## 静态内部类

1. 如果一个类要被声明为static的，只有一种情况，就是静态内部类。

2. 静态内部类跟静态方法一样，只能访问外部类的静态的成员变量和方法，不能访问非静态的方法和属性，但是普通内部类可以访问任意外部类的成员变量和方法。

3. 静态内部类可以声明普通成员变量和方法，而普通内部类不能声明static成员变量和方法。

4. 静态内部类可以单独初始化，普通内部类不行，例子如下：

静态内部类

Inner i = new Outer.Inner();

普通内部类

Outer o = new Outer();

Inner i = o.new Inner();

根本原因：**静态内部类，不持有外部引用，普通内部类，则持有**。

5. 加载一个类时，其内部类不会同时被加载。一个类被加载，当且仅当其某个静态成员（静态域、构造器、静态方法等）被调用时发生。

可看如下例子：

public class Outer {

static {

System.out.println("load outer class...");

}

//静态内部类

static class StaticInner {

static {

System.out.println("load static inner class...");

}

static void staticInnerMethod() {

System.out.println("static inner method...");

}

}

public static void main(String[] args) {

Outer outer = new Outer(); //此刻其内部类是否也会被加载？

System.out.println("===========分割线===========");

Outer.StaticInner.staticInnerMethod(); //调用内部类的静态方法

}

}

运行结果：

load outer class…

==========分割线==========

load static inner class…

static inner method…

调用构造方法时，外部类Outer被加载，但这时其静态内部类StaticInner却未被加载。直到调用该内部类的静态方法（在分割线以下），StaticInner才被加载。可以做类似的实验验证非静态内部类的情况(普通内部类，只要不被new，那也不会加载)。

根据内部类不会在其外部类被加载的同时被加载的事实，我们可以引申出单例模式的一种实现方式：

public class Singleton {

private Singleton() {}

static class SingletonHolder {

private static final Singleton instance = new Singleton();

}

public static Singleton getInstance() {

return SingletonHolder.instance;

}

}

这样做的好处是，**可以确保多线程的情况下是安全的，且是唯一的**。

因为，当执行了SingletonHolder.getInstance()， SingletonHolder 类才被加载，instance 变量一次性初始化，后面直接返回的，就是instance对象。