1 重写方法要复制粘贴，不能自己写，避免写错。

2 父子的同名方法都是私有的话不算重写，因为无法访问（少见）。

3 构造方法只能在new对象时被调用。

4 查看类时，ctrl+左键点击类名就可以查看类的源码。

5 接口中的方法是额外行为，不是类中的行为，比如学生喝酒。表示扩展的能力。

6 implements和extends都是复数。

7 类实现接口。

8 一个类文件中只能有一个类是public。

9 接口中的变量都是常量，值不能被修改。是静态的。



10 class文件是二进制，反编译就是从二进制到源码。

11 接口中的变量用得比较少，主要是方法。

12 可以在继承类的同时实现接口。

13 接口的名字以Inter结尾表示他是接口，与类相区分。

14 类可以在创建时添加接口，并且可以有多个接口。类中会自动出现需要重写的接口中的抽象方法。若不重写的话该类是抽象类。

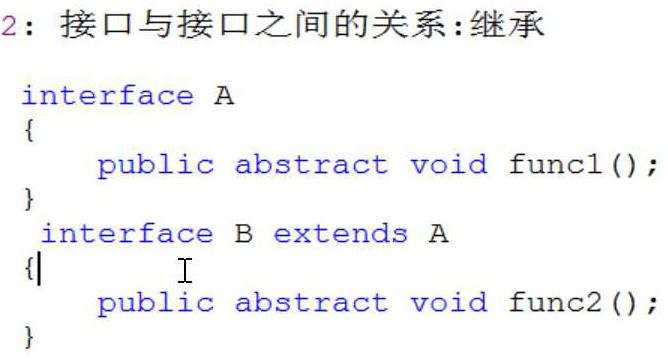
15 接口之间用逗号相连

16 注意继承写在接口的前面：



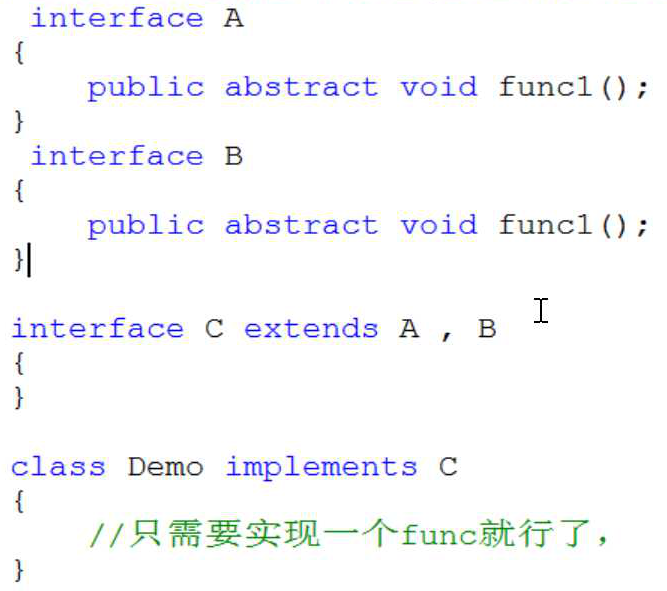
17 接口只是方法的声明，符合高内聚，低耦合的要求。

18 接口与接口之间的关系是继承。



19 一个接口可以继承多个接口，仍然用逗号间隔。

20 多个接口中有相同的方法时只需要实现一个，因为没有方法体：



不过实际开发中不会出现。

21 接口中的方法父类中已经实现了的话不用再重写，因为父类已经实现了（并且子类已经继承了）。

22 **接口中方法的默认权限是public，而类中的默认权限不是（或者小于）public，因此类和他实现的接口都不加前缀是就会报错，因为权限缩小了。**

23 适配器模式：若类不想实现接口中的所有方法时，可以设置一个实现接口部分方法的中间类，然后将该中间类作为父类。其余未实现的方法直接加上空括号。一般不用自己写，系统会提供适配器。

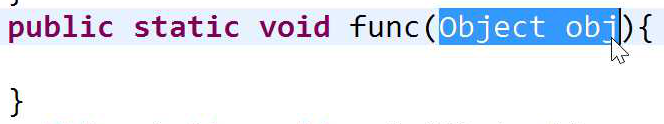
24 接口和抽象类的区别：

抽象类中可以有非抽象的方法。接口方法描述的是额外行为，抽象类描述的是本质特性。

25 多态：父类的引用指向子类对象。调用方法时执行的是子类的方法。

26 多态的作用：1避免了方法重载。提高代码的扩展性和兼容性。2作为返回值：弗雷作为返回值类型，可以返回任意子类。

27 万能方法（可以传入任何子类）：



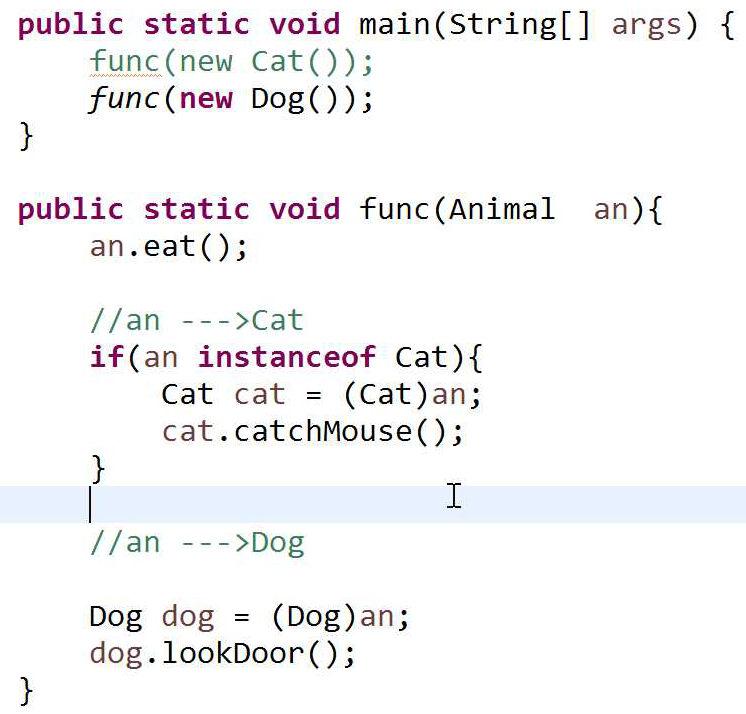
28 多态不能直接调用子类的特有行为，因为编译看左边，运行看右边。解决方法是向下转型（强制转换）。

29 编译的时候只看右边是因为编译的时候类文件还不存在。运行看右边是因为只有对象才能调用方法。

30 強转的底层实现是将地址赋给了不同类型的变量，不同的变量可以访问不同的内存区域。

31 main是static方法，他调用的方法必须是static静态方法。

32 instanceof关键字判断类型：

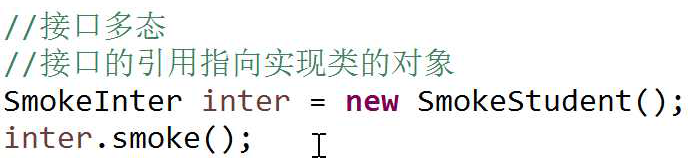




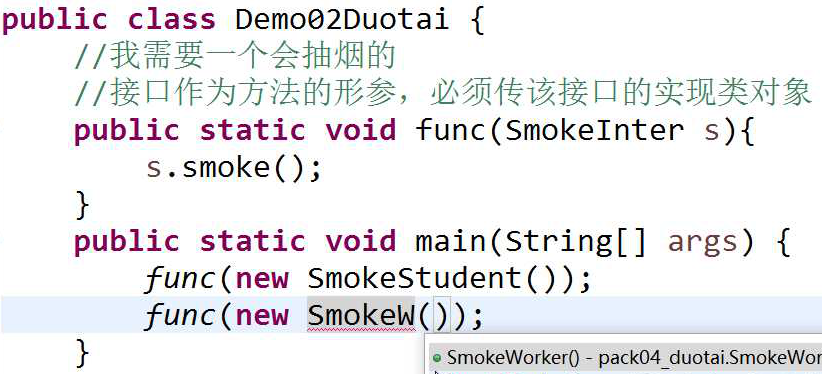
33 instanceof关键字用于反射，来判断到底是哪个子类对象。

34 多态的前提（应用场景）是继承，方法重写和父类引用指向子类对象。

35 接口的多态：

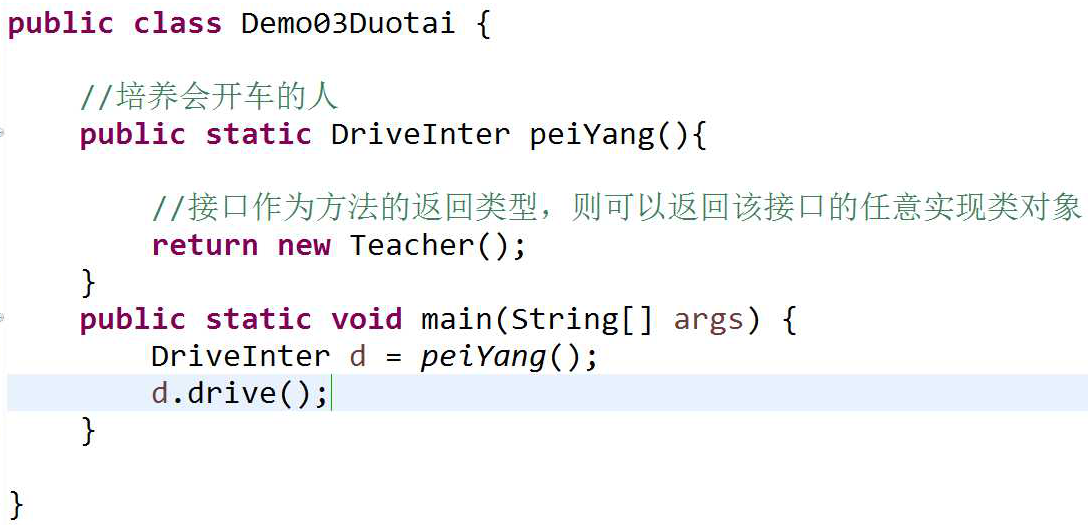


接口作为形参（只要是实现了该接口的对象都可以作为实参）。



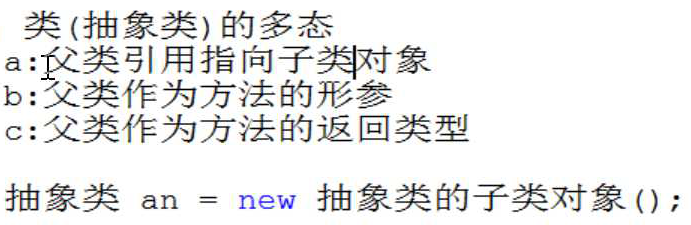
36 先点击类名+ctrl+1+回车：自动添加接口内的方法的快捷键。

37 接口还可以作为方法的返回类型。



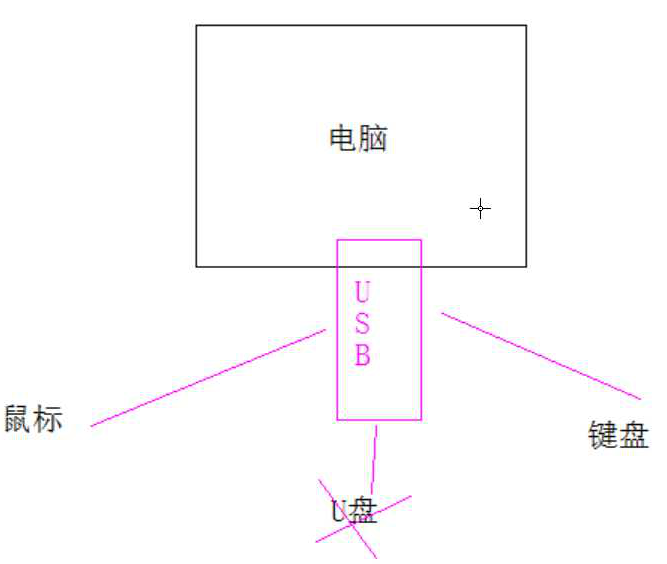
38 总结一句，出现接口的地方都可以用实现该接口的类对象。

39 抽象类和类一样，也可以用多态（因为抽象类也是类）：



40 当继承抽象类和实现接口都可以时，应该用接口，因为只能单继承，但接口可以实现不止一个。

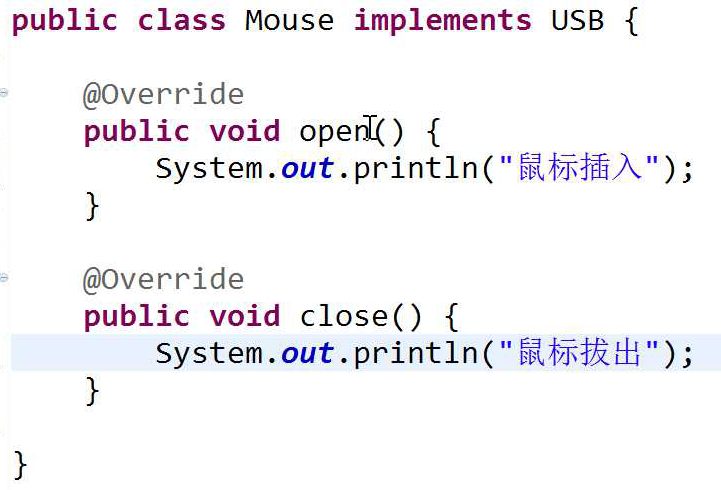
41 鼠标键盘相当于类，use接口相当于接口：



表示只要实现了接口的类都会具有某种特点，都有某种功能。

注意接口作为方法形参：





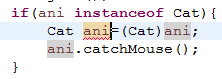
42 接口提供了规范，并且降低了类之间的耦合度。

43 **接口和抽象类都有多态。**

44 多态的向上转型和向下转型（强制转换）。进入方法之前向上转型，进入方法之后向下转型来进行反射操作。

45 接口中的方法默认使用public、abstract修饰，接口中的属性默认使用public、static、final修饰。

46 为什么会报错



47 **多态中没有用private修饰的成员变量访问时，访问的为父类类型的。**

48 方法的定义不能写在main里面。

49 main方法里引用的方法必须是static方法，因为main方法就是static静态方法。

50 xjad软件的作用是反编译，比如将class字节码文件反编译成.java文件。

51 字节码文件在bin（binary，二进制）文件夹中，.java文件在src文件夹中。

52 访问权限（无关类指的是非子类）（其中空的（default）最常用）：

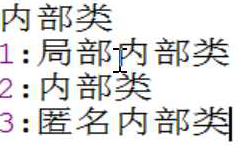


53



54 abstract和ststic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）：

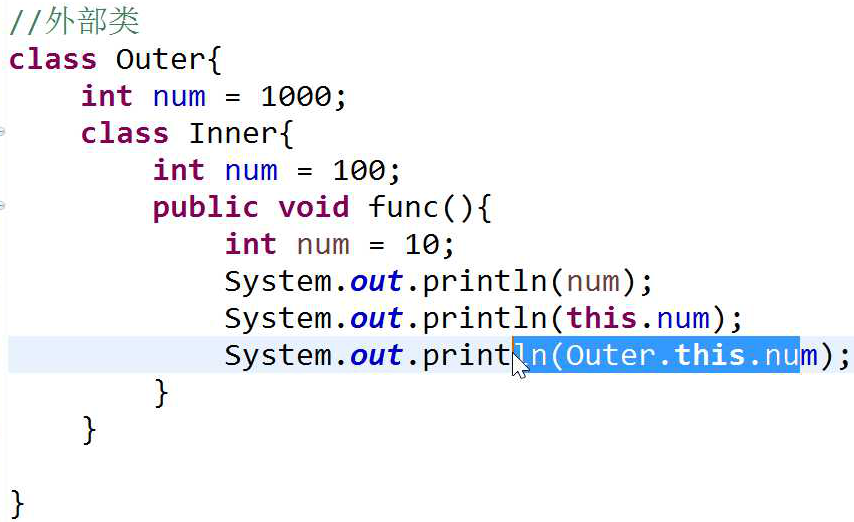


final的变量会被常量替换掉，因此不会访问内存。

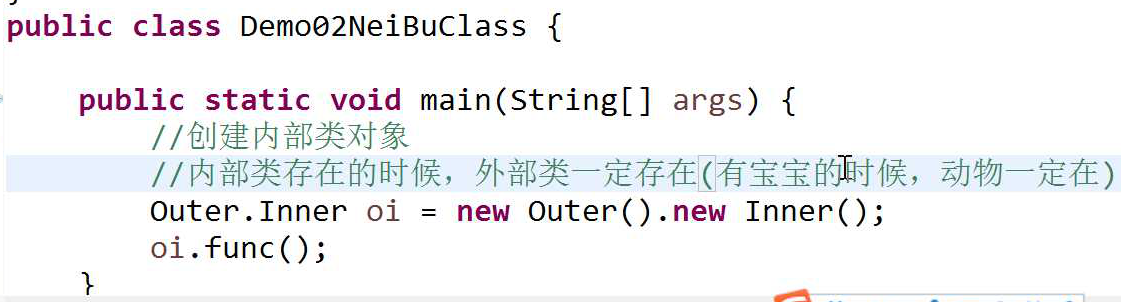
59 final放在数据类型的前面，例如final int

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：

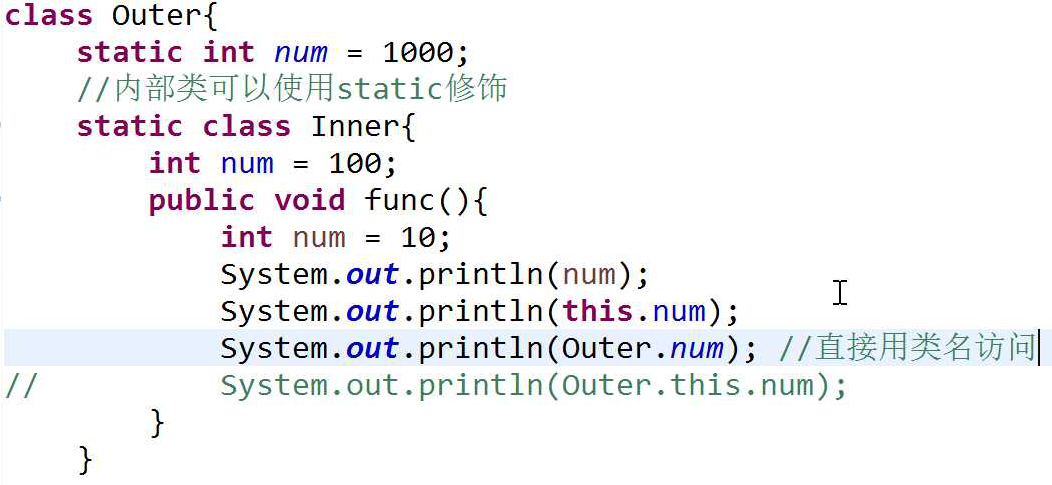


62 创建内部类对象：





63 内部类可以用ststic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，非静态不能用this访问，只能用类名访问：

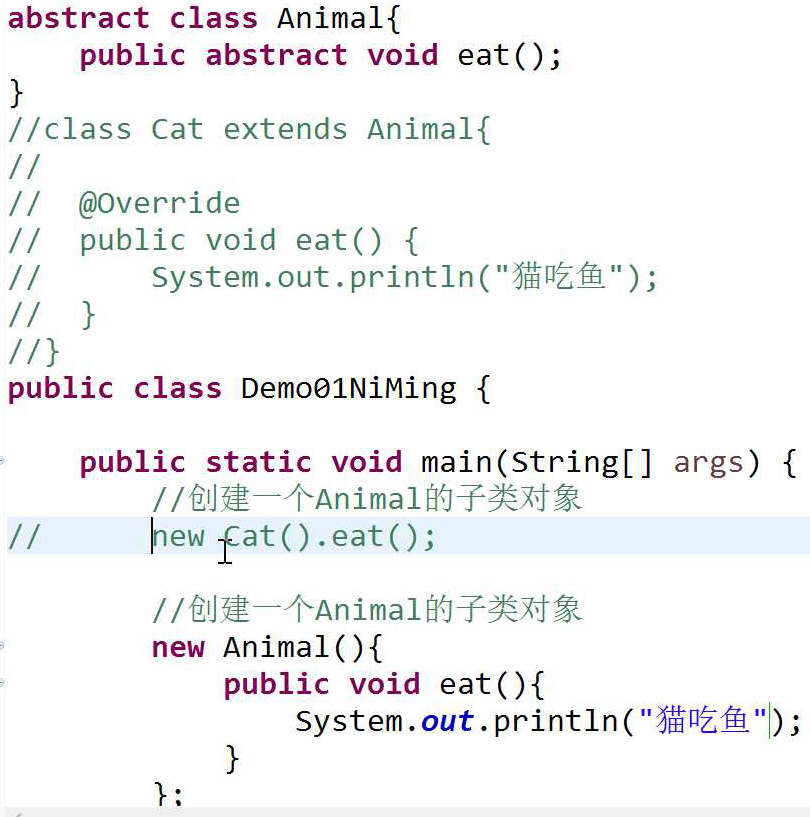


64 内部类可以私有，表示只能在外部类的内部使用和创建对象。

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）：



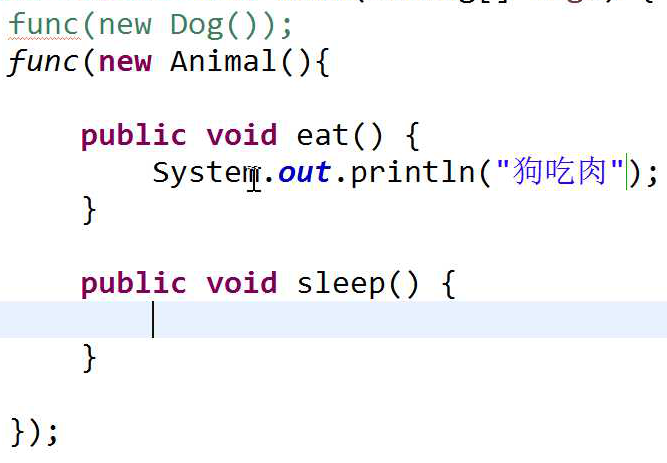
67 创建抽象类的匿名内部类时alt+反斜杠会自动补全抽象类里面的抽象方法。

68 若是不匿名就成了多态：



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。

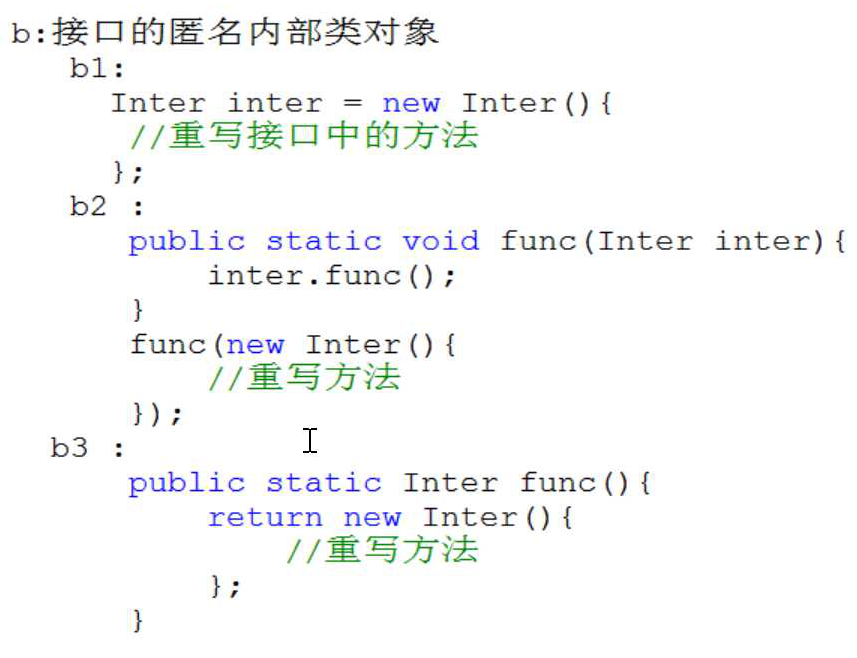
70 匿名内部类做参数（实参）（可以是普通类或父类也可以是抽象类或接口）：



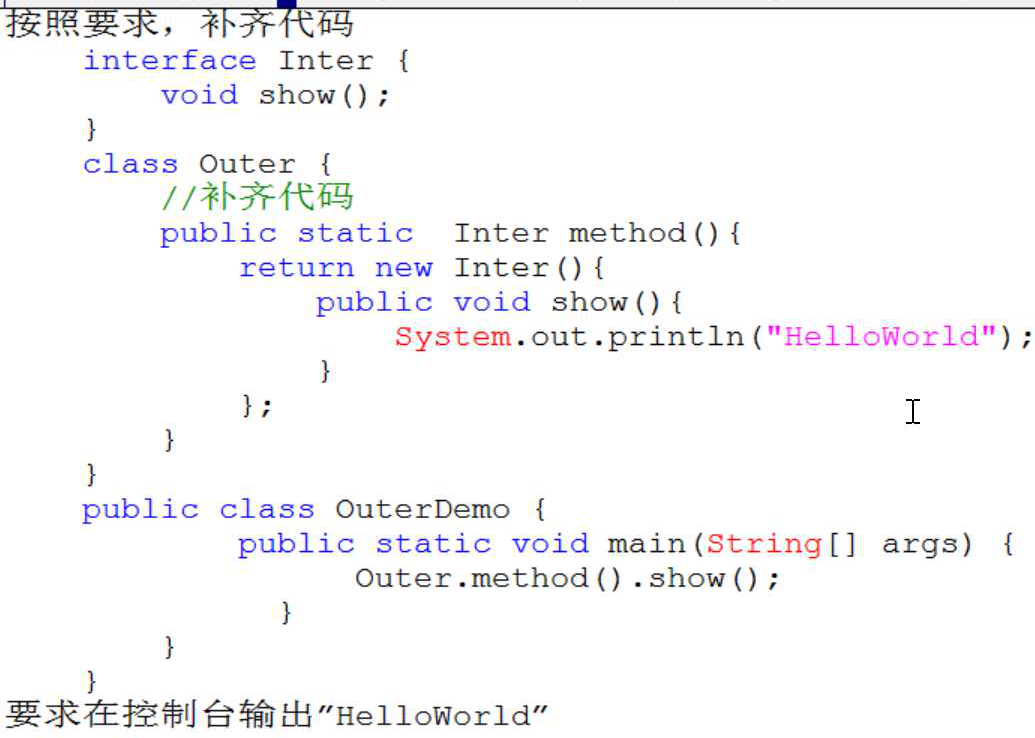
71 总之，任何出现类的地方都可以用匿名内部类代替。

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）：？？？？？？？

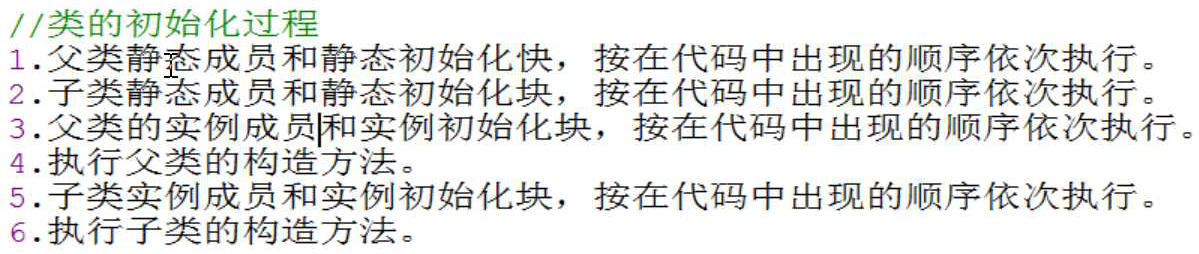


75 静态和匿名内部类用到的较多。

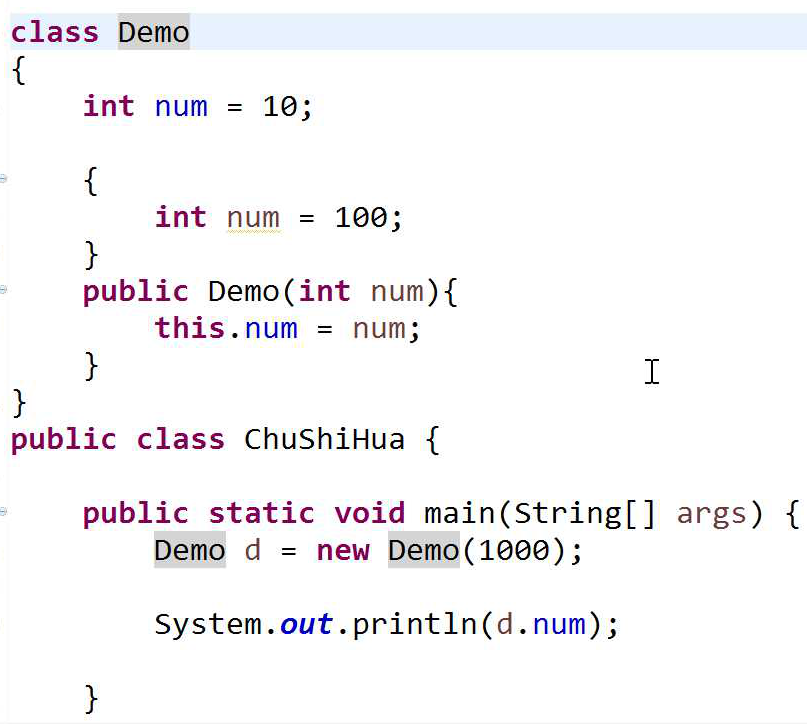
76 static静态用于数据共享。一个人改了其他人也受影响，可以加final防止修改。

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。**

**78 类的初始化过程：**



79 练习题：



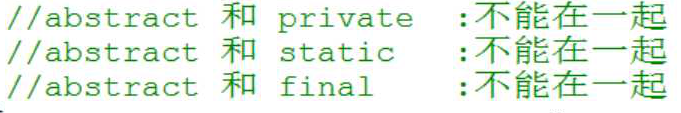
定义两个num没出错是因为代码块里的变成了局部变量，相当于一个成员变量，还有一个同名的局部变量。

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。

88