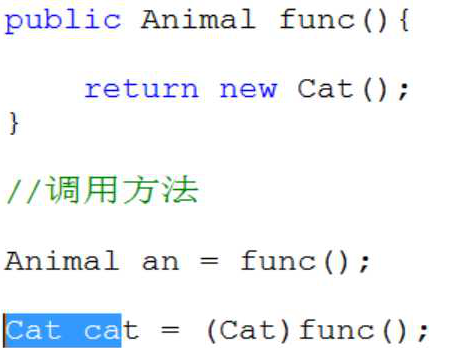
1 匿名内部类要么实现一个接口要么实现一个类，不能既实现类又继承接口。

2 多态只有行为没有变量。

3 类，抽象类和接口都有多态等。

4 必须返回animal再向下转型：



5 接口做方法形参很常见，接口做返回值不常见。

6 今天的重点是ststic,final和内部类。

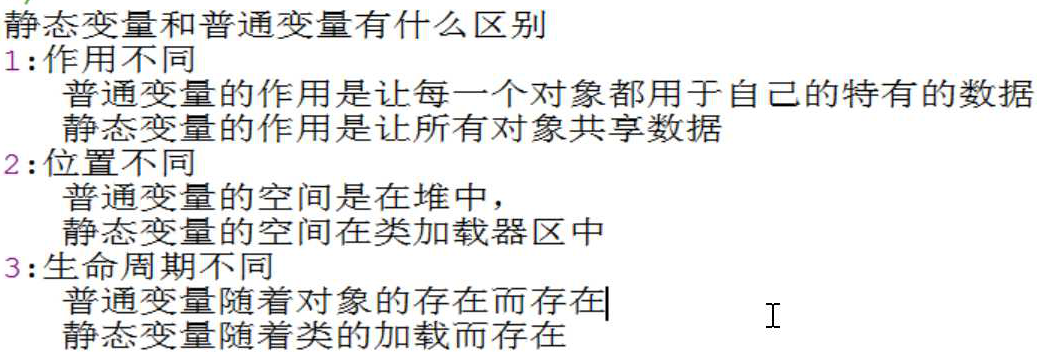
7 static可以修饰变量和方法。

8 static修饰的变量不属于任何一个对象，而是属于所有对象共享的（属于类）。

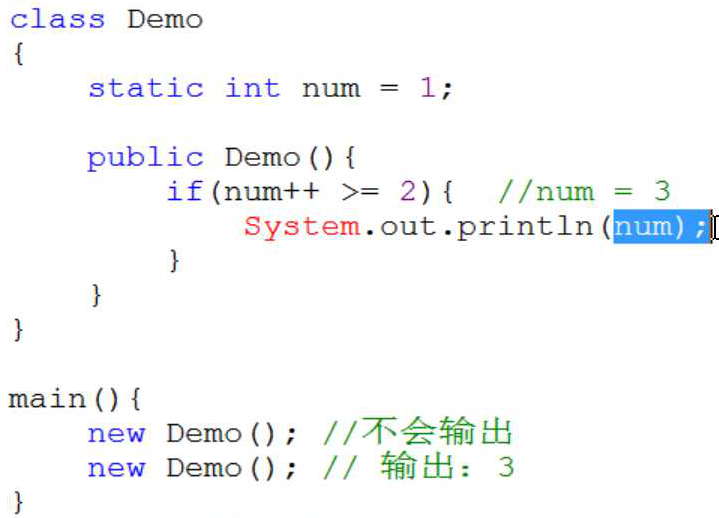
9 static修饰的变量一般用public而不是private。(原因见第十条)

10 静态变量（static）可以直接用类名调用，也可以用对象对象访问。但是最好用类名访问。**但是静态变量加private修饰之后不能用类名访问，所以不能用private。**

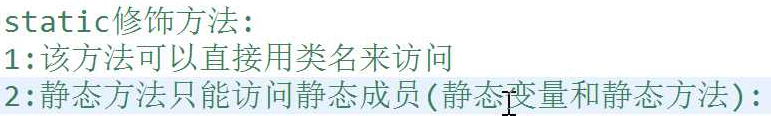
11 普通变量和静态变量的区别：



12 static变量并非不能改变，只是共享而已。



13 static修饰方法的特点：









14 普通方法叫实例方法，静态方法叫类方法。变量同理。

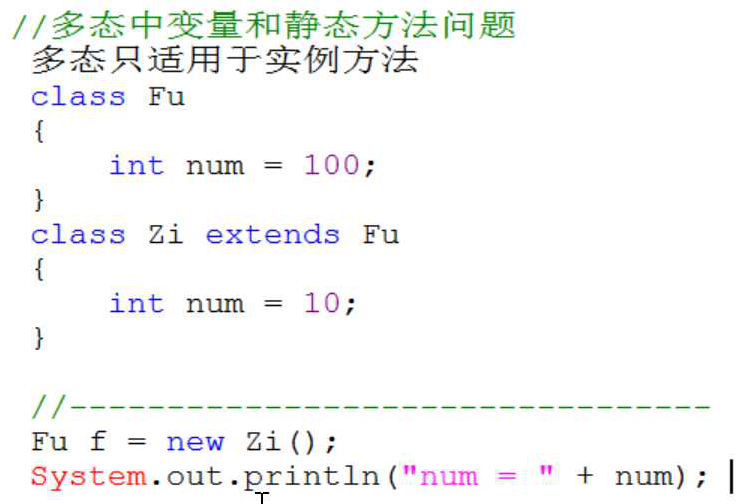
15  **多态只适用于实例方法。（见18条和19条）！！！！**

**16 什么时候应该用static方法（总之就是保证不引用成员变量就行）：**

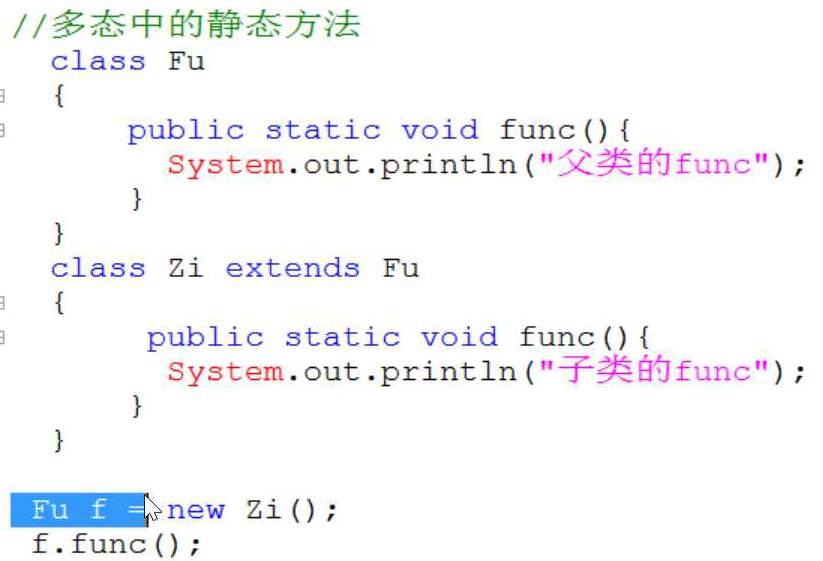
1. **没有成员变量，只有成员方法。**
2. **某个方法不引用成员变量。**

**17 将构造方法私有可以阻止创建对象。**

18 变量不符合多态的原理，因为编译和运行都看左边（实际开发遇不到）：

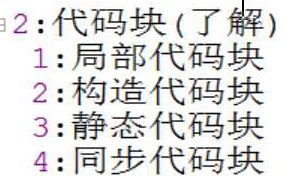


19 多态同样不适合于静态方法，仍然是编译和运行都看左边：

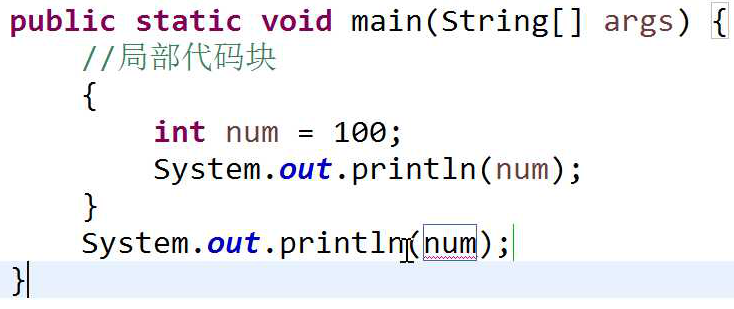


20  **Arrays，Math等叫工具类。工具类里面是静态方法。**

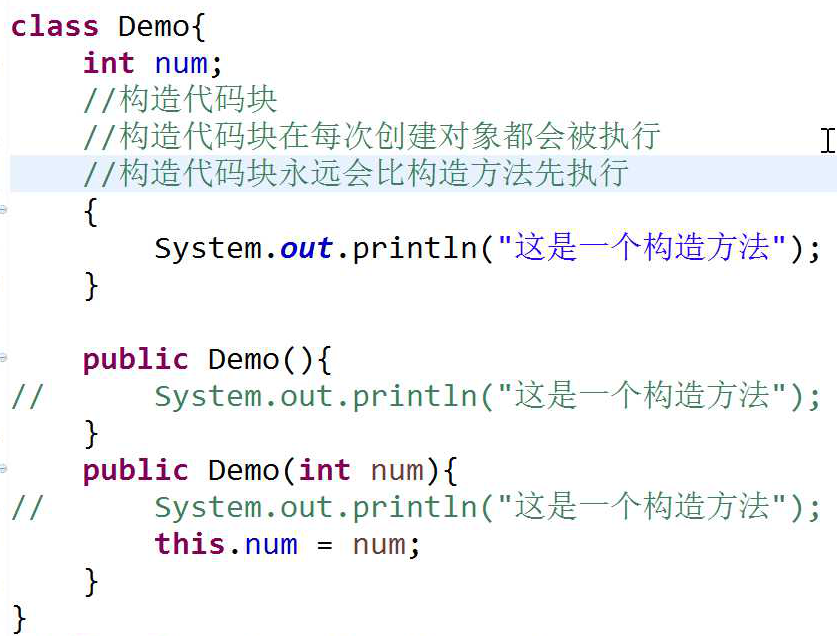
21 静态代码块（面试用的多，实际用得少）



22 出了局部代码块变量就不存在了，因此作用是尽快释放内存（释放该变量的空间）：

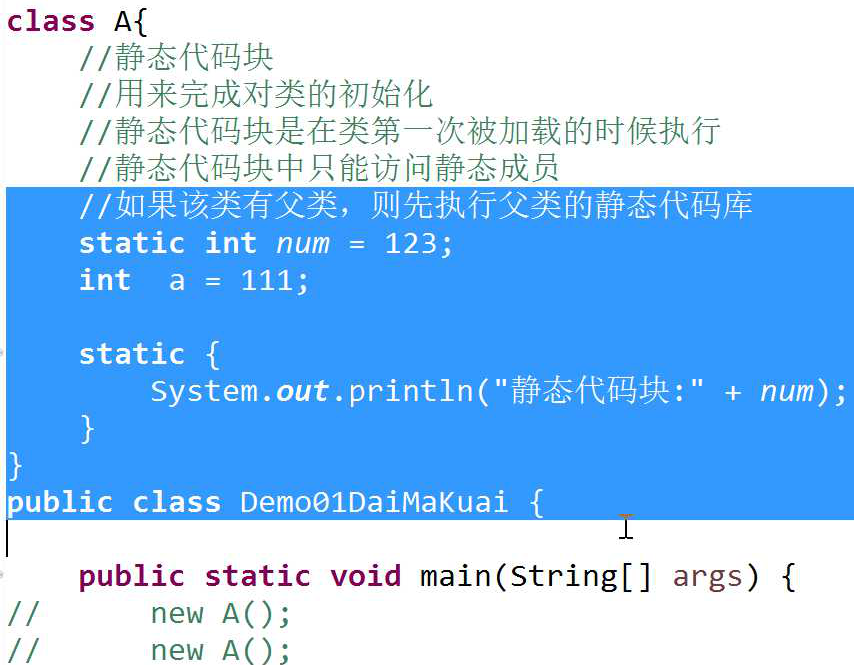


23 构造代码块（区分构造代码块和静态代码块）：



**子类还能继承父类的构造代码块**，并且会先执行父类的构造代码块。

24 静态代码块：在构造代码块前面加static。



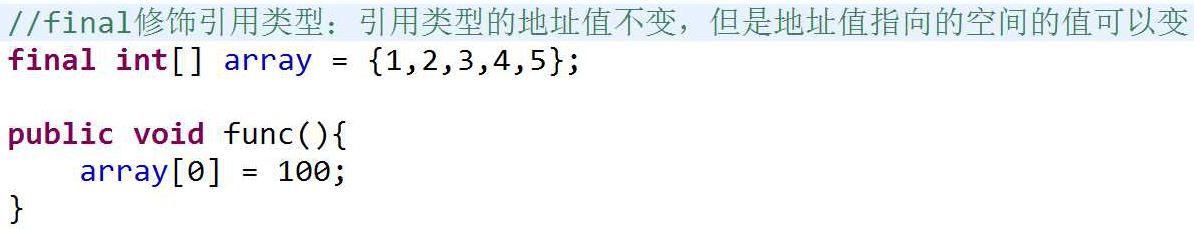
25 final修饰的类不能被继承（因为继承意味着修改和覆盖）。当不想类中的方法被重写（打破类的封装性），就会用final修饰类。

26 final还可以修饰方法。则该方法不能被重写。



27 final修饰变量：该变量只能被赋值一次。因为只能赋值一次，所以一般不通过方法赋值。一般会直接赋值或者在构造代码块中赋值。

28 final修饰引用类型变量：



29 final将变量变为常量，常量是用大写的。用final修饰也会用public 和static修饰，以避免创建实例，并保证权限（因为并不怕被修改，因为无法修改）



30 final修饰的集合仍然可以改变（只要地址不变），解决的方法是动态代理。

31 final修饰成员变量不赋值会报错，修饰局部变量只有在开始使用时仍未赋值才会报错，因为前者（成员变量）有默认值（默认值可能不是作者想要的却又自动赋值为默认值），后者（局部变量）没有默认值。

32 final在编译时会被丢弃。并且应用该变量的地方会被直接替换成常量。证明方式：将.class文件反编译。

33 包的作用是管理类文件。包其实就是文件夹。

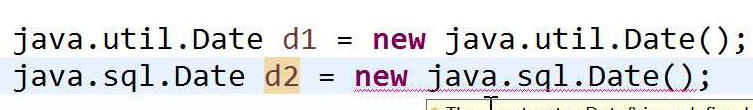
34 发给用户的是.class文件而不是.java文件。

35 报名还可以用于区分类名。因此包名要全球唯一，因此用公司的域名反过来写。类的全名是包名.类名。

36 包必须小写。

37 导包要精确到类名，可以加\*，不过最好不要这样，很占空间。

38 当两个包同时有相同类名时，不用导包（因为导包也无法区分），直接写类的全名（即java.包名.类名）（必须写全名）：



52 访问权限（无关类指的是非子类）（其中空的（default）最常用）：

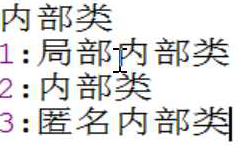


53



54 abstract和static不能放一起。

55 内部类的分类：



56 局部内部类主要是应付面试。局部内部类是定义在**方法中**的内部类。

57 方法中不能定义方法，因此主方法中不能定义方法。

58 num必须加final，因为对象在堆中，变量在栈中，防止变量被销毁之后对象仍然引用他（开发不用，面试用）：

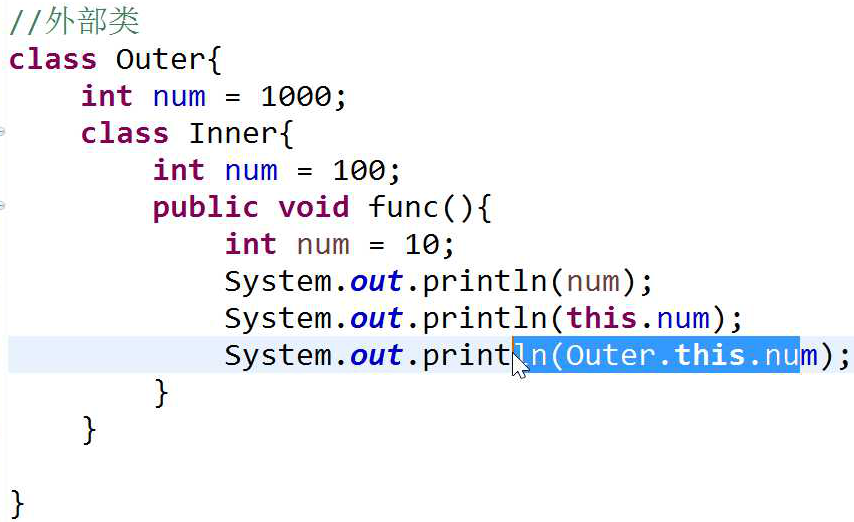


final的变量会被常量替换掉，因此不会访问内存。

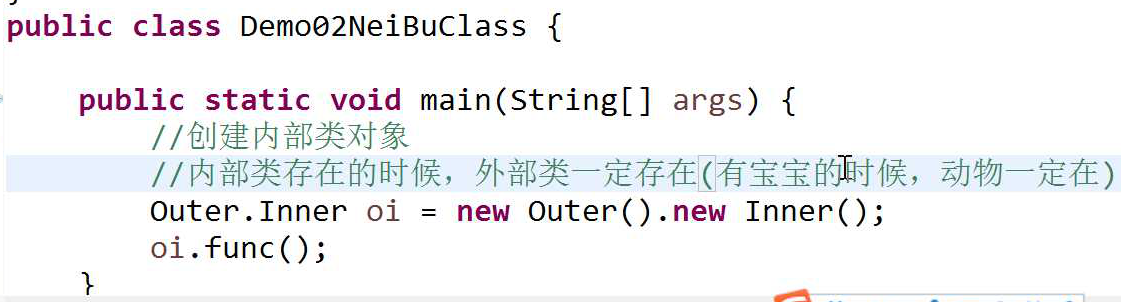
59 final放在数据类型的前面，例如final int

60 外部类可以有多个内部类，内部类还可以有内部类。

61  **内部类访问外部类用Outer.this：**

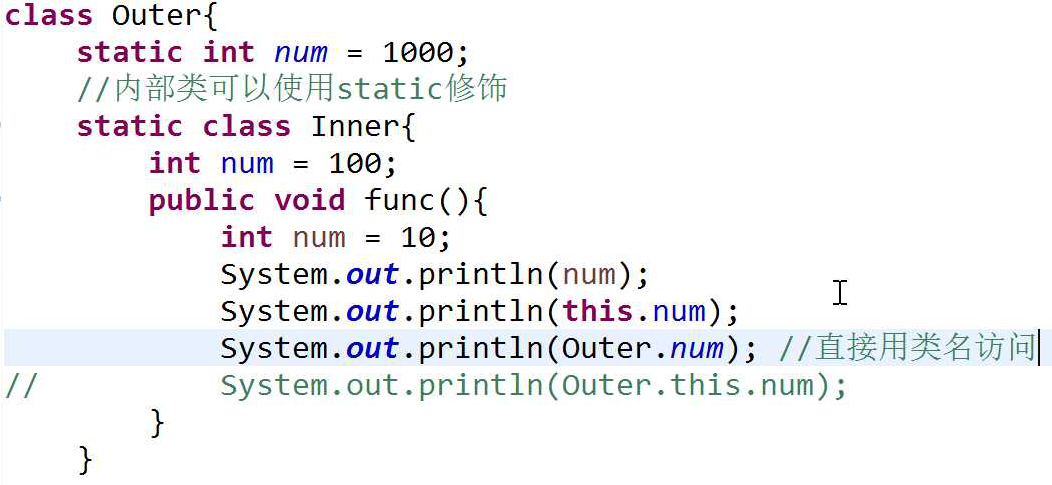


62 创建内部类对象：





63 内部类可以用static修饰，外部类不可以，因为前者算是类的成员。同时静态了里面又不能有非静态，静态（变量和方法）不能用this访问，只能用类名访问：

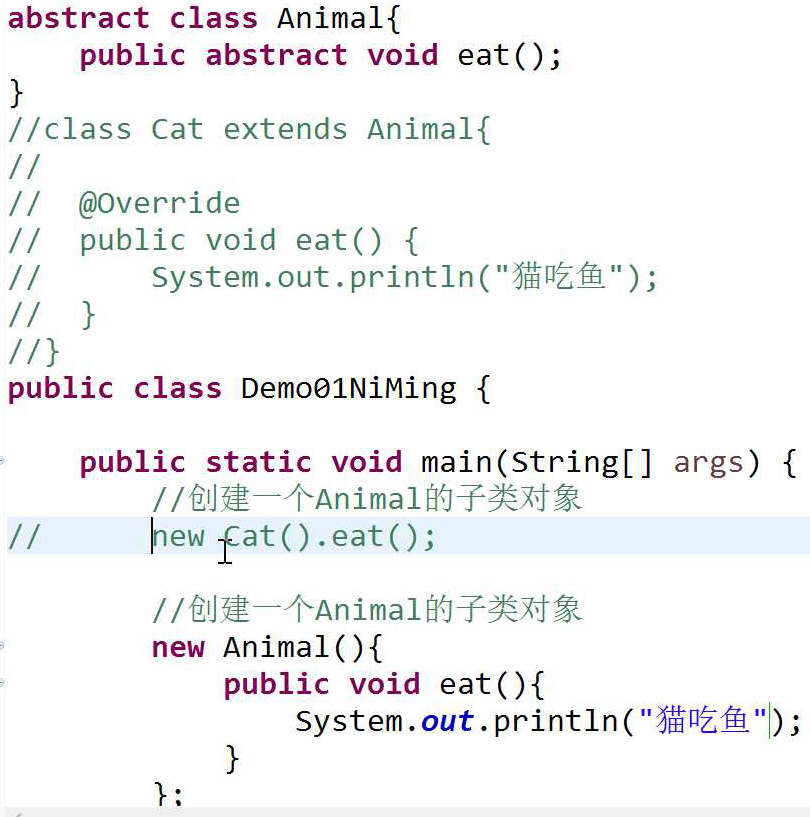


64 内部类可以私有，表示只能在外部类的内部使用和创建对象（同外部类的成员）。

65 内部类静态之后的创建对象的方法（外部类只有一个类名，未创建对象）：



66 创建抽象类的匿名内部类（其中抽象类表示的是他的子类，而不是抽象类本身）：



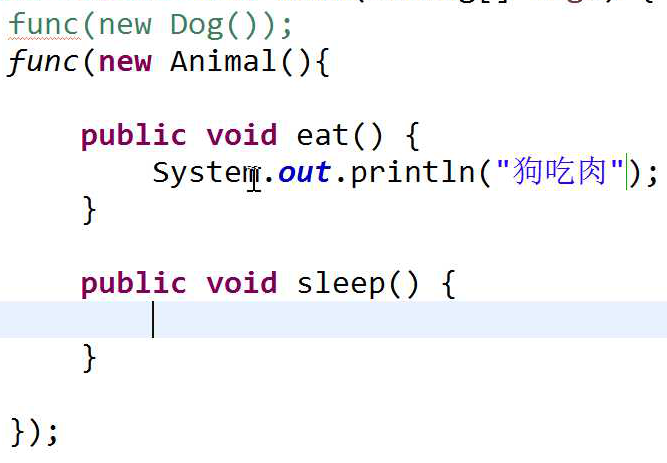
67 创建抽象类的匿名内部类时alt+反斜杠会自动补全抽象类里面的抽象方法。

68 若是不匿名就成了多态（虽然等号两边都写的Animal，但确实是Animal的子类而不是他本身）：



69 **匿名内部类的分类：接口的匿名内部类和抽象类的匿名内部类。**

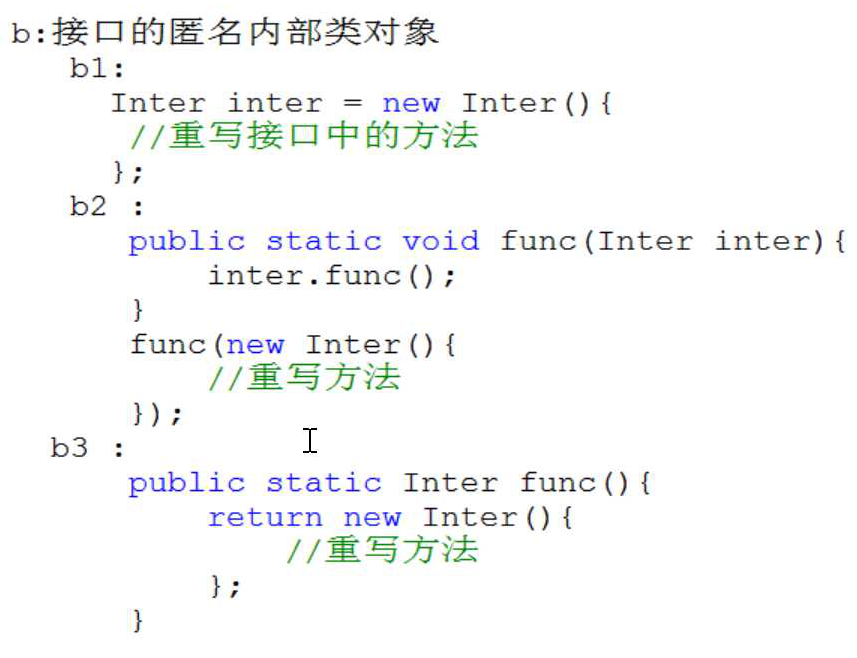
70 匿名内部类做参数（实参）（可以是普通类或父类也可以是抽象类或接口）：



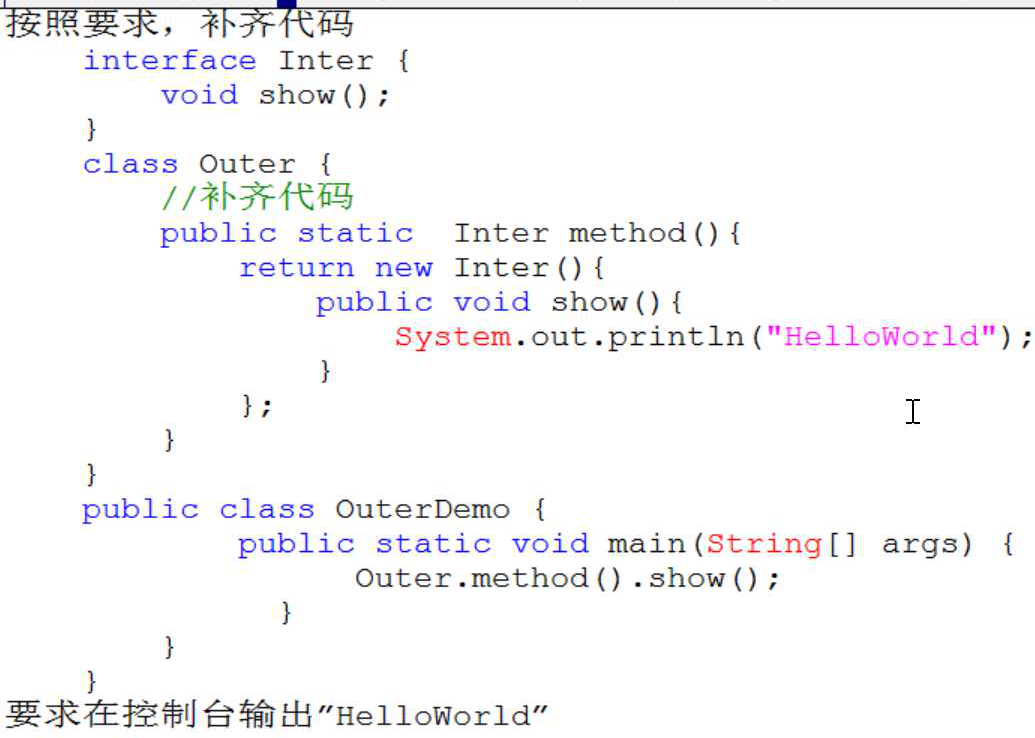
71 总之，任何出现类的地方都可以用匿名内部类代替。

72 匿名内部类降低了代码的可读性，尤其是有多个参数的时候，只有最后一个参数用匿名内部类才清晰。

73 B2用的是最多的:



74 练习题（补全代码）（注意加上public）：？？？？？？？

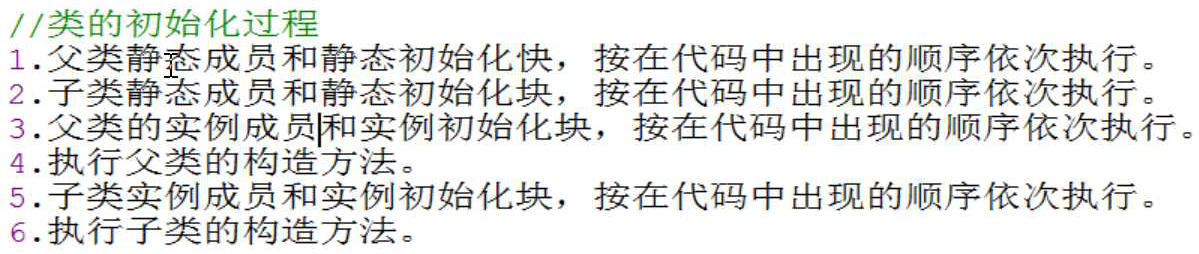


75 静态和匿名内部类用到的较多。

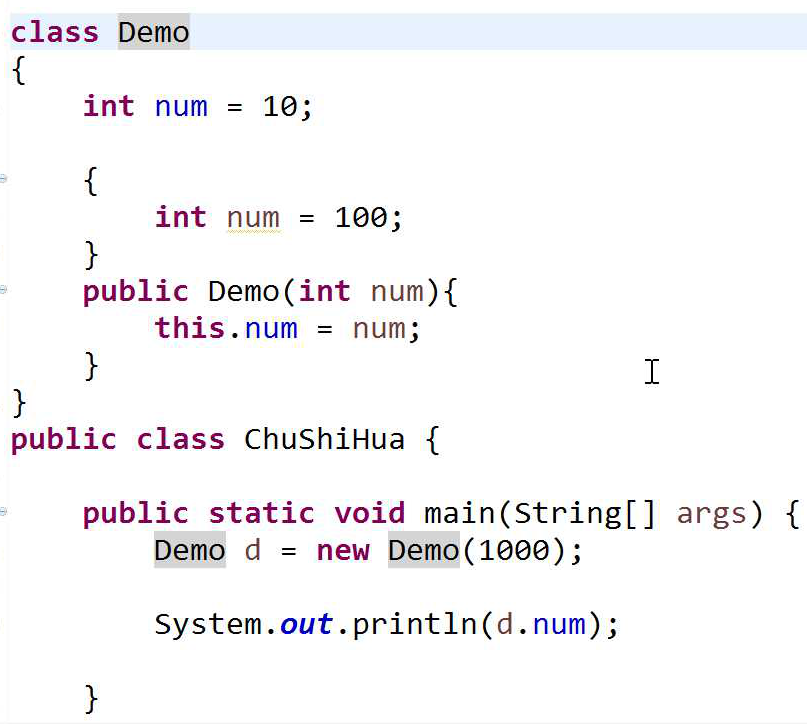
76 static静态用于数据共享。一个人改了其他人也受影响，可以加final防止修改。

77 **静态方法只能访问静态成员，静态代码块只能访问静态成员，静态内部类只能访问静态成员（指的是外部类中的静态成员，其实在静态内部类中各种成员都可以引用）。**

**78 类的初始化过程（静态优先，父类优先，成员和初始化块优先于构造方法（因为有成员才能进行初始化））：（main方法也算成员）**



79 练习题：



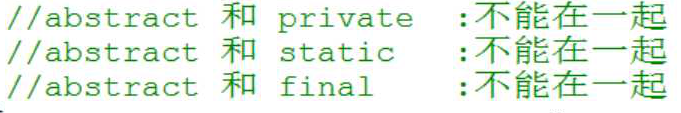
定义两个num没出错是因为代码块里的变成了局部变量，相当于一个成员变量，还有一个同名的局部变量。

80 下面这两种谁在前面先执行谁（按顺序）：



81 修饰符的总结有文档

82 这三对不能放一起：



83 protected是修饰成员的，不能修饰外部类。

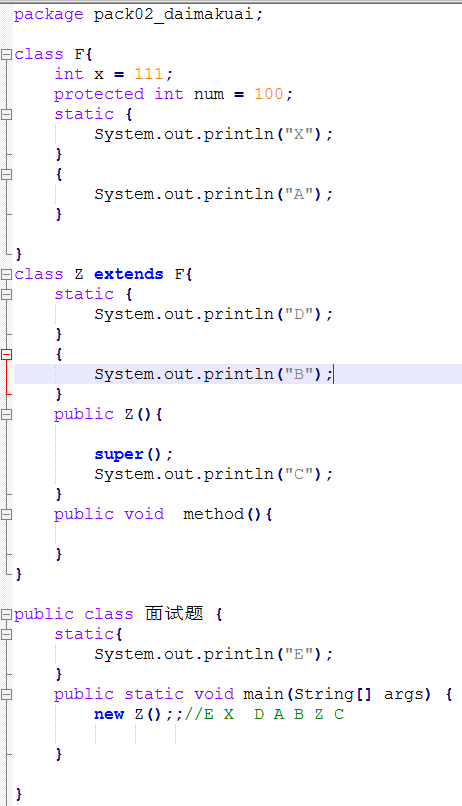
84 一个类由六部分组成：成员变量，成员方法，**构造方法，构造代码块，**静态代码块，内部类。

85 修饰符的位置没有严格的规定，不过一般是按顺序的。

86 带参数的构造函数应该不会自动去父类找相应的带参数的构造函数，要用super()。

87 overload:重载。**Override:重写。该注解的作用是强制让子类覆写父类的方法，例如返回值类型必须相同等（否则就不叫覆写了）。**

88 **代码块的面试题（关于各种代码块的执行顺序）：父类的static比子类的static优先级高，但是父类的非static优先级低于子类的static。（总之就是static优先级高）**



89 非静态方法可以访问静态变量。

90 被final修饰的类不能被继承但可以继承别的类。

91 包的声明要放到有效代码的第一行（无效代码指的是注释）。

92