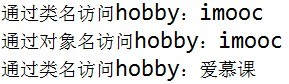
**static**

**Java** 中的 **static** 使用之静态变量

Java 中被 static 修饰的成员称为静态成员或类成员。它属于整个类所有，而不是某个对象所有，即被类的所有对象所共享。静态成员可以使用类名直接访问，也可以使用对象名进行访问。当然，鉴于他作用的特殊性更推荐用类名访问~~

使用 static 可以修饰变量、方法和代码块。

例如，我们在类中定义了一个 静态变量 hobby ，操作代码如下所示：

运行结果：

注意：静态成员属于整个类，当系统第一次使用该类时，就会为其分配内存空间直到该类被卸载才会进行资源回收！

~~

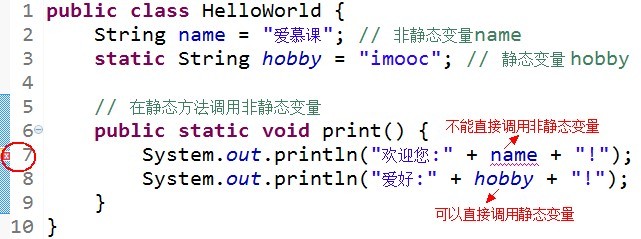
**Java** 中的 **static** 使用之静态方法

与静态变量一样，我们也可以使用 static 修饰方法，称为静态方法或类方法。其实之前我们一直写的 main 方法就是静态方法。静态方法的使用如：



运行结果：

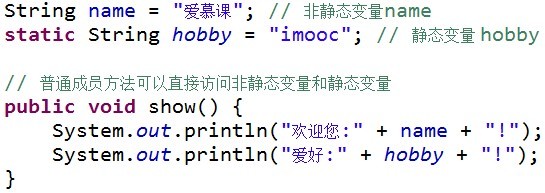
需要注意：

1、 静态方法中可以直接调用同类中的静态成员，但不能直接调用非静态成员。如：

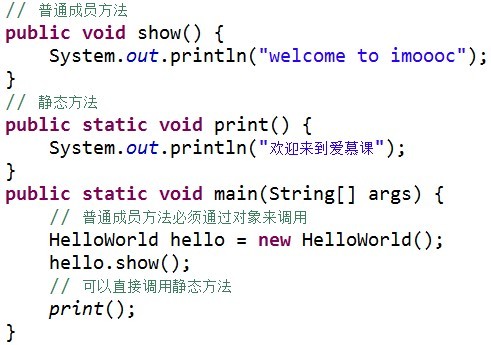
如果希望在静态方法中调用非静态变量，可以通过创建类的对象，然后通过对象来访问非静态变量。如：



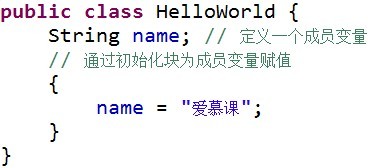
2、 在普通成员方法中，则可以直接访问同类的非静态变量和静态变量，如下所示：



3、 静态方法中不能直接调用非静态方法，需要通过对象来访问非静态方法。如：



**Java** 中的 **static** 使用之静态初始化块

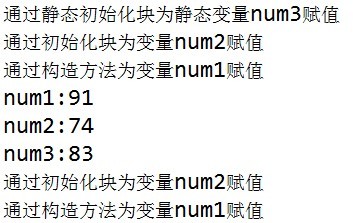
Java 中可以通过初始化块进行数据赋值。如：

在类的声明中，可以包含多个初始化块，当创建类的实例时，就会依次执行这些代码块。如果使用 static 修饰初始化块，就称为静态初始化块。

需要特别注意：静态初始化块只在类加载时执行，且只会执行一次，同时静态初始化块只能给静态变量赋值，不能初 始化普通的成员变量。

我们来看一段代码：

运行结果：



通过输出结果，我们可以看到，程序运行时静态初始化块最先被执行，然后执行普通初始化块，最后才执行构造方 法。由于静态初始化块只在类加载时执行一次，所以当再次创建对象 hello2 时并未执行静态初始化块。

