Static subclass vs non-static subclass

区别

1. 访问权限的区别 1

Non-static subclass 里不能有 static variable，比如说对于 node class，是不可以把 data 和 next 设成 static 的，不然编译器会报错的。但是 static subclass 里能申明 static variables

2. 访问权限的区别 2

non-static subclass 可以访问其外面的 class 的 variable 和 methods, 即使外面的那个 variable 或者 method 被设置成 private!!! 这个我们很少见，你可以自己尝试一下哦。但是，static subclass 只能访问外部 class 里面的 static variable 和 method（注意这里是从内部的 class 访问外面，而不是从外面访问里面）

3. initialize 的区别

non-static subclass 和 static-subclass 如果要在你的包含 main 方法的 class 里面 initialize 的话，它们的 initialize 的方式完全不同哦，我用上次写的 linkedlist 来举个例子：

上次我们的 linkedlist 里面我们写的 static class Node，假设我们想要 initialize 的话（不是在 linkedlist 那个 class 里写 main method，而是在外面新建 class 写 main method，我们需要只要写：

```
Node n1 = new LinkedList.Node();
```

注意看这里我们的写法，Node 是 static class，所以左边边标签直接写出来，然而右边我们是通过 LinkedList.Node()来找到这个 class 的具体位置的，上面这个是 static

Generally，建立一个 static subclass 的对象的方法：

```
StaticInnerClass sic= new OuterClass.StaticInnerClass();
```

如果我们把 Node class 改成 non-static，那么我们想要在这个 main method 里新建一个 Node 需要如下步骤：

```
LinkedList list = new LinkedList();
LinkedList.Node n1 = list.new Node();
```

我们需要先建立 LinkedList 的对象，然后再通过 LinkedList 的对象去新建一个 Node

Generally，建立一个 non-static subclass 的对象的方法：

```
OuterClass oc = new OuterClass();
OuterClass.InnerClass ic = oc.new InnerClass();
```

总结：

1. static class 只能是 subclass，外部 class 不能作为 static class

2. non-static subclass 可以在这个 subclass 里面直接访问所有关于外部 class 的所有 variable 和 method，包括 private，可以看成外部的 class 和内部的 non-static subclass 的关系如同父亲和亲儿子，父亲可以给儿子共享所有的父亲拥有的内容，包括 private，所以如果在 main 里想要建立一个亲儿子的对象，得先问父亲同不同意，先建立父亲的对象，才能通过父亲的对象建立儿子的对象，而且儿子的 type 并非儿子，而是父亲.儿子（父亲.儿子 儿子 1 = 父亲.new 儿子(); 这样建立对象，左边是从属关系，右边体现的是儿子出不出生得经过父亲同意）

3. static subclass 只能访问外部 class 里面的 static 内容，外部的如果没写 static 或者设置成了 private，那么内部的 static class 就无法访问那些内容了。可以看成 static-subclass 和外部的 class 关系是公司里的上下级，内部的下级只能看上级的公开资料，但是关系并不亲密，如果要从 main method 新建一个 static-subclass 的对象时，就是（下级 下级 1 = new 上级.下级();这样建立对象，左边是下级本身的标签，右边是想要新建下级，得新分配一个上下级的关系）

测试代码

```
public class OuterClass {

	public class InnerClass{}
	public static class StaticInnerClass{}

	public static void main(String[] args) {

		OuterClass oc = new OuterClass();
		InnerClass ic=oc.new InnerClass();
		StaticInnerClass sic=new OuterClass.StaticInnerClas
s();
	}

}
```

补充：关于为什么要有 static-subclass：

如果内部类与外部类关系不紧密，不需要访问外部类的所有属性或方法，那么设计成静态内部类。而且，由于静态内部类与外部类并不会保存相互之间的引用，因此在一定程度上，还会节省那么一点内存资源。因此，静态内部类，和外部类没有什么“强依赖”上的关系，也就是说，没有外部类提供一些属性，内部静态类也可以创建并初始化。

既然上面已经说了什么时候应该用静态内部类，那么如果需求不符合静态内部类所提供的一切好处，那就应该考虑使用内部类了。最大的特点就是：你在内部类中需要访问有关外部类的所有属性及方法。