

# P252-263作业+内容梳理

23:41

2021年5月13日

## 本章作业

1. 编写类A01, 定义方法max, 实现求某个double数组的最大值, 并返回 **Homework01.java**
2. 编写类A02, 定义方法find, 实现查找某字符串数组中的元素查找, 并返回索引, 如果找不到, 返回-1. **Homework02.java**
3. 编写类Book, 定义方法updatePrice, 实现更改某本书的价格, 具体: 如果价格 > 150, 则更改为150, 如果价格 > 100, 更改为100, 否则不变 **Homework03.java**

## 本章作业

4. 编写类A03, 实现数组的复制功能copyArr, 输入旧数组, 返回一个新数组, 元素和旧数组一样 **Homework04.java**
5. 定义一个圆类Circle, 定义属性: 半径, 提供显示圆周长功能的方法, 提供显示圆面积的方法 **Homework05.java**
6. 编程创建一个Cale计算类, 在其中定义2个变量表示两个操作数, 定义四个方法实现求和、差、乘、商(要求除数为0的话, 要提示) 并创建两个对象, 分别测试 **Homework06.java**

## 本章作业

7. 设计一个Dog类, 有名字、颜色和年龄属性, 定义输出方法show()显示其信息。并创建对象, 进行测试。【提示 this.属性】 **Homework07.java**

8. 给定一个Java程序的代码如下所示, 则编译运行后, 输出结果是()

```
public class Test { //Homework08.java
    int count = 9;
    public void count1() {
        count=10;
        System.out.println("count1=" + count);
    }
    public void count2() {
        System.out.println("count1=" + count++);
    }
    public static void main(String args[]) {
        new Test().count1();
        Test t1= new Test();
        t1.count2();
        t1.count2();
    }
}
```

9. 定义Music类, 里面有音乐名name、音乐时长times属性, 并有播放play功能和返回本身属性信息的功能方法getInfo. **Homework09.java**

10. 试写出以下代码的运行结果 () // **Homework10.java**

```
class Demo{
    int i=100;
    public void m () {
        int j=i++;
        System.out.println("i="+i);
        System.out.println("j="+j);
    }
}
class Test{
    public static void main(String[] args){
        Demo d1=new Demo();
        Demo d2 = d1;
        d2.m();
        System.out.println(d1.i);
        System.out.println(d2.i);
    }
}
```

11. 在测试方法中, 调用method方法, 代码如下, 编译正确, 试写出method方法的定义形式, 调用语句为: System.out.println(method(method(10.0,20.0),100); Homework11.java

12. 创建一个Employee类, 属性有(名字, 性别, 年龄, 职位, 薪水), 提供3个构造方法, 可以初始化 (1) (名字, 性别, 年龄, 职位, 薪水), (2) (名字, 性别, 年龄) (3) (职位, 薪水), 要求充分复用构造器 Homework12.java

13. 将对象作为参数传递给方法。 Homework13.java

题目要求:

(1) 定义一个Circle类, 包含一个double型的radius属性代表圆的半径, 一findArea()方法返回圆的面积。

(2) 定义一个类PassObject, 在类中定义一个方法printAreas(), 该方法的定义如下: public void printAreas(Circle c, int times) //方法签名

(3) 在printAreas方法中打印输出1到times之间的每个整数半径值, 以及对应的面积。例如, times为5, 则输出半径1, 2, 3, 4, 5, 以及对应的圆面积。

(4) 在main方法中调用printAreas()方法, 调用完毕后输出当前半径值。程序运行结果如图所示

Radius	Area
1.0	3.141592653589793
2.0	12.566370614359172
3.0	28.274333882308138
4.0	50.26548245743669
5.0	78.53981633974483

精英平  
教育 英

## 本章作业

14. 扩展题, 学员自己做。 Homework14.java

有个人 Tom 设计他的成员变量. 成员方法, 可以电脑猜拳.

电脑每次都会随机生成 0, 1, 2

0 表示 石头 1 表示剪刀 2 表示 布

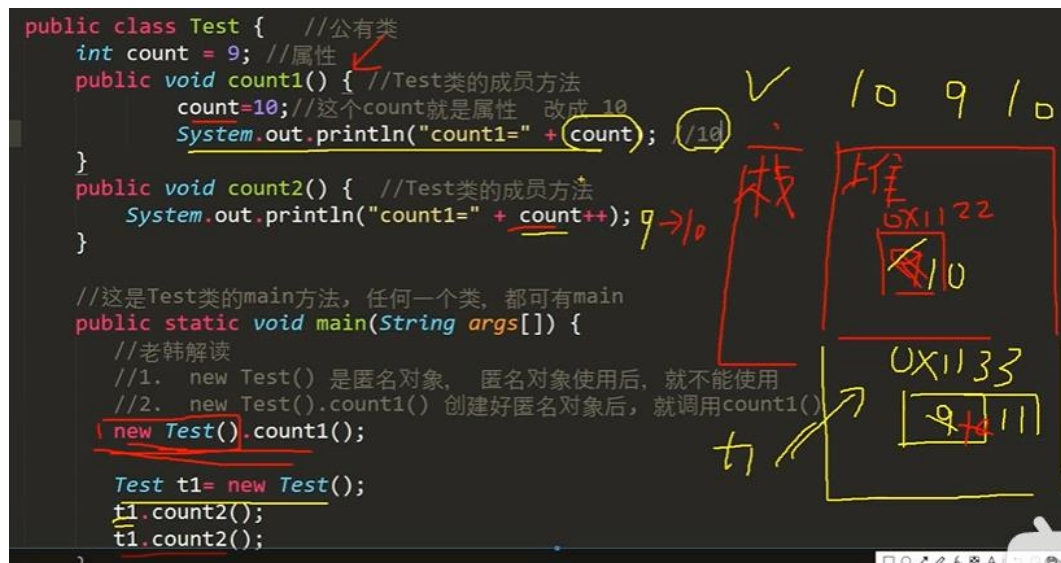
并要可以显示 Tom的输赢次数 (清单)



# P264-272idea使用与快捷键

2021年5月13日

23:41



## 8.3.5 IDEA 常用快捷键

- 1) 删除当前行，默认是 `ctrl + Y` 自己配置 `ctrl + d`
- 2) 复制当前行，自己配置 `ctrl + alt + 向下光标`
- 3) 补全代码 `alt + /`
- 4) 添加注释和取消注释 `ctrl + /` 【第一次是添加注释，第二次是取消注释】
- 5) 导入该行需要的类 先配置 `auto import`，然后使用 `alt + enter` 即可
- 6) 快速格式化代码 `ctrl + alt + L`
- 7) 快速运行程序 自己定义 `alt + R`
- 8) 生成构造器等 `alt + insert` [提高开发效率]
- 9) 查看一个类的层级关系 `ctrl + H` [学习继承后，非常有用]
- 10) 将光标放在一个方法上，输入 `ctrl + B`，可以定位到方法 [学继承后，非常有用]
- 11) 自动的分配变量名，通过 在后面加 `.var` [老师最喜欢的]
- 12) 还有很多其它的快捷键...

## 8.3.6 模板/自定义模板

file -> settings -> editor -> Live templates ->  
查看有哪些模板快捷键/可以自己增加模板

模板可以高效的完成开发，提高速度

```
public class TestTemplate {  
    //main就是一个模板的快捷键。  
    public static void main(String[] args) {  
        //sout模板快捷键  
        System.out.println("hello,world");  
  
        //fori模板快捷键  
  
    }  
}
```



# P273-278包的使用细节

22:49

2021年5月14日

## 包

### 包的三大作用

1. 区分相同名字的类
2. 当类很多时,可以很好的管理类 [看Java API 文档]
3. 控制访问范围

### 包基本语法

`package com.hspedu;`

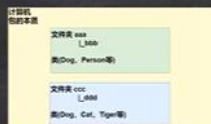
说明:

1. `package` 关键字,表示打包.
2. `com.hspedu`: 表示包名

## 包

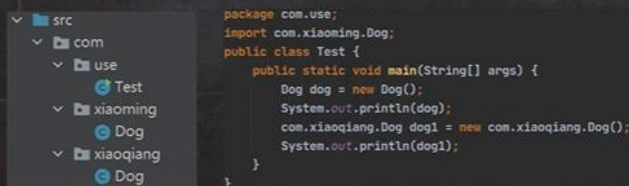
### 包的本质分析(原理)

包的本质 实际上就是创建不同的文件夹/目录来保存类文件, 画出示意图。



### 快速入门

使用打包技术来解决上面的问题, 不同包下Dog类



## 包

### 包的命名

#### ✓ 命名规范:

只能包含数字、字母、下划线、小圆点,但不能用数字开头, 不能是关键字或保留字  
demo.class.exec1  
demo.12a  
demo.ab12.oa

#### ✓ 命名规范

一般是小写字母+小圆点一般是  
com.公司名.项目名.业务模块名  
比如: com.hspedu.oa.model; com.hspedu.oa.controller;  
举例:  
com.sina.crm.user //用户模块  
com.sina.crm.order // 订单模块  
com.sina.crm.utils //工具类

## 包

### 常用的包

一个包下,包含很多的类,java中常用的包有:

- `java.lang.*` //lang包是基本包, 默认引入, 不需要再引入.
- `java.util.*` //util 包, 系统提供的工具包, 工具类, 使用 Scanner
- `java.net.*` //网络包, 网络开发
- `java.awt.*` //是做java的界面开发, GUI

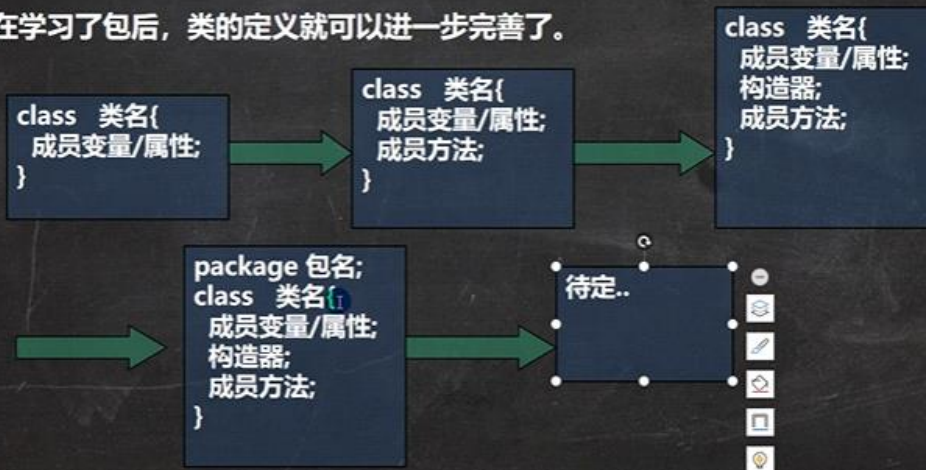
## ● 注意事项和使用细节

### PkgDetail.java

1. package 的作用是声明当前类所在的包，需要放在类的最上面，一个类中最多只有一句package
2. import指令 位置放在package的下面，在类定义前面,可以有多句且没有顺序要求。

## 类定义的进一步完善

在学习了包后，类的定义就可以进一步完善了。



# P279-280访问修饰符

2021年5月15日

18:29

## 访问修饰符

### ● 基本介绍

java提供四种访问控制修饰符号，用于控制方法和属性(成员变量)的访问权限（范围）：

1. 公开级别:用**public** 修饰,对外公开
2. 受保护级别:用**protected**修饰,对子类和同一个包中的类公开
3. 默认级别:没有修饰符号,向同一个包的类公开.
4. 私有级别:用**private**修饰,只有类本身可以访问,不对外公开.

## 访问修饰符

### ● 4种访问修饰符的访问范围

	访问级别	访问控制修饰符	同类	同包	子类	不同包
1	公开	<b>public</b>	✓	✓	✓	✓
2	受保护	<b>protected</b>	✓	✓	✓	X
3	默认	没有修饰符	✓	✓	X	X
4	私有	<b>private</b>	✓	X	X	X

可下载!!!

### ● 使用的注意事项

- 1) 修饰符可以用来修饰类中的属性，成员方法以及类
- 2) 只有默认的和public才能修饰类！，并且遵循上述访问权限的特点。
- 3) 因为没有学习继承，因此关于在子类中的访问权限，我们讲完子类后，在回头讲解
- 4) 成员方法的访问规则和属性完全一样。

//com.hspedu.modifier :



# P281-285封装的使用

2021年5月15日 18:58

## 面向对象编程-封装

### ● 封装介绍

封装(encapsulation)就是把抽象出的数据[属性]和对数据的操作[方法]封装在一起,数据被保护在内部,程序的其它部分只有通过被授权的操作[方法],才能对数据进行操作。

## 面向对象编程-封装

### ● 封装的理解和好处

- 1) 隐藏实现细节: 方法(连接数据库) <-- 调用(传入参数..)
- 2) 可以对数据进行验证, 保证安全合理  
Person {name, age}

### ● 封装的实现步骤 (三步)

- 1) 将属性进行私有化private 【不能直接修改属性】
- 2) 提供一个公共的(public)set方法, 用于对属性判断并赋值  
public void setXxx(类型 参数名) { //Xxx 表示某个属性  
    //加入数据验证的业务逻辑  
    属性 = 参数名;  
}
- 3) 提供一个公共的(public)get方法, 用于获取属性的值  
public 数据类型 getXxx() { //权限判断, Xxx 某个属性  
    return xx;  
}

### ● 快速入门案例

✓ 看一个案例

那么在java中如何实现这种类型呢? 请大家看一个小程序(Encap01.java), 不能随便查看人的年龄, 工资等隐私, 并对设置的年龄进行合理的验证。年龄合理就设置, 否则给默认  
\* 年龄必须在 1-120, 年龄, 工资不能直接查看, name的长度在 2-6 之间

```
3 public class Person {
4
5     public String name;
6     private int age;
7     private double salary;
8     private String job;
9 }
```

### ● 将构造器和setXxx结合

✓ 看一个案例

```
// 还有一种开发方式, 就是将构造方法和setXxx结合
public Person(String name, int age, double salary, String job) {
    this.setName(name);
    this.setAge(age);
    this.setJob(job);
    this.setSalary(salary);
}
```

### ● 课堂练习

com.hspedu.encap: AccountTest.java 和 Account.java

创建程序, 在其中定义两个类: Account和AccountTest类体会Java的封装性。

1. Account类要求具有属性: 姓名 (长度为2位3位或4位)、余额(必须>20)、密码 (必须是六位), 如果不满足, 则给出提示信息, 并给默认值
2. 通过setXxx的方法给Account 的属性赋值。
3. 在AccountTest中测试

提示知识点:

```
String name = "";
int len = name.length();
```



# P286-288继承入门

2021年5月15日 22:48

## 面向对象编程-继承

### ● 为什么需要继承

一个小问题,还是看个程序[com.hspedu.extend\_包: Extends01.java], 提出代码复用的问题。

我们编写了两个类,一个是Pupil类(小学生),一个是Graduate(大学毕业生).  
问题: 两个类的属性和方法有很多是相同的,怎么办?

=> 继承(代码复用性~)

## 面向对象编程-继承

### ● 继承基本介绍和示意图

继承可以解决代码复用,让我们的编程更加靠近人类思维.当多个类存在相同的属性(变量)和方法时,可以从这些类中抽象出父类,在父类中定义这些相同的属性和方法,所有的子类不需要重新定义这些属性和方法,只需要通过**extends**来声明继承父类即可。  
画出继承的示意图

### ● 继承的基本语法

```
class 子类 extends 父类{  
}
```

- 1) 子类就会自动拥有父类定义的属性和方法
- 2) 父类又叫 超类, 基类。
- 3) 子类又叫派生类。

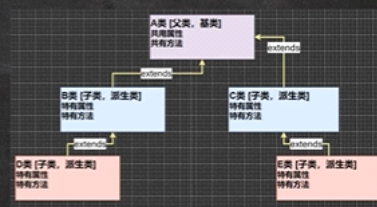
### ● 继承基本介绍和示意图

继承可以解决代码复用,让我们的编程更加靠近人类思维.当多个类存在相同的属性(变量)和方法时,可以从这些类中抽象出父类,在父类中定义这些相同的属性和方法,所有的子类不需要重新定义这些属性和方法,只需要通过**extends**来声明继承父类即可。  
画出继承的示意图

### ● 继承的基本语法

```
class 子类 extends 父类{  
}
```

- 1) 子类就会自动拥有父类定义的属性和方法
- 2) 父类又叫 超类, 基类。
- 3) 子类又叫派生类。



# P289-293继承使用细节

2021年5月16日 14:54

## 面向对象编程-继承

### ● 继承给编程带来的便利

- 1) 代码的复用性提高了
- 2) 代码的扩展性和维护性提高了

### ● 继承的深入讨论/细节问题

#### ● //[com.hspedu.extend\_包:ExtendsDetail.java]

1. 子类继承了所有的属性和方法, **非私有的属性和方法可以在子类直接访问**, 但是私有属性和方法不能在子类直接访问, 要通过父类提供公共的方法去访问

```
public class Base {
    public int n1 = 100;
    protected int n2 = 200;
    int n3 = 300;
    private int n4 = 400;
    public Base() {
        System.out.println("base()...");
    }
    public void test100() {
        System.out.println("test100");
    }
    protected void test200() {
        System.out.println("test200");
    }
    void test300() {
        System.out.println("test300");
    }
    private void test400() {
        System.out.println("test400");
    }
}
```

```
public class Sub extends Base {
    public Sub() {
        System.out.println("sub()...");
    }
    public void sayOk() {
        //我们发现 父类的非private属性和方法
        //都可以访问
    }
}
```

```
Sub sub = new Sub();
System.out.println(sub.n1);
System.out.println(sub.n2);
System.out.println(sub.n3);
System.out.println(sub.n4);
```

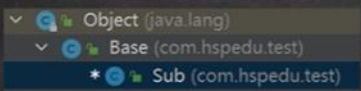
### ● 继承的深入讨论/细节问题

2. 子类必须调用父类的构造器, 完成父类的初始化
3. 当创建子类对象时, 不管使用子类的哪个构造器, 默认情况下总会去调用父类的无参构造器, 如果父类没有提供无参构造器, 则必须在子类的构造器中用 **super** 去指定使用父类的哪个构造器完成对父类的初始化工作, 否则, 编译不会通过 [举例说明]

4. 如果希望指定去调用父类的某个构造器, 则显式的调用一下: **super(参数列表)**

5. **super**在使用时, 必须放在构造器第一行(**super只能在构造器中使用**)

6. **super()** 和 **this()** 都只能放在构造器第一行, 因此这两个方法不能共存在一个构造器

7. java所有类都是Object类的子类, 

8. 父类构造器的调用不限于直接父类! 将一直往上追溯直到Object类(顶级父类)





# P294继承本质详讲

2021年5月16日 16:11

9. 子类最多只能继承一个父类(指直接继承), 即java中是单继承机制。  
思考: 如何让A类继承B类和C类? 【 】

10. 不能滥用继承, 子类和父类之间必须满足 is-a 的逻辑关系

• Person is a Music?  
Person Music  
Music extends Person

Animal  
Cat extends Animal

## 面向对象编程-继承

### • 继承的本质分析(重点)

#### ✓ 案例

我们看一个案例来分析当子类继承父类, 创建子类对象时, 内存中到底发生了什么? 老韩提示: 当子类对象创建好后, 建立查找的关系

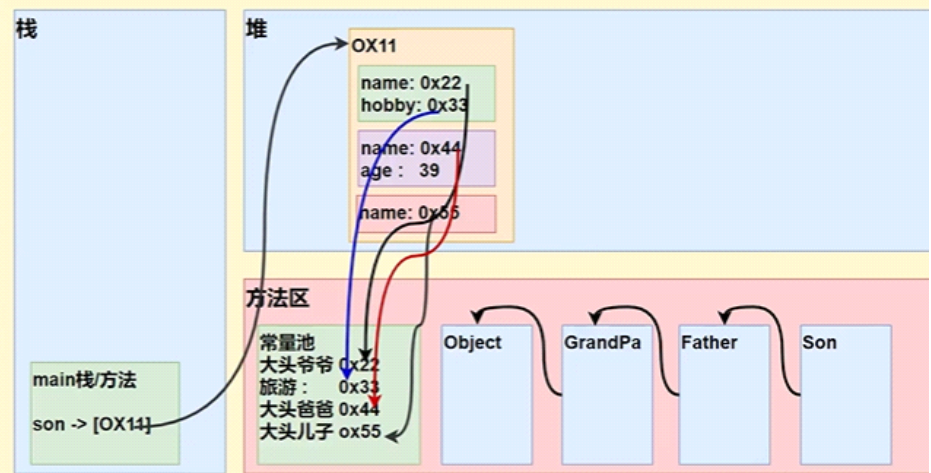
• //com.hspedu.extend\_包:ExtendsTheory.java

#### ✓ 子类创建的内存布局

```
class GrandPa {  
    String name = "大头爷爷";  
    String hobby = "旅游";  
}  
class Father extends GrandPa {  
    String name = "大头爸爸";  
    int age = 39;  
}  
class Son extends Father {  
    String name = "大头儿子";  
}  
Son son = new Son();  
son.name = ?  
son.age = ?  
son.hobby = ?
```

```
public class ExtendsTheory {  
    public static void main(String[] args) {  
        Son son = new Son(); //内存的布局  
    }  
}  
class GrandPa { //爷类  
    String name = "大头爷爷";  
    String hobby = "旅游";  
}  
class Father extends GrandPa { //父类  
    String name = "大头爸爸";  
    int age = 39;  
}  
class Son extends Father { //子类  
    String name = "大头儿子";  
}
```

jvm的内存: 继承的内存布局





# P295-297继承课堂练习

2021年5月16日 19:28

## ● 课堂练习

● //放在 **com.hspedu.extend .exercise** 包下

### 1. 案例1 **ExtendsExercise01.java**

```
class A{
    A(){ System.out.println("a"); }
    A(String name) {System.out.println("a name");}
}
class B extends A{
    B(){ this("abc"); System.out.println("b");}
    B(String name){ super(); System.out.println("b name");}
}
main中: B b=new B(); 会输出什么?
a , b name, b
2min
```

### 2. 案例2 **ExtendsExercise02.java**

```
class A{//A类
    public A(){
        System.out.println("我是A类");
    }
}

class B extends A{ //B类,继承A
    //main方法中: C c =new C(); 输出什
    么内容
    public B(){
        System.out.println("我是B类的无参构造");
    }
    public B(String name){
        System.out.println(name+ "我是B类的有参构造");
    }
}

class C extends B{
    public C(){
        this("hello");
        System.out.println("我是c类的无参构造");
    }
    public C(String name){
        super("hahah");
        System.out.println("我是c类的有参构造");
    }
}
```

### 3. 案例3 **ExtendsExercise03.java**

编写Computer类, 包含CPU、内存、硬盘等属性, getDetails方法用于返回Computer的详细信息

编写PC子类, 继承Computer类, 添加特有属性【品牌brand】

编写NotePad子类, 继承Computer类, 添加特有属性【演示color】

编写Test类, 在main方法中创建PC和NotePad对象, 分别给对象中特有的属性赋值, 以及从Computer类继承的属性赋值, 并使用方法并打印输出信息。

# P298-301super使用细节

2021年5月16日 21:16

## super关键字

- 基本介绍  
super代表父类的引用，用于访问父类的属性、方法、构造器

- 基本语法

//com.hspedu.super\_包下 Super01.java

1. 访问父类的属性，但不能访问父类的private属性 [案例]  
super.属性名;
2. 访问父类的方法，不能访问父类的private方法  
super.方法名(参数列表);
3. 访问父类的构造器(这点前面用过):  
super(参数列表);只能放在构造器的第一句，只能出现一句!

```
public class A {
    public int n1 = 100;
    protected int n2 = 200;
    int n3 = 300;
    private int n4 = 400;
    public void test100() {}
    protected void test200() {}
    void test300() {}
    private void test400() {}
}
```

```
class B extends A{
    public void say(){
        //通过super访问父类的属性 super.属性名，但是不能访问private属性
        System.out.println(super.n1);
        System.out.println(super.n2);
        System.out.println(super.n3);
        //通过super访问父类的方法 super.方法名(参数列表)，
        //但是不能访问父类的private方法
        super.test100();
        super.test200();
        super.test300();
    }
}
```

## super关键字

- super给编程带来的便利/细节

//SuperDetail.java

1. 调用父类的构造器的好处 (分工明确，父类属性由父类初始化，子类的属性由子类初始化)
2. 当子类中有和父类中的成员 (属性和方法) 重名时，为了访问父类的成员，必须通过super。如果没有重名，使用super、this、直接访问是一样的效果! [举例]

```
class A {
    public void say() {
        System.out.println("A say~~");
    }
}
class B extends A {
    public void test(){
        //say();
        //this.say();
        super.say();
    }
}
```

```
System.out.println("B类的sum()");
//希望调用父类-A 的cal方法
//这时，因为子类B没有cal方法，因此我可以使用下面三种方式
```

```
//找cal方法时，顺序是：
// (1) 先找本类，如果有，则调用
// (2) 如果没有，则找父类(如果有，并可以调用，则调用)
// (3) 如果父类没有，则继续找父类的父类，整个规则，就是一样，直到 Object类
// 提示：如果查找方法的过程中，找到了，但是不能访问，则报错，cannot access
// 如果查找方法的过程中，没有找到，则提示方法不存在
//cal();
//this.cal(); //等价 cal
super.cal(); //找cal方法的顺序是直接查找父类，其他的规则一样
```

//演示访问属性的规则

//n1 和 this.n1 查找的规则是

//(1) 先找本类，如果有，则调用

//(2) 如果没有，则找父类(如果有，并可以调用，则调用)

//(3) 如果父类没有，则继续找父类的父类，整个规则，就是一样，直到 Object类

// 提示：如果查找属性的过程中，找到了，但是不能访问，则报错，cannot access

// 如果查找方法的过程中，没有找到，则提示方法不存在

System.out.println(n1);

System.out.println(this.n1);

## super关键字

### ● super和this的比较

No.	区别点	this	super
1	访问属性	访问本类中的属性, 如果本类没有此属性则从父类中继续查找	访问父类中的属性
2	调用方法	访问本类中的方法, 如果本类没有此方法则从父类继续查找.	直接访问父类中的方法
3	调用构造器	调用本类构造器, 必须放在构造器的首行	调用父类构造器, 必须放在子类构造器的首行
4	特殊	表示当前对象	子类中访问父类对象

## super关键字

3. super的访问不限于直接父类, 如果爷爷类和本类中有同名的成员, 也可以使用super去访问爷爷类的成员; 如果多个基类(上级类)中都有同名的成员, 使用super访问遵循就近原则。A->B->C



# P302-305方法重写override

2021年5月16日 22:02

## 方法重写/覆盖(override)

### • 基本介绍

简单的说:方法覆盖(重写)就是子类有一个方法,和父类的某个方法的名称、返回类型、参数一样,那么我们就说子类的这个方法覆盖了父类的方法

### • 快速入门

//com.hspedu.override\_包下 Override01.java

```
class Animal{
    public void cry(){
        System.out.println("动物叫唤。");
    }
}

class Dog extends Animal{
    public void cry(){
        System.out.println("小狗汪汪叫 ...");
    }
}
```

### • 注意事项和使用细节

方法重写也叫方法覆盖, 需要满足下面的条件 //看Animal和Dog]

1. 子类的方法的形参列表,方法名称,要和父类方法的形参列表,方法名称完全一样。【演示】
2. 子类方法的返回类型和父类方法返回类型一样, 或者是父类返回类型的子类  
比如 父类 返回类型是 Object ,子类方法返回类型是String 【演示】

```
public Object getInfo(){    public String getInfo(){
```

3. 子类方法不能缩小父类方法的访问权限 【演示】 public > protected > 默认>private

```
void sayOk(){    public void sayOk(){
```

### • 课堂练习

#### ✓ 题1

请对方法的重写和重载做一个比较: 2min

名称	发生范围	方法名	形参列表	返回类型	修饰符
重载(overload)	本类	必须一样	类型, 个数或者顺序至少有一个不同	无要求	无要求
重写(override)	父子类	必须一样	相同	子类重写的方法, 返回的类型和父类返回的类型一致, 或者是其子类	子类方法不能缩小父类方法的访问范围

### • 课堂练习[学员先做]

#### OverrideExercise.java

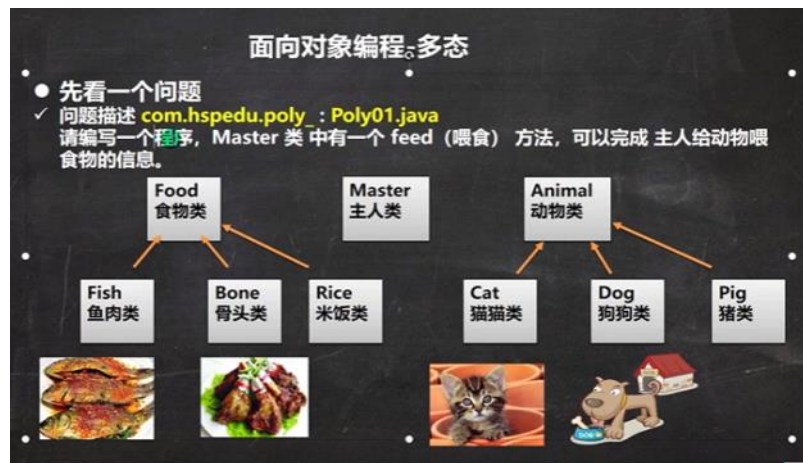
#### ✓ 题2

1. 编写一个Person类, 包括属性/private (name、age) , 构造器、方法say(返回自我介绍的字符串)。
2. 编写一个Student类, 继承Person类, 增加id、score属性/private, 以及构造器, 定义say方法(返回自我介绍的信息)。
3. 在main中, 分别创建Person和Student对象, 调用say方法输出自我介绍。

# P306-309多态入门

2021年5月17日

22:37



- 先看一个问题
- 使用传统的方法来解决 (private属性)

```
class Master {
    private String name;
    public Master(String name) {
        super();
        this.name = name;
    }
    //喂食
    public void feed(Dog dog, Bone bone) {
        System.out.println(this.name + "给" + dog.getName() + "喂" + bone.getName());
    }
    public void feed(Cat cat, Fish fish) {
        System.out.println(this.name + "给" + cat.getName() + "喂" + fish.getName());
    }
    //喂食猪
    //.....
    //.....
}
```

- 传统的方法带来的问题是什么? 如何解决?

引出我们要讲解的多态-> 代码的复用性不高, 而且不利于代码维护

面向对象编程-多态

- 多态基本介绍  
方法或对象具有多种形态。是面向对象的第三大特征，多态是建立在封装和继承基础之上的。
- 多态的具体体现

1. 方法的多态 `PloyMethod.java`  
重写和重载就体现多态 [案例说明:]

```
A a = new A();
//这里通过不同的多态对象调用sum方法，展示去调用不同的方法。
//因此sum方法重载，也是多态的体现。
System.out.println(a.sum(10, 50));
System.out.println(a.sum(10, 50));

B b = new B();
System.out.println(b.say("Hello"));
B obj = new B();
System.out.println(obj.say("ok"));
```

A extends B say sum

- 多态的具体体现

2. 对象的多态 (核心, 困难, 重点)  
老韩重要的几句话(记住):  
(1) 一个对象的编译类型和运行类型可以不一致  
(2) 编译类型在定义对象时, 就确定了, 不能改变  
(3) 运行类型是可以变化的。  
(4) 编译类型看定义时 = 号的 左边, 运行类型看 = 号的 右边

案例: `com.hspedu.poly_objpoly : PolyObject.java`  
`Animal animal = new Dog();` [animal 编译类型是Animal, 运行类型Dog]  
`animal = new Cat();` [animal 的运行类型变成了 Cat, 编译类型仍然是 Animal]

```
Animal animal = new Dog();
//编译时animal是Animal, 运行是animal是Dog
animal.cry();
//animal变成了cat
animal = new Cat();
animal.cry();
```



# P310-312多态向上向下转型和属性重写

2021年5月18日 9:52

## 面向对象编程-多态

- 多态注意事项和细节讨论

**com.hspedu.poly\_detail 包: PolyDetail.java**

- ✓ 多态的前提是: 两个对象(类)存在继承关系
- ✓ 多态的向上转型

- 1) 本质: 父类的引用指向了子类的对象
- 2) 语法: 父类类型 引用名 = new 子类类型();
- 3) 特点: 编译类型看左边, 运行类型看右边。  
可以调用父类中的所有成员(需遵守访问权限),  
不能调用子类中特有成员;  
最终运行效果看子类的具体实现!

```
class Animal{
    String name = "动物";
    int age = 10;
    public void sleep(){
        System.out.println("睡");
    }
    public void run(){
        System.out.println("跑");
    }
    public void eat(){
        System.out.println("吃");
    }
    public void show(){
        System.out.println("hello,你好");
    }
}
class Cat extends Animal{
    public void eat(){
        System.out.println("猫吃鱼");
    }
    public void catchMouse(){
        System.out.println("猫抓老鼠");
    }
}
```

- 多态注意事项和细节讨论
- ✓ 多态的向下转型

- 1) 语法: 子类类型 引用名 = (子类类型) 父类引用;
- 2) 只能强转父类的引用, 不能强转父类的对象
- 3) 要求父类的引用必须指向的是当前目标类型的对象
- 4) 当向下转型后, 可以调用子类类型中所有的成员

## 面向对象编程-多态

- 多态注意事项和细节讨论
- ✓ 属性没有重写之说! 属性的值看编译类型 **PolyDetail02.java**
- ✓ instanceof 比较操作符, 用于判断对象的运行类型是否为XX类型或XX类型的子类型  
【举例说明】 **PolyDetail03.java**

```
//aa 编译类型 AA, 运行类型是BB
AA aa = new BB();
System.out.println(aa instanceof AA);
System.out.println(aa instanceof BB);

Object obj = new Object();
System.out.println(obj instanceof AA); //false
String str = "hello";
//System.out.println(str instanceof AA);
System.out.println(str instanceof Object); //true
```

```
class Base {
    int count = 10;
}
class Sub extends Base{
    int count = 20;
}
```

```
//aa 编译类型 AA, 运行类型是BB
AA aa = new BB();
System.out.println(aa instanceof AA);
System.out.println(aa instanceof BB);

Object obj = new Object();
System.out.println(obj instanceof AA); //false
String str = "hello";
//System.out.println(str instanceof AA);
System.out.println(str instanceof Object); //true
```



# P313-315多态课堂练习+动态绑定机制

2021年5月18日 10:56

面向对象编程-多态

- 课堂练习 `com.hspedu.poly_exercise_包 PolyExercise01.java`

请说出下面的每条语言, 哪些是正确的, 哪些是错误的, 为什么? 2min后老师评讲

```
public class PolyExercise01 {
    public static void main(String[] args) {
        double d = 13.4; //ok
        long l = (long)d; //ok
        System.out.println(l); //13
        int in = 5; //ok
        boolean b = (boolean)in; //不对, boolean -> int
        Object obj = "Hello"; //可以, 向上转型
        String objStr = (String)obj; //可以, 向下转型
        System.out.println(objStr); // hello

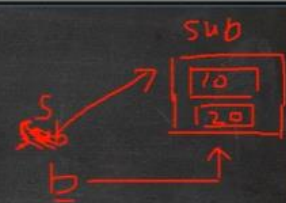
        Object objPri = new Integer(5); //可以, 向上转型
        String str = (String)objPri; //错误, 指向Integer的父类引用, 转成String
        Integer str1 = (Integer)objPri; //可以, 向下转型
    }
}
```

面向对象编程-多态

- 课堂练习 `PolyExercise02.java` 3min

```
class Base{//父类
    int count = 10;
    public void display(){
        System.out.println(this.count);
    }
}
class Sub extends Base{//子类
    int count = 20;
    public void display(){
        System.out.println(this.count);
    }
}
```

```
public class PolyExercise02{//主类
    public static void main(String[] args) {
        Sub s = new Sub(); //ok
        System.out.println(s.count); //10
        s.display(); //20
        Base b = s;
        System.out.println(b == s); //T
        System.out.println(b.count); //10
        b.display(); //20
    }
}
```



面向对象编程-多态

- java的动态绑定机制(非常非常重要.)

Java重要特性: 动态绑定机制 `DynamicBinding.java` `com.hspedu.poly_dynamic_`

```
class A{//父类
    public int i = 10;
    public int sum() {
        return getI() + 10;
    }
    public int sum1() {
        return i + 10;
    }
    public int getI() {
        return i;
    }
}
```

```
class B extends A{//子类
    public int i = 20;
    public int sum() {
        return i + 20;
    }
    public int getI() {
        return i;
    }
    public int sum1() {
        return i + 10;
    }
}
```

```
//main方法中
A a = new B();//向上转型
System.out.println(a.sum()); //?40
System.out.println(a.sum1()); //30
2min
```

- java的动态绑定机制
- 1. 当调用对象方法的时候, 该方法会和该对象的内存地址/运行类型绑定
- 2. 当调用对象属性时, 没有动态绑定机制, 哪里声明, 那里使用

# P316-318多态数组、多态参数

2021年5月18日 14:30

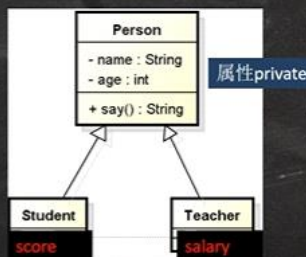
## 面向对象编程-多态

### ● 多态的应用

- 1) 多态数组 `com.hspedu.poly.polyarr` 包 `PloyArray.java`  
数组的定义类型为父类类型, 里面保存的实际元素类型为子类类型

应用实例: 现有一个继承结构如下: 要求创建1个Person对象、2个Student对象和2个Teacher对象, 统一放在数组中, 并调用say方法。

应用实例升级: 如何调用子类特有的方法, 比如Teacher有一个teach, Student有一个study怎么调用?



```
Person[] persons = new Person[5];
persons[0] = new Person("jack", 20);
persons[1] = new Student("jack", 18, 100);
persons[2] = new Student("smith", 19, 30.1);
persons[3] = new Teacher("scott", 30, 20000);
persons[4] = new Teacher("king", 50, 25000);

//循环遍历多态数组, 调用say
for (int i = 0; i < persons.length; i++) {
    //老师提示: person[i] 编译类型是 Person ,运行类型是根据实际情况有JVM来判断
    persons[i].say(); //动态绑定机制
}
```

```
//循环遍历多态数组, 调用say
for (int i = 0; i < persons.length; i++) {
    //老师提示: person[i] 编译类型是 Person ,运行类型是根据实际情况有JVM来判断
    System.out.println(persons[i].say()); //动态绑定机制
    //这里大家聪明。
    if (persons[i] instanceof Student) { //判断person[i] 的运行类型是不是Student
        Student student = (Student) persons[i]; //向下转型
        student.study();
        //小伙伴也可以使用一条语句 ((Student) persons[i]).study();
    } else if (persons[i] instanceof Teacher) {
        Teacher teacher = (Teacher) persons[i];
        teacher.teach();
    } else {
        System.out.println("你的类型有误, 请自己检查...");
    }
}
```

## 面向对象编程-多态

### ● 多态的应用

#### 2) 多态参数

方法定义的形参类型为父类类型，实参类型允许为子类类型

应用实例1：前面的主人喂动物

应用实例2: `com.hspedu.poly.polyparameter` 包 `PloyParameter.java`

定义员工类Employee，包含姓名和月工资[private]，以及计算年工资getAnnual的方法。普通员工和经理继承了员工，经理类多了奖金bonus属性和管理manage方法，普通员工类多了work方法，普通员工和经理类要求分别重写getAnnual方法

测试类中添加一个方法showEmpAnnal(Employee e)，实现获取任何员工对象的年工资,并在main方法中调用该方法 [e.getAnnual()]

测试类中添加一个方法，testWork,如果是普通员工，则调用work方法，如果是经理，则调用manage方法



# P319-321Object详解 (==、equals)

2021年5月18日 18:53

方法摘要	
<code>protected Object clone()</code>	创建并返回此对象的一个副本。
<code>boolean equals(Object obj)</code>	指示其他某个对象是否与此对象“相等”。
<code>protected void finalize()</code>	当垃圾回收器确定不存在对该对象的更多引用时，由对象的垃圾回收器调用此方法。
<code>Class&lt;T&gt; getClass()</code>	返回此 <code>Object</code> 的运行时常类。
<code>int hashCode()</code>	返回该对象的哈希码值。
<code>void notify()</code>	唤醒在此对象监视器上等待的单个线程。
<code>void notifyAll()</code>	唤醒在此对象监视器上等待的所有线程。
<code>String toString()</code>	返回该对象的字符串表示。
<code>void wait()</code>	在其他线程调用此对象的 <code>notify()</code> 方法或 <code>notifyAll()</code> 方法前，导致当前线程等待。
<code>void wait(long timeout)</code>	在其他线程调用此对象的 <code>notify()</code> 方法或 <code>notifyAll()</code> 方法，或者超过指定的时间量
<code>void wait(long timeout, int nanos)</code>	在其他线程调用此对象的 <code>notify()</code> 方法或 <code>notifyAll()</code> 方法，或者其他某个线程中断

## • equals方法

> ==和equals的对比 [面试题]

`com.hspedu.Object_ : Equals01.java`

1. == : 既可以判断基本类型，又可以判断引用类型
2. == : 如果判断基本类型，判断的是值是否相等。示例: `int i=10; double d=10.0;`
3. == : 如果判断引用类型，判断的是地址是否相等，即判定是不是同一个对象【案例说明】

```
// == 比较引用类型，比较的是地址
A obj1 = new A();
A obj2 = new A();
A obj3 = obj1; // 引用赋值，其实际的是地址
```

## Object类详解

### • equals方法

4. equals: 是Object类中的方法，只能判断引用类型，如何看Jdk源码，看老师演示:
5. 默认判断的是地址是否相等，子类中往往重写该方法，用于判断内容是否相等。比如Integer,String【看看String和Integer的equals源代码】

```
public boolean equals(Object obj) {
    return (this == obj);
}
```

Object的equals

```
public boolean equals(Object obj) {
    if (obj instanceof Integer) {
        return value == ((Integer)obj).intValue();
    }
    return false;
}
```

Integer的equals

```
// 这是一些类已经重写了equals方法，变成比较内容了String,Integer
Integer num1 = new Integer(1);
Integer num2 = new Integer(1);
System.out.println("num1.equals(num2)="+ num1.equals(num2));
String str1 = new String("hello");
String str2 = new String("hello");
System.out.println("str1.equals(str2)="+ str1.equals(str2));
```

## Object类详解

### • 如何重写equals方法

应用实例: 判断两个Person对象的内容是否相等，如果两个Person对象的各个属性值都一样，则返回true，反之false。 `EqualsExercise01.java`

```
class Person{
    private String name;
    private int age;
    private char gender;
}
```

# P322-324equals课堂练习

2021年5月19日 10:58

## Object类详解

### ● 如何重写equals方法

应用实例: 判断两个Person对象的内容是否相等, 如果两个Person对象的各个属性值都一样, 则返回true, 反之false。 **EqualsExercise01.java**

```
class Person{
    private String name;
    private int age;
    private char gender;
}
```

### ● 课堂练习题

#### **EqualsExercise02.java**

```
Person p1 = new Person();
p1.name = "hspedu";

Person p2 = new Person();
p2.name = "hspedu";

System.out.println(p1==p2); //False
System.out.println(p1.name.equals(p2.name)); //T
System.out.println(p1.equals(p2)); //False

String s1 = new String("asdf");

String s2 = new String("asdf");
System.out.println(s1.equals(s2)); //T
System.out.println(s1==s2); //F
3min
```

```
class Person{//类
    public String name;
}
```

### ● 课堂练习题

//代码如下 **EqualsExercise03.java 2min**

```
int it = 65;
float fl = 65.0f;
System.out.println("65和65.0f是否相等? " + (it == fl)); //T
char ch1 = 'A'; char ch2 = 12;
System.out.println("65和 'A' 是否相等? " + (it == ch1)); //T
System.out.println("12和ch2是否相等? " + (12 == ch2)); //T

String str1 = new String("hello");
String str2 = new String("hello");
System.out.println("str1和str2是否相等? " + (str1 == str2)); //F

System.out.println("str1是否equals str2? " + (str1.equals(str2))); //T
System.out.println("hello" == new java.sql.Date()); //编译错误
```

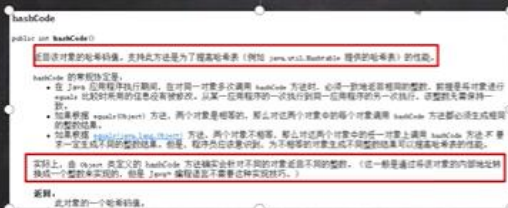


# P325-327 hashCode、toString、finalize

2021年5月19日 11:12

## Object类详解

### ● hashCode方法



### //老韩的6个小结:

- 1) 提高具有哈希结构的容器的效率!
- 2) 两个引用，如果指向的是同一个对象，则哈希值肯定是一样的!
- 3) 两个引用，如果指向的是不同对象，则哈希值是不一样的!
- 4) 哈希值主要根据地址号来的!，不能完全将哈希值等价于地址。
- 5) 案例演示[HashCode.java]: obj.hashCode() [测试: A obj1 = new A(); A obj2 = new A(); A obj3 = obj1]
- 6) 后面在集合，中hashCode 如果需要的话，也会重写

### ● toString方法

#### ✓ 基本介绍

默认返回: 全类名+@+哈希值的十六进制, 【查看Object 的 toString方法】  
子类往往重写toString方法, 用于返回对象的属性信息

```
public String toString() {  
    return getClass().getName() + "@" + Integer.toHexString(hashCode());  
}
```

- ✓ 重写toString方法, 打印对象或拼接对象时, 都会自动调用该对象的toString形式。  
案例演示: Monster [name, job, sal] 案例: ToString.java
- ✓ 当直接输出一个对象时, toString 方法会被默认的调用

### ● finalize方法

#### //Finalize.java

1. 当对象被回收时, 系统自动调用该对象的finalize方法。子类可以重写该方法, 做一些释放资源的操作【演示】
2. 什么时候被回收: 当某个对象没有任何引用时, 则jvm就认为这个对象是一个垃圾对象, 就会使用垃圾回收机制来销毁该对象, 在销毁该对象前, 会先调用 finalize方法。
3. 垃圾回收机制的调用, 是由系统来决定(即有自己的GC算法), 也可以通过 System.gc() 主动触发垃圾回收机制, 测试: Car [name]

老韩提示: 我们在实际开发中, 几乎不会运用 finalize, 所以更多就是为了应付面试。



# P328-p334断点调试

2021年5月19日 15:33

## 断点调试

### ● 一个实际需求

1. 在开发中，新手程序员在查找错误时，这时老程序员就会温馨提示，可以用断点调试，一步一步的看源码执行的过程，从而发现错误所在。
2. 重要提示：在断点调试过程中，是运行状态，是以对象的运行类型来执行的。

### ● 断点调试介绍

1. 断点调试是指在程序的某一行设置一个断点，调试时，程序运行到这一行就会停住，然后你可以一步一步往下调试，调试过程中可以看各个变量当前的值，出错的话，调试到出错的代码行即显示错误，停下。进行分析从而找到这个Bug
2. 断点调试是程序员必须掌握的技能。
3. 断点调试也能帮助我们查看java底层源代码的执行过程，提高程序员的Java水平。

## 断点调试

### ● 断点调试的快捷键：

F7(跳入) F8(跳过) shift+F8(跳出) F9(resume,执行到下一个断点)

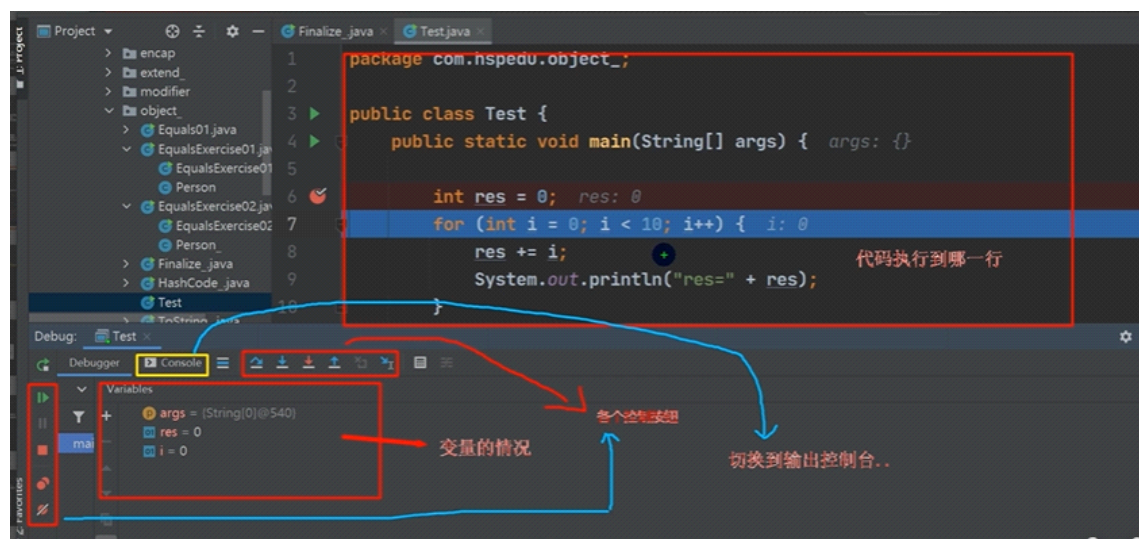
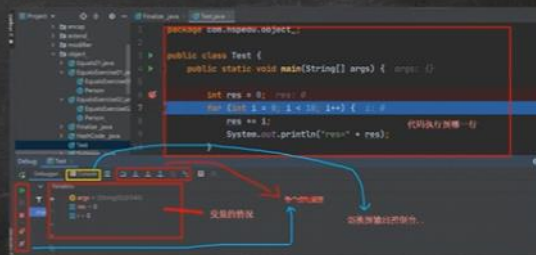
F7: 跳入方法内

F8: 逐行执行代码。

shift+F8: 跳出方法

### ● 断点调试应用案例

看几段代码，演示调试过程。



## ● 断点调试应用案例

- ✓ 案例1 `com.hspedu.debug_包 Debug01.java`  
看一下变量的变化情况

```
public static void main(String[] args) {  
  
    //debug 数组  
    int sum = 0;  
    for (int i = 0; i < 10; i++) {  
        sum += i;  
        System.out.println(" i = " + i);  
        System.out.println(" sum = " + sum);  
    }  
    System.out.println("end...");  
}
```

截图+说明 F8 逐行执行代码..

- ✓ 案例2  
看一下数组越界的异常 `Debug02.java`

```
public static void main(String[] args) {  
  
    //debug 数组  
    int arr[] = new int[5];  
    for (int i = 0; i <= arr.length; i++) {  
        System.out.println("arr[" + i + "] = " + arr[i]);  
    }  
}
```

截图+说明

- ✓ 案例3  
演示如何追源码，看看java设计者是怎么实现的。(提高编程思想)。  
小技巧：将光标放在某个变量上，可以看到最新的数据。 `Debug03.java`

```
public static void main(String[] args) {  
  
    //debug 源码  
    int arr[] = {8,-1,199,70,10};  
    //排序, 插入sort的源码  
    Arrays.sort(arr);  
    for (int i = 0; i < arr.length; i++) {  
        System.out.print(arr[i] + "\t");  
    }  
}
```

截图+说明 F7(跳入)

#### ✓ 案例4

演示如何直接执行到下一个断点F9 **resume**。

**老韩小技巧:** 断点可以再debug过程中，动态的下断点

```
7~ public static void main(String[] args) {  
8  
9     //debug 源码  
10    int arr[] = {8,-1,199,70,10};  
11    //排序,进入sort的源码  
12    Arrays.sort(arr);  
13    for (int i = 0; i < arr.length; i++) {  
14        System.out.print(arr[i] + "\t");  
15    }  
16  
17    System.out.println("hello100");  
18    System.out.println("hello200");  
19    System.out.println("hello300");  
20    System.out.println("hello400");  
21    System.out.println("hello500");  
22    System.out.println("hello600");  
23    System.out.println("hello700");  
24  
25 }
```

截图+说明 **Debug04.java**

```
public static void main(String[] args) {  
    //创建对象的流程  
    //(1) 加载 Person类信息  
    //(2) 初始化 2.1默认初始化, 2.2 显式初始化 2.3 构造器初始化  
    //(3) 返回对象的地址  
    Person jack = new Person("jack", 20);  
    System.out.println(jack);  
}
```