# 再次深入final关键字- 匿名类用到的变量为什么一定要是final的呢?


提起final变量,大家都是耳熟能详,

1. final成员变量表示常量，只能被赋值一次，赋值后值不再改变。
2. final类不能被继承，没有子类，final类中的方法默认是final的。
3. final方法不能被子类的方法覆盖，但可以被继承。
4. final不能用于修饰构造方法
5. final 常与 static一起用,作为常量来使用.

## 提问: 那么在一些方法的参数中定义为final是干嘛的?

答: 不希望这个变量在方法里面被修改,防止无意的修改而影响到调用方法外的变量

不知道这个答案是从哪个语言带过来的, 感觉把final当成const了.

假设出现以下方法定义:

```
void fuc(final String string);
void fuc1(String string);
```

对于这种写法,String本身就是一个不可以变的对象,并且其是作为基本参数

类型,传参进入后,方法里面改变了String的值外面也不会改变,因此这两种写法实际结果是一样.只不过加了final,在string被赋值时IDE会直接报红.

```
void fuc(final A a);
void fuc(A a);
```

对于这种写法, A是一个对象,这样写不是代表A里面内容不能被改变,而是a不能被赋值为新对象了.

## 为什么使用匿名内部类的时候参数一定要加上final

让我们写一个类看一下

```
public class Test {

    class Person {
        String name;
        int age;
    }

    private void func(final Person a) {
        new Thread() {
            @Override
            public void run() {
                super.run();
                a.name = "hello";
            }
        }.start();
    }
}
```

javac一下该文件,生成了3个class文件,我们看其中的两个

内部类Person的class

```
class Test$Person {
    String name;
    int age;

    Test$Person(Test var1) {
```

```
        this.this$0 = var1;
    }
}
```

匿名内部类Thread的class

```
class Test$1 extends Thread {
    Test$1(Test var1, Person var2) {
        this.this$0 = var1;
        this.val$a = var2;
    }

    public void run() {
        super.run();
        this.val$a.name = "hello";
    }
}
```

从这个匿名内部类的class文件我们可以看出两点

1. 其构造参数中增加了一个调用类.即我们所说的持有外部类的引用.
2. 我们定义的final参数被当做构造方法传了进来.至于为什么要把这个参数当做构造函数参数传进来,因为调用它的方法参数是存在栈里面的,其生命周期随着这个方法的调用结束而结束.而我们的异步任务可不是,有可能会执行很长时间.

那么final的关键字作用就凸显了,Person参数要拷贝到内部类中，而拷贝会带来不一致性, `func`中是一个异步的操作,负责改变a的name的值. 假设Person a 不是final的.那么a可以被任意指向新的对象,那么传给这个异步任务的对象还是老对象,这就造成了不一致.因此Java需要强制约束对象的一致性.因此必须是final的.

本文作者：Anderson/Jerey_Jobs

博客地址： http://jerey.cn/
简书地址：Anderson大码渣
github地址：https://github.com/Jerey-Jobs