## 回顾

抽象方法&抽象类

1. 什么时候需要定义为抽象方法?当这个方法没有办法写具体的实现的时候
2. 使用关键字abstract修饰 的方法就称为抽象方法,没有方法体,没有具体实现.
3. 一个抽象类中可以包含抽象方法和普通方法, 如果一个类中包含一个以上的抽象方法,这个类就必须定义为抽象类
4. 抽象类中有构造方法,不能用abstract修饰,不能直接实例化,父类只能被继承,由子类进行实例化:向上转型:父类 父引用=new 子类();此时,父类的引用可以调用父类的属性和方法+子类和父类的共有的属性和方法
5. 一个类继承了抽象类,就必须重写父类的所有抽象方法,如果不重写,该子类也必须定义为抽象类

## 接口

**接口（英文：Interface），在JAVA编程语言中是一个抽象类型，是抽象方法的集合，接口通常以interface来声明。一个类通过继承接口的方式，从而来继承接口的抽象方法。**

**接口并不是类，编写接口的方式和类很相似，但是它们属于不同的概念。类描述对象的属性和方法。接口则包含类要实现的方法。**

**引出问题:**一个人多角色:演员,歌手,每一个角色都是一个类,能否让人去继承每一个角色的类 否

**出现问题:**java中类是单继承,不能同时继承歌手和演员类

**解决问题:**将角色定义成接口,让类去实现多个接口,java中的接口是支持多继承的.

定义一个接口: 修饰符[public] interface 接口名[I类似类名]{

//公共常量 public static final 数据类型 常量名[大写] = 值;

//抽象方法

//默认方法/静态方法[jdk1.8才有,低于1.8是不能出现默认方法的]

}

继承的写法: extends

interface 接口名 extends 接口1,接口2{

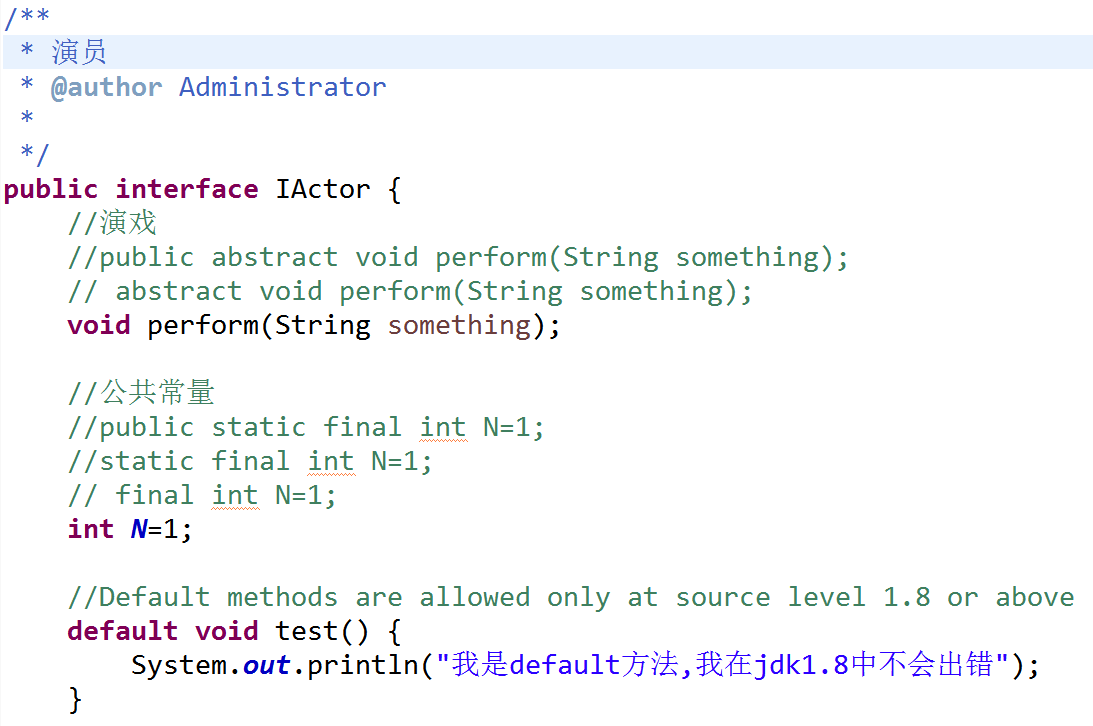
}

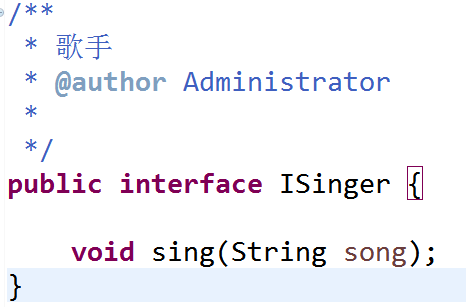
implements 实现

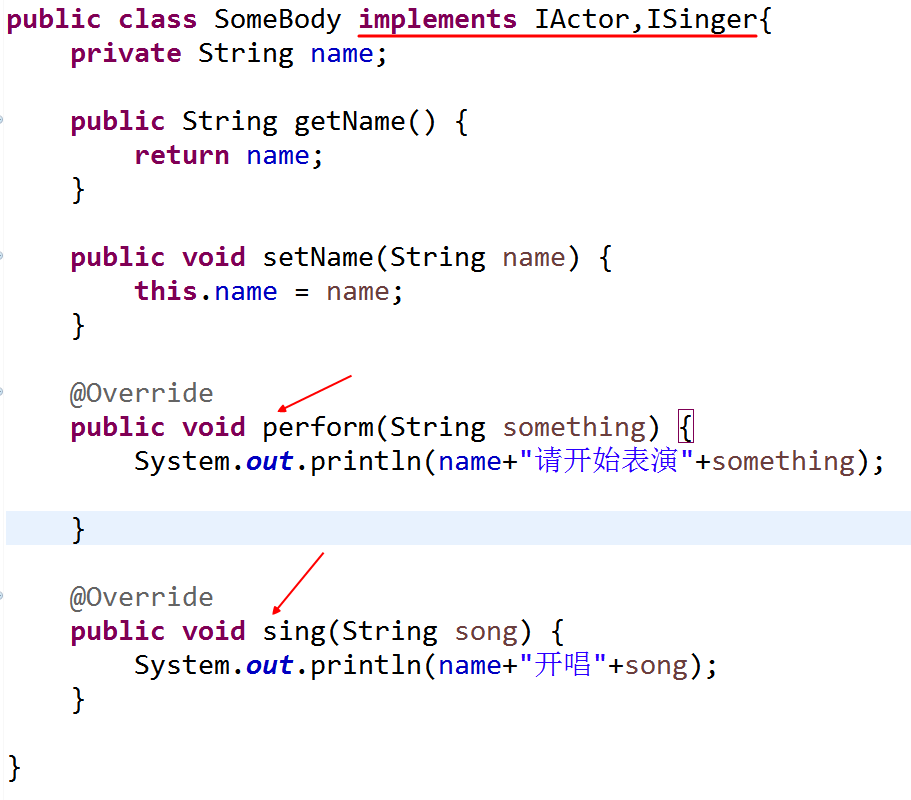
class 类名 implements 接口1,接口2{

}

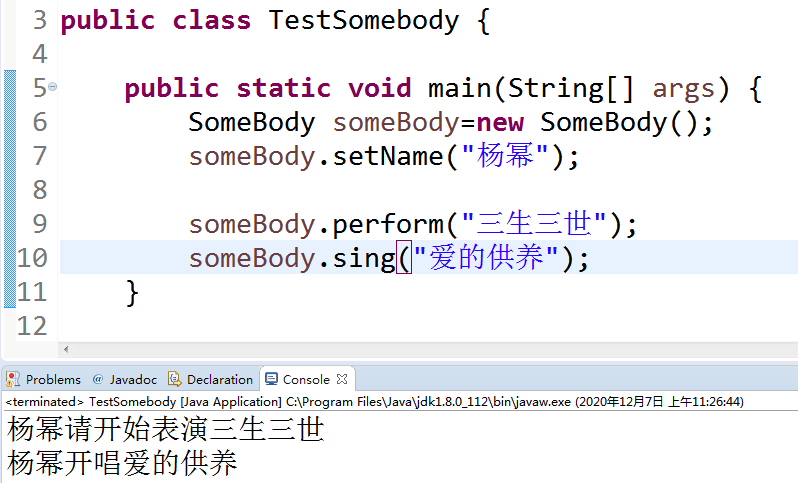
刚才的案例:



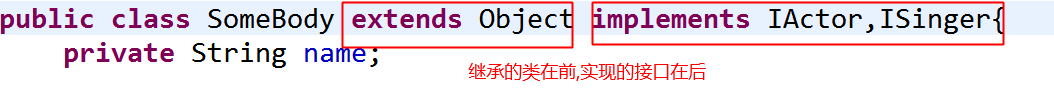


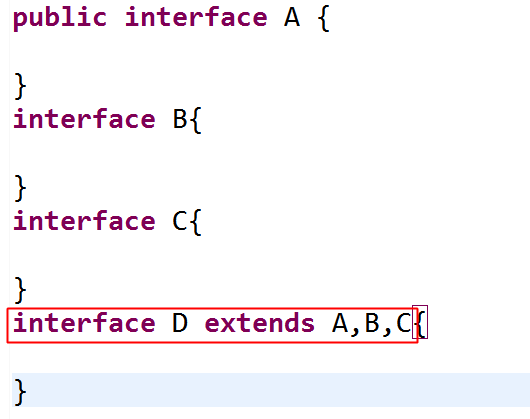


测试类:



**注意:接口的定义,接口中包含的内容,接口支持多继承的写法**





## 【面试题】抽象类和接口的区别:

1. 抽象类使用abstract关键字进行修饰,而接口使用interface进行定义。

2. 抽象类只支持单继承,接口可以支持多继承

3. 抽象类中的成员变量可以是各种类型的，而接口中的成员变量只能是public static final类型的静态常量；

4. 抽象类可以有抽象方法和普通方法,也可以有变量和常量,可以有构造方法

5. 接口中可以包含抽象方法和静态常量,如果是jdk1.8的话,可以包含默认方法 和静态方法,没有构造方法

## 多态再说明:

#### 实现方式:

1. [编译时多态]:对象和方法在程序运行之前就已经确定了,执行的时候不会存在动态判断的情况了,这种称为编译时多态; 实现方式就是方法的重载;

Class Pig{

Public void eat(){

Sysout(“吃猪食”);

}

Public void eat(String food){ //面包

Sysout(“吃”+food);

}

Public void eat(String food,String drink){ //面包 茅台

Sysout(“吃”+food+”,配…”+drink); //

}

}

1. [运行时多态]使用继承: 创建父类 ,创建子类,在子类中重写父类的方法,应用场景中将父类作为参数传递,实际调用的传递具体的对象(子类对象) ,那么实际执行的时候,根据不同的子类对象来执行特定子类的方法;
2. [运行时多态]使用接口来实现: 创建父接口,创建接口的实现类;在实现类中实现接口中的抽象方法,应用场景中将父接口作为参数,实际调用的时候传递具体的实现类的对象, ,那么实际执行的时候,根据不同的子类对象来执行特定子类的方法;(老师在上课的案例修改成使用接口)
3. 优点：

可替换性（substitutability）

多态对已存在代码具有可替换性

**可扩充性（extensibility）**

**多态对代码具有可扩充性。增加新的子类不影响已存在类的多态性、继承性，以及其他特性的运行和操作。实际上新加子类更容易获得多态功能.**

接口性（interface-ability）

多态是超类通过方法签名，向子类提供了一个共同接口，由子类来完善或者覆盖它而实现的

灵活性（flexibility）

它在应用中体现了灵活多样的操作，提高了使用效率

简化性（simplicity）

多态简化对应用软件的代码编写和修改过程，尤其在处理大量对象的运算和操作时，这个特点尤为突出和重要

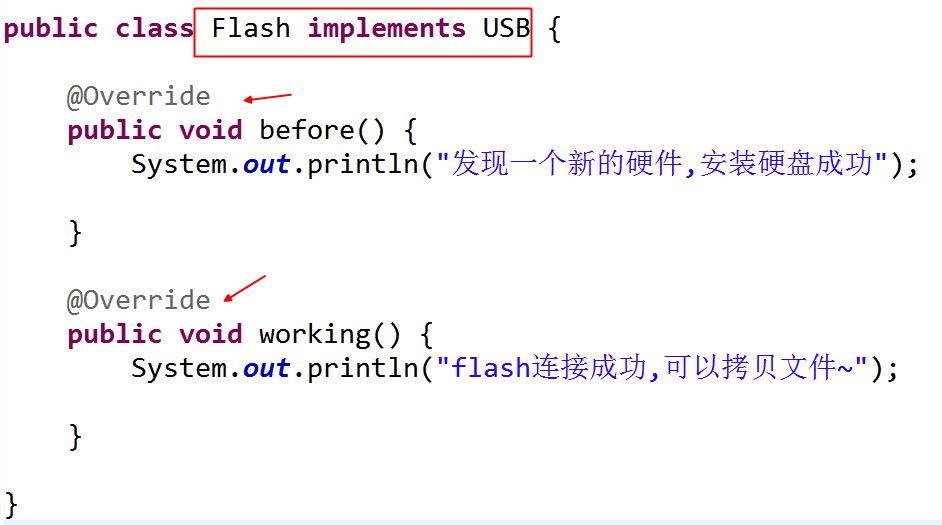
## 课堂练习

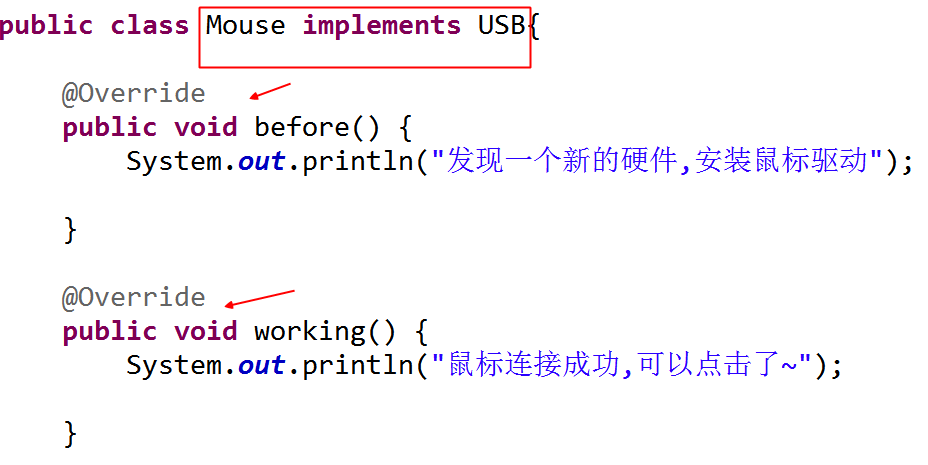
模拟USB接口的工作: USB是一个接口，键盘、鼠标、U盘都可以插在USB接口上 ；创建一个计算机，安装设备，让这个设备工作就可以了;

1. 创建一个接口,定义2个抽象方法(准备工作之前,开始工作)



1. 创建接口的实现类:键盘、鼠标、U盘,去实现接口中的抽象方法

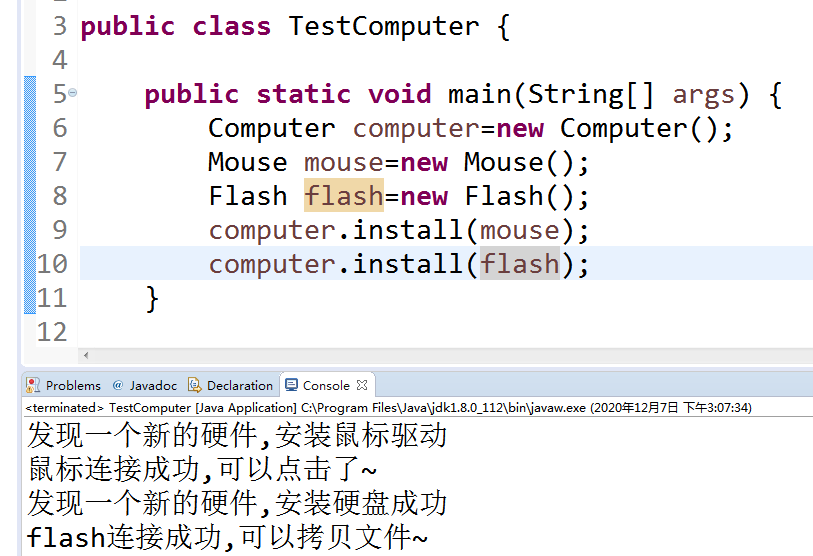




3.创建一个计算机类,安装usb设备 install(接口){调用接口的2个方法}



4. 测试:



## 访问修饰符

1. 作用：在Java中实现信息隐藏的是访问控制权限机制；

访问控制权限包括4个访问修饰符：public、protected、缺省 和private。

1. 权限如下：

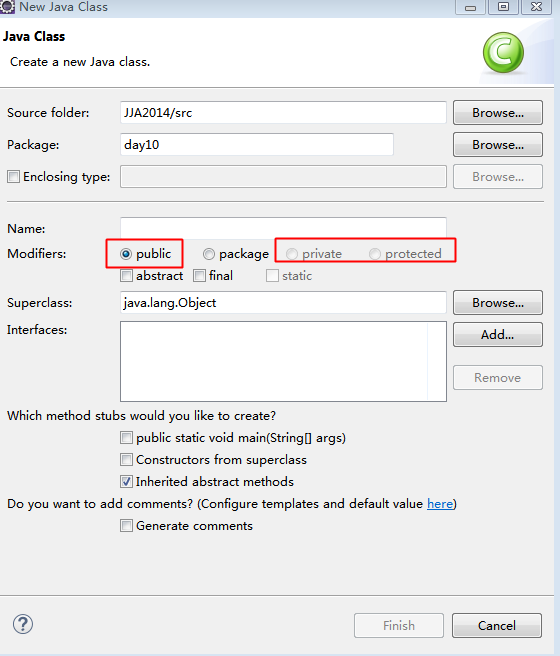
public: 不受任何限制，本类或非本类均可随意访问 (公开的,谁都可以调,可跨包,全友好)

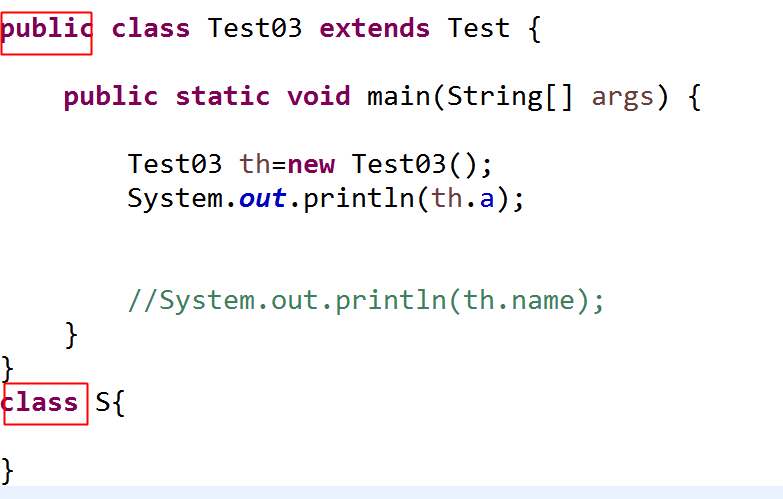
protected：同一个包中的其它类也可访问（同包友好）

本类及其子类可以访问：允许不同包中的子类访问父类的保护方法和保护属性。(跨包父子友好)

1. 缺省(不使用任何修饰符)：只有相同包中的类可以访问。 (同包友好)
2. private：只有本类可以访问，其余都不可以。(本类友好)

对类使用的访问修饰符只有public和缺省两种，对成员的修饰符四个都可以。





## 非访问修饰符-三大关键字

### 抽象:abstract

1,一个类中只要有一个或一个以上抽象方法,必须用abstract声明为抽象类;

2,抽象类中可以有具体的实现方法,也可以没有抽象方法;

3,抽象类中的所有的抽象方法必须被它的子类实现,如果子类没有实现,则该子类继续为抽象类

4,抽象类不能被直接实例化,但可以由抽象父类指向的子类实例来调用抽象父类中的具体实现方法;通常作为一种默认行为;

5,要使用抽象类中的方法,必须有一个子类继承于这个抽象类,并实现抽象类中的抽象方法,通过子类的实例去调用;

### 最终的:final

1,可用于修饰:成员变量,非抽象类(不能与abstract同时出现),非抽象的成员方法,以及方法参数

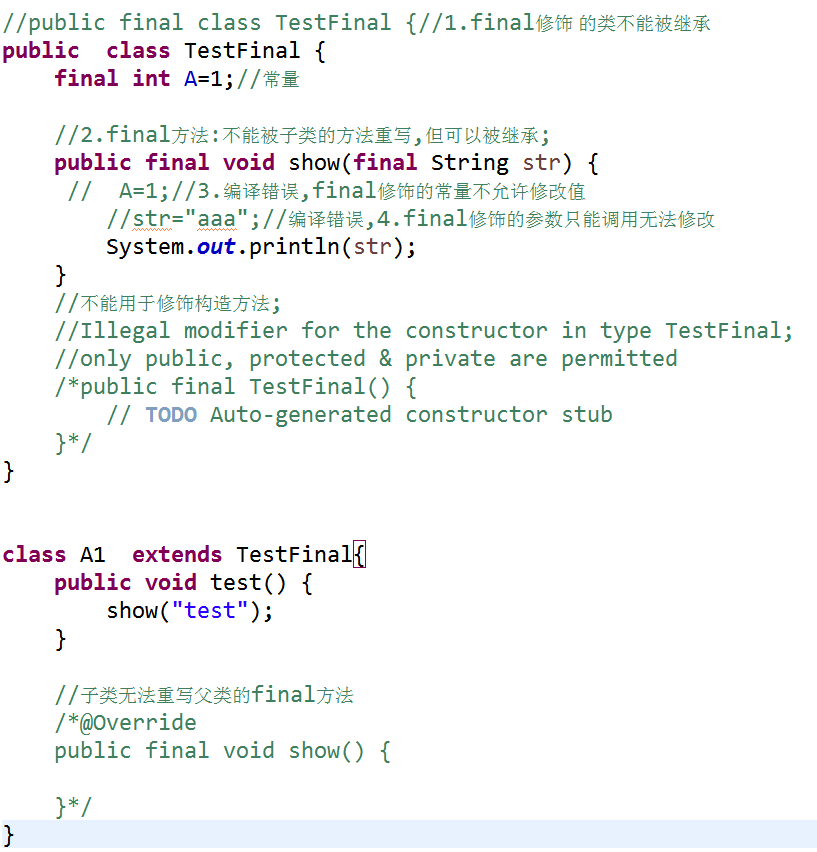
2,final类:表示该类不能被继承,没有子类

3,final方法:不能被子类的方法重写,但可以被继承;

4,final变量:表示常量,只能赋值一次,赋值后不能被修改.final变量必须定义初始化;

5,final参数:只能使用该参数,不能修改该参数的值;

6,final不能用于修饰构造方法;



## 静态:static

class 类{

static int a; //静态变量

static void method(); //静态方法

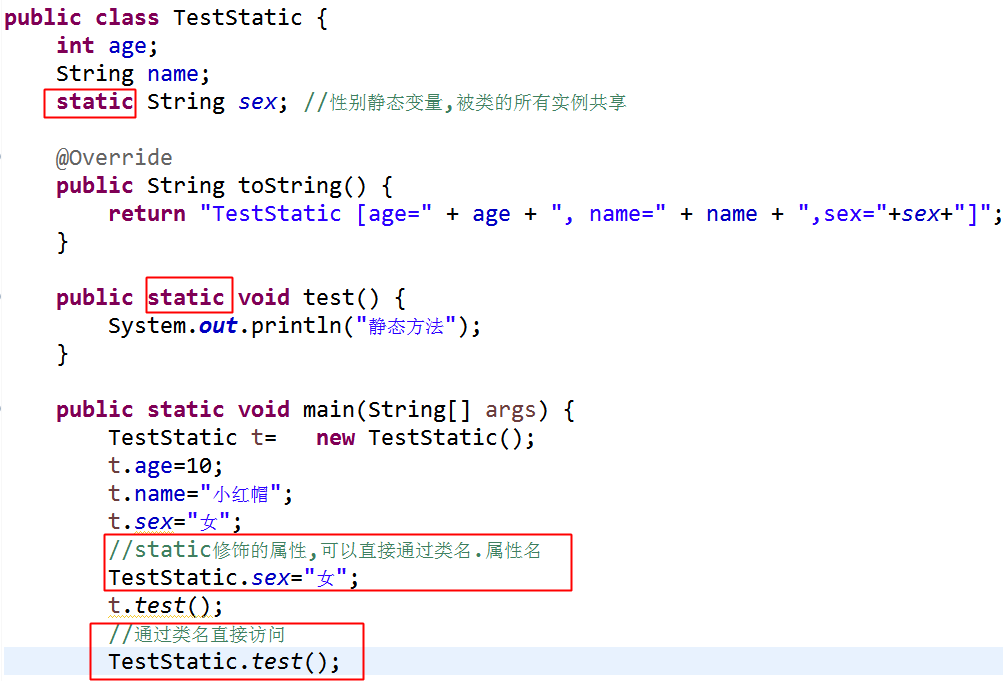
static{...} //静态块

}

1,static修饰的成员变量它不依赖类特定的实例，被类的所有实例共享，在内存中唯一。

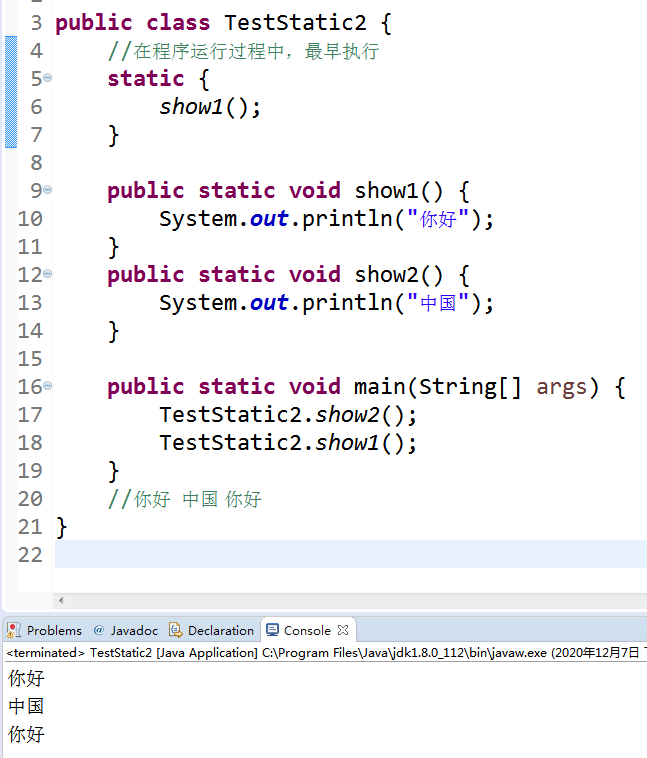


2,static变量和static方法一般是通过类名直接访问,但也可以通过类的实例来访问(不推荐这种访问方式)

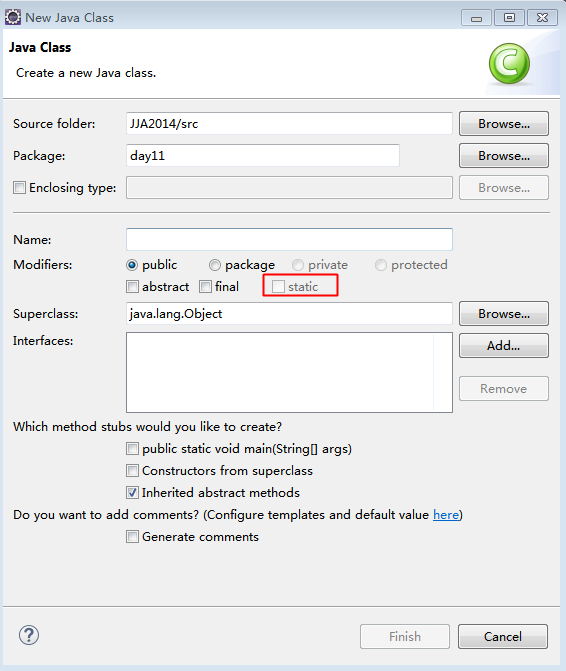


3,非静态的方法中可以访问（调用）静态方法和静态变量，但是静态方法中不能访问非静态方法和非静态变量

4静态块：在程序运行过程中，最早执行：初始化static修饰的成员时，可以将他们统一放在一个以static开始，用花括号包裹起来的块状语句中。



1. static不能修饰主类以及构造方法;



6.用public修饰的static变量和static方法,在任何地方都可以通过类名直接来访问,但用private修饰的static变量和static方法,只能在声明的本类方法及静态块中访问,在静态块不能用this访问,因为this属于非静态变量.

注意：static abstract final三个关键字是否可以混用?

## 三个关键字混搭使用：

①static和final同时使用

1,static final用来修饰成员变量和成员方法，可简单理解为“全局常量”！

2,对于变量，表示一旦给值就不可修改，并且通过类名可以访问。

3,对于方法，表示不可重写，并且可以通过类名直接访问。

②为什么不能同时加abstract与static呢？

static是静态的，就是在编译时已经是确定的东西，不能是抽象的。  
因为static修饰的方法是静态方法，其可以直接被类所调用。而abstract修饰的方法为抽象方法，即无方法体的方法，不能够被直接调用，需要在子类或实现类中去编写完整的方法处理逻辑后才能使用。

③为什么不能同时加abstract与final呢？

因为final是最终的，不可以被继承，不可以被重写。  
而abstract是抽象的，只能被继承，被重写。

应用广泛： 将某一个不会修改的变量设置为静态常量 public static final String username ="root"

# 总结:

1. 接口:公共常量/抽象方法/jdk1.8:默认方法,静态方法
2. 多态总结:实现3种:重载,使用继承,使用接口
3. 访问修饰符:4个
4. 三大关键字: