Day 10

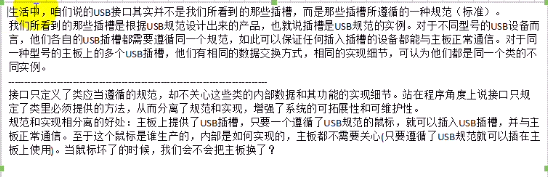
**01什么是接口和为什么需要接口**



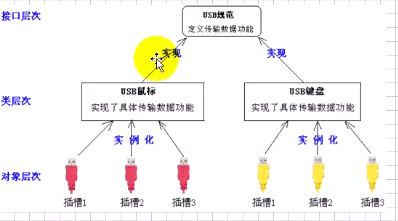








USB是一种协议（规范）：应该规范所有的usb设备具有的功能。



面向接口的编程：

接口和实现类体现了真正的多态。

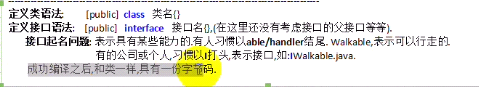
**02接口的定义和接口的成员**

Java中的接口表示规范，主要用来规范类的行为。

一般命名的时候使用able后缀或者使用I前缀。





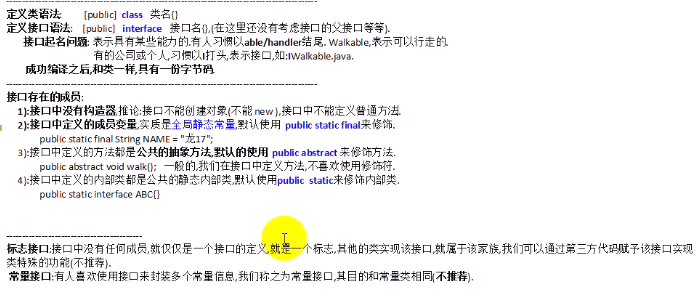


一个接口编译后就会生成一个字节码文件

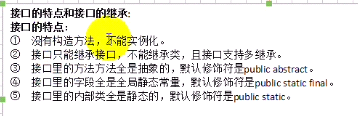
接口存在的成员：

* 接口中没有构造器，接口不能创建对象
* 接口中定义的成员变量，实质是全局静态常量，默认使用public static final修饰
* 接口中定义的方法都是公共的抽象方法，默认使用public abstract修饰
* 接口中定义的内部类都是public static修饰的





**03接口的特点和接口的继承**



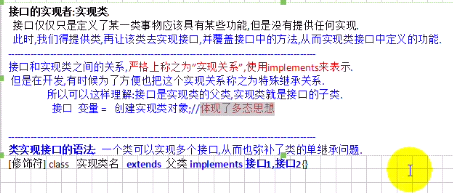
接口不能显示的定义构造器

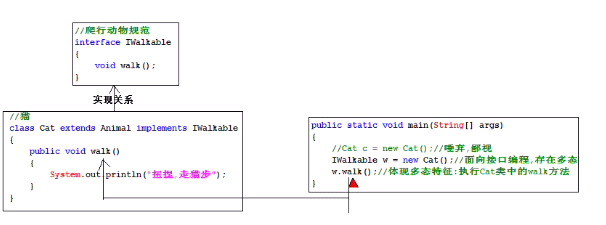
类是单继承，接口支持多继承

接口继承关键词为extends

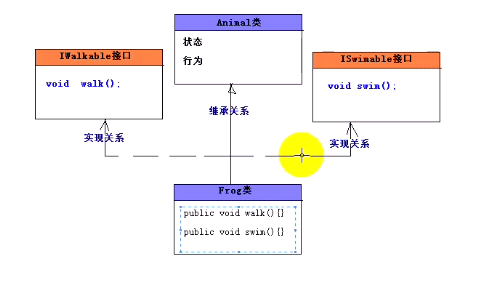
类实现接口关键词为implements

**04接口的实现关系**

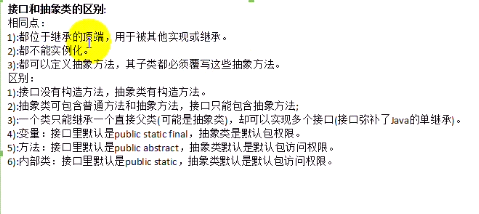




**05类和类以及类和接口的关系图**



**06接口和实现类的区别**



如果接口和抽象类可完成相同的功能，尽量使用接口，面向接口编程

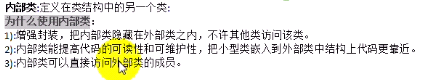
**07面向接口编程思想**

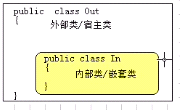


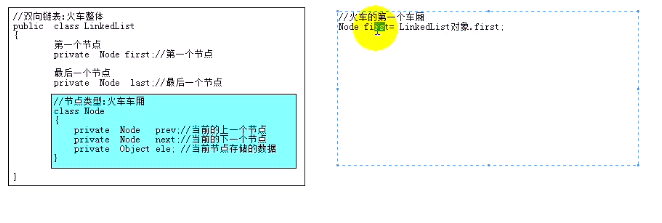


//======================20171017

**08内部类概述**





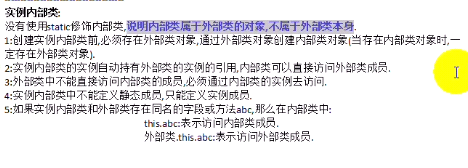


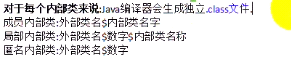


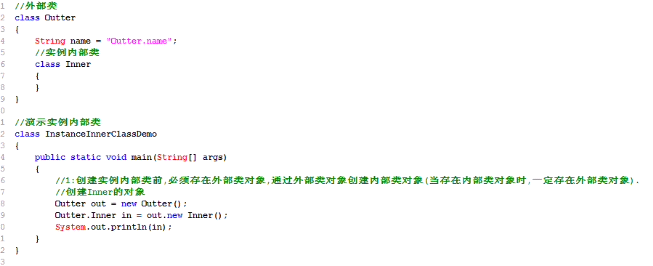
**09实例内部类和静态内部类分析**



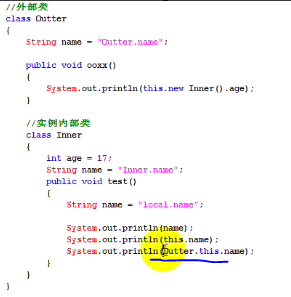
实例内部类：



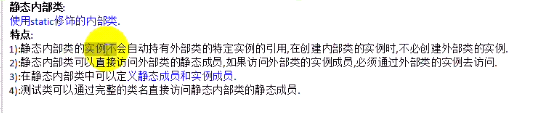




**内部类访问外部类成员使用outter.this.字段或方法名**



静态内部类：

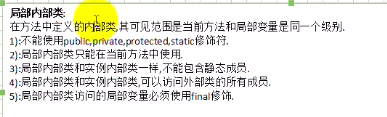


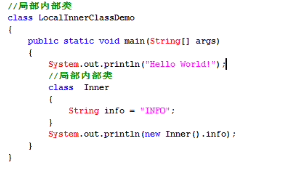
有静态内部类的时候可以没有外部类对象

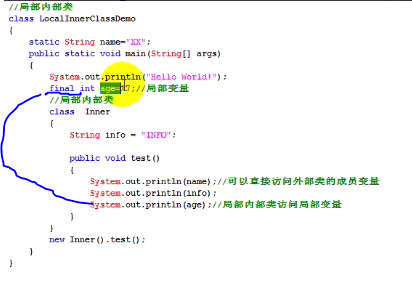


**10局部内部类**

可以把局部内部类当做局部变量(**所以修饰符可以是final但是不能是访问控制符，因为局部变量不能被访问控制符修饰**)



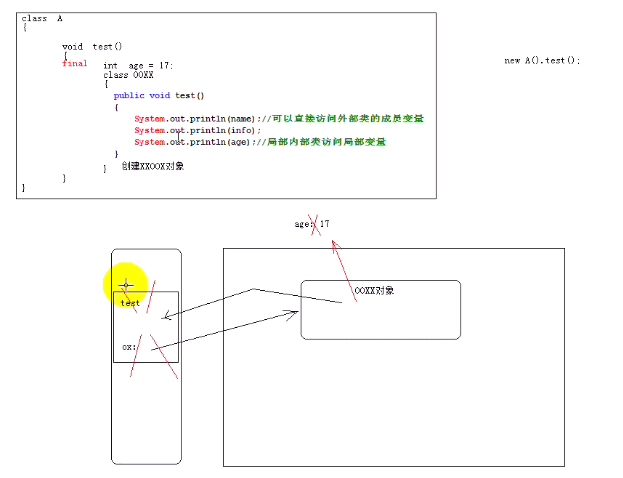






原因：如果当前方法不是main方法，那么当前方法调用完毕之后，当前方法的栈帧被销毁，方法内的局部变量的空间全部被销毁的时候，还存在堆内存，依然持有对局部的引用，但是方法被销毁的时候局部变量已经被销毁了。、

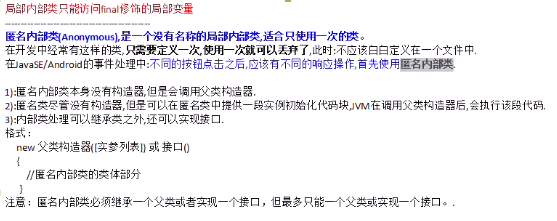
此时出现：在堆内存中，一个对象引用一个不存在的数据，为了避免该问题我们使用final修饰局部变量，从而编程常量永驻内存空间，即使方法销毁后也可以被持有。



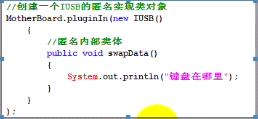
Age作为常量时就会被放在常量池里面，所以当方法调用完毕栈帧被销毁后仍然能够存在，而此时未被垃圾回收器回收的对象也可以引用此常量。

局部内部类几乎不会使用。

**11匿名内部类分析**



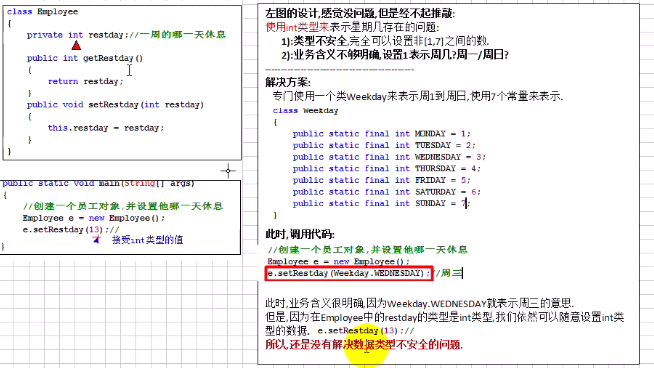
匿名内部类连类名都没有因此也就没有构造器。



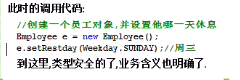
虽然是new接口，这只是语法格式，表名该匿名类实现了此接口。

匿名内部类访问局部变量也要加final修饰符，因为匿名内部类是特殊的局部内部类。

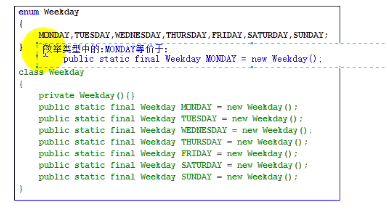
12引入枚举类型和枚举的模拟



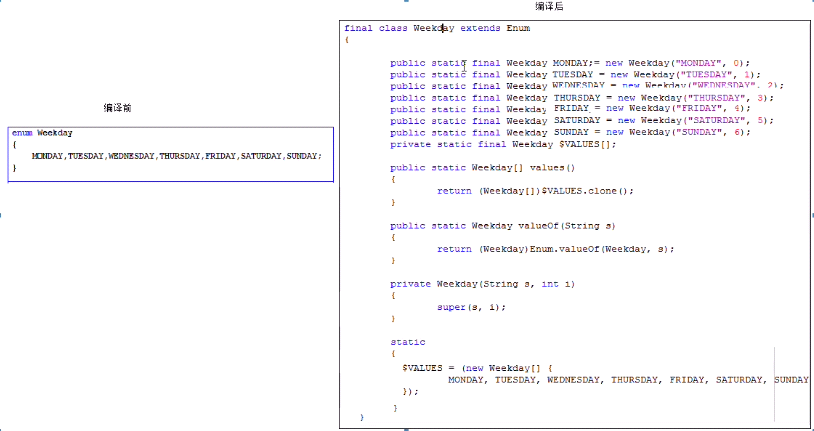


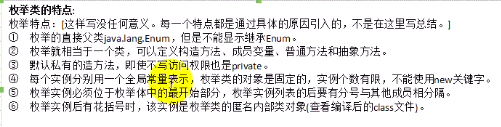


13枚举的定义和特点以及底层分析



反编译即可

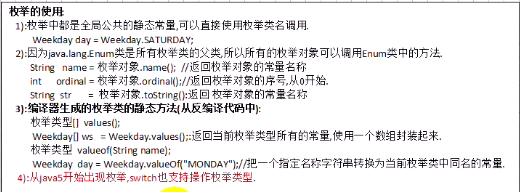


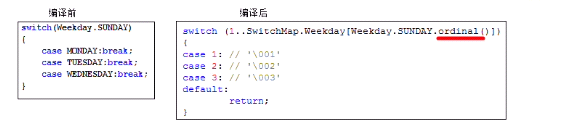


底层实现是

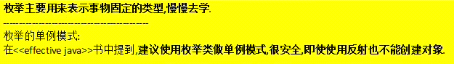
枚举底层没有无参数构造器

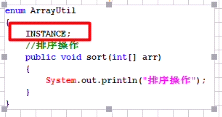
14枚举的操作细节





Switch只支持int类型，其他的都是自动转换了或者做了其他的改变





15小结

、