参考:<http://www.cnblogs.com/Alex--Yang/p/3386863.html>

[Java静态类](http://www.cnblogs.com/Alex--Yang/p/3386863.html)

  在Java世界里，经常被提到静态这个概念，static作为静态成员变量和成员函数的修饰符，意味着它为该类的所有实例所共享（me：存放在jvm的常量方法区中，存在于jvm运行的整个生命周期），也就是说当某个类的实例修改了该静态成员变量，其修改值为该类的其它所有实例所见。最近一个项目里频繁用到static修饰的内部类，再读了一下《Effective Java》才明白为什么会用static来修饰一个内部类也就是本文的中心——静态类。

  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。如果在外部类声明为static，程序会编译都不会过。在一番调查后个人总结出了3点关于内部类和静态内部类（俗称：内嵌类）

1.静态内部类跟静态方法一样，只能访问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

 2.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。

3.静态内部类可以单独初始化:

Inner i = newOuter.Inner();

 普通内部类初始化：

Outer o = newOuter();

Inner i = o.newInner();

  静态内部类使用场景一般是**当外部类需要使用内部类，而内部类无需外部类资源，并且内部类可以单独创建的时候**会考虑采用静态内部类的设计，在知道如何初始化静态内部类，在《Effective Java》第二章所描述的静态内部类builder阐述了如何使用静态内部类：

[复制代码](javascript:void(0);)

publicclass Outer {

private String name;

privateint age;

publicstaticclass Builder {

private String name;

privateint age;

publicBuilder(int age) {

this.age = age;

}

public Builder withName(String name) {

this.name = name;

returnthis;

}

public Builder withAge(int age) {

this.age = age;

returnthis;

}

public Outer build() {

return new Outer(this);

}

}

privateOuter(Builder b) {

this.age = b.age;

this.name = b.name;

}

}

[复制代码](javascript:void(0);)

静态内部类调用外部类的构造函数，来构造外部类，由于静态内部类可以被单独初始化说有在外部就有以下实现：

public Outer getOuter()

{

Outer outer = newOuter.Builder(2).withName("Yang Liu").build();

return outer;

}

对于静态类**总结**是：1.**如果类的构造器或静态工厂中有多个参数，设计这样类时，最好使用Builder模式，特别是当大多数参数都是可选的时候。**

**2.如果现在不能确定参数的个数，最好一开始就使用构建器即Builder模式。**