Dog();
```

```
        d.legs = 4;
        d.color = "白狗";
        d.cry();
        Cat c = new Cat();
        c.color="黑猫";
        c.legs = 4;
        c.cry();
    }
}
class Animal{    //父类（基类）
   String color;
    int legs;
    public void cry(){
        System.out.println("动物叫唤。。。。"+color+"--"+legs);
    }
}
//下面都是子类
class Dog extends Animal{

}
class Cat extends Animal{

}
class Bird extends Animal{

}
```

继承之后，子类中会拥有一份和父类中一样的数据（属性和方法）

●继承的好处和弊端(掌握)

A:继承的好处

a:提高了代码的复用性

b:提高了代码的维护性

c:让类与类之间产生了关系，是多态的前提

B:继承的弊端

会强加给子类一些子类可能并不需要的功能

```
public class Demo1 {
    public static void main(String[] args) {
        Dog d = new Dog();
        d.legs = 4;
        d.color = "白狗";
```

```
        d.cry();
        d.catMouse();

        Cat c = new Cat();
        c.color="黑猫";
        c.legs = 4;
        c.cry();
        c.catMouse();
    }
}
class Animal{   //父类（基类）
   String color;
    int legs;
    public void cry(){
        System.out.println("动物叫唤。。。。"+color+"--"+legs+"我很可爱");
    }
    public void catMouse(){  //这个方法有些子类不需要，会强加给它
      System.out.println("抓老鼠");
    }
}
//下面都是子类
class Dog extends Animal{

}
class Cat extends Animal{

}
class Bird extends Animal{

}
```

类的耦合性增强了。

开发的原则：高内聚，低耦合。

耦合：类与类的关系

内聚：就是自己完成某件事情的能力

●Java 中类的继承特点(掌握)

A:Java 中类的继承特点

a:Java 只支持单继承，不支持多继承。(一个儿子只能有一个爹)

有些语言是支持多继承，格式：extends 类 1,类 2,...

```
public class A {
    public void show(){
        System.out.println("show");
    }
```

```
}
class B {

}
//不能继承多个爹
class C extends A,B{  //子类

}
```

b:Java 支持多层的继承(继承体系)

```
public class A {   //爷爷
    public void show(){
        System.out.println("show");
    }
}
class B extends A{  //父类

}
class C extends B{  //子类

}
class D extends C{  //孙类

}
```

B:案例演示

Java 中类的继承特点

如果想用这个体系的所有功能用最底层（最后的子类）的类创建对象

如果想看这个体系的共性功能,看最顶层的类

```
public class Demo1 {
    public static void main(String[] args) {
        //最底层的类具备继承体系中所有的功能
       BaoZi b =new BaoZi();
        b.cry();
        b.catMouse();
        b.paiShu();
//最顶层的类具备整个继承体系中公有的功能
       Animal a = new Animal();
        a.cry();//公有的功能
    }
}
class Animal{   //父类（基类）
    String color;
    int legs;
    public void cry(){
        System.out.println("动物叫唤。。。。"+color+"--"+legs+"我很可爱");
```

```
    }

}
class Cat extends Animal{
    public void catMouse(){
        System.out.println("猫抓老鼠");
    }
}
class BaoZi extends Cat{
    public void paiShu(){
        System.out.println("爬树");
    }
}
```

```
public class Demo2 {
    public static void main(String[] args) {
        SiCong s = new SiCong();
        s.wanDa();
        s.zhiHui();
        s.getMoney();
        s.feiYu();

        JianLin j = new JianLin();
        j.zhiHui();
        j.wanDa();
    }
}
class JianLin{
    long money = 5000000000L;
    private String wife = "思聪的亲妈";//私有的属性和方法不能被继承
   public void wanDa(){
        System.out.println("万达集团"+wife);
    }
    public void zhiHui(){
        System.out.println("智慧");
    }
}
class SiCong extends JianLin{
    private String wife="等等等";
    public void getMoney(){
        System.out.println(money);
        System.out.println(wife);
    }
    public void feiYu(){
        System.out.println("飞鱼网咖");
    }
```

```
}
```

●继承的注意事项和什么时候使用继承(掌握)

A:继承的注意事项

a: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)

b:子类不能继承父类的构造方法，但是可以通过 super(马上讲)关键字去访问父类构造方法。

c:不要为了部分功能而去继承

项目经理 姓名 工号 工资 奖金

程序员 姓名 工号 工资

从这个题里面来看，要用项目经理继承程序员，但是不符合生活的常理，在这里抽取一个公共类（员工类）作为他们的共同父类

```
public class Demo3 {
    public static void main(String[] args) {
        Programer p = new Programer();
        p.name="张工";
        p.no = 1001;
        p.salary = 10000;
        p.work();

        System.out.println("-------------");
        Manager m = new Manager();
        m.name ="李工";
        m.no = 1101;
        m.salary = 20000;
        m.allowance = 8000;
        m.work();
        m.managerWork();

    }
}
//员工类
class Employ{
    String name;
    int no;
    int salary;//工资
   public void work(){
        System.out.println(name+"-"+no+"-"+salary+"好好工作");
    }

}
class Programer extends  Employ{
}
class Manager extends Employ{
    int allowance;
```

```
    public void managerWork(){
        System.out.println("员工管理"+allowance);
    }
}
```

●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

2、Student

姓名　年龄　学号　　好好学习　　吃饭

Teacher　姓名　年龄　工号　　好好教书　　吃饭

职工

Perosn　　姓名　年龄　　吃饭

3、香蕉　　苹果　　梨子　　西瓜　　继承　水果类

4、将今天的课堂案例　写一遍