链滴

永远不要使用双花括号初始化实例!!!

作者：washmore
原文链接：https://ld246.com/article/1498563483898
来源网站：链滴

说在前面

一段时间以来,经常发现有人使用类似如下代码装逼耍酷

```
Map source = new HashMap(){{
    put("firstName", "John");
    put("lastName", "Smith");
    put("organizations", new HashMap(){{
        put("it", new HashMap(){{
            put("id", "1234");
        }});
        put("oa", new HashMap(){{
            put("id", "5678");
        }});
    }});
}};
```

我们称之为双花括号初始化(构造)实例,看起来是挺酷的!

如果你对这个语法只是拿来用过,知其然不知其所以然,没关系,实际上它挺简单的,一共做了两件事

- 使用下面的代码实例化一个继承自Map的匿名类

```
new HashMap(){}
```

- 然后使用下面的代码 继续实例化一个匿名类

```
{
    put("id", "1234");
}
```

原理详见Oracle官方文档:Initializing Fields

实际上,这就是一个执行构造函数的过程~

为什么需要拒绝以这种方式初始化实例?

可读性 --这是最不重要的一个原因

双括号初始化实例的代码片段对于没有接触过的人来看,很难直观的理解他的作用,起码,从语法上来讲,觉得别扭(虽然写的人挺爽);

每个实例一个类型

看起来,这段代码只是一个简单的实例化对象的过程,除此之外,什么事情都没有发生 ,实际上,他同时也创建一个不可复用的(匿名内部)类!

如果你只是偶尔这么做一次,那还勉强可以接受,如果你在一个体量庞大的企业级应用中大范围的使用种做法,那对ClassLoader来说简直就是灾难,他需要在堆内存中为这些生成的所有的匿名内部类保持一引用!

如果你不信的话,你尝试建一个Test类,然后copy上面的代码,然后编译,你就会发现生成了如下文件:

```
Test$1$1$1.class
Test$1$1$2.class
Test$1$1.class
Test$1.class
Test.class
```

holy shit!并不是你想象中理所当然的就一个Test类的class...

最重要的原因是这样做会导致内存泄漏!!!

这是所有匿名内部类都有的一个致命的问题--**他们会在内部维护一个外部容器类的引用**,想像一下,你在个庞大的应用中创建了一个如下代码类:

```
public class ReallyHeavyObject {

    // Just to illustrate...
    private int[] tonsOfValues;
    private Resource[] tonsOfResources;

    // This method almost does nothing
    public void quickHarmlessMethod() {
        Map source = new HashMap(){{
            put("firstName", "John");
            put("lastName", "Smith");
            put("organizations", new HashMap(){{
                put("0", new HashMap(){{
                    put("id", "1234");
                }});
                put("abc", new HashMap(){{
                    put("id", "5678");
                }});
            }});
        }};

        // Some more code here
    }
}
```

成吨的资源需要在ReallyHeavyObject被GC的时候清理,然而你可能并没有意识到这一点,你只是:若无事的调用了quickHarmlessMethod(),完成了ReallyHeavyObject的使命,而且以后可能不会再调用第次```

如果有一天,另一个开发者改动了你的代码,在quickHarmlessMethod方法中返回了source对象,如下码所示:

```
    public Map quickHarmlessMethod() {
        Map source = new HashMap(){{
            put("firstName", "John");
            put("lastName", "Smith");
            put("organizations", new HashMap(){{
                put("0", new HashMap(){{
                    put("id", "1234");
                }});
                put("abc", new HashMap(){{
```

```
                put("id", "5678");
            }});
        }});
    }};

    return source;
}
```

那你又犯了另一个罪行!如前文所说,所有匿名内部类都会在内部维护一个外部容器类的引用,所以,你在心之下成功的将ReallyHeavyObject这个类通过此方法暴露了出去;

如果不信的话,请尝试以下代码:

```
public static void main(String[] args) throws Exception {
    Map map = new ReallyHeavyObject().quickHarmlessMethod();
    Field field = map.getClass().getDeclaredField("this$0");
    field.setAccessible(true);
    System.out.println(field.get(map).getClass());
}
```

会返回内容:

```
class ReallyHeavyObject
```

如果你还不信的话,就请打开调试模式自行查看!你会发现不仅这个匿名Map持有了外部类的引用,就连的子类,也持有!

延伸

也许你会讲,第三个问题可以通过将方法静态化来解决,这没错!但这是基于你对以上几点有深刻的认知前提下,万一哪天你的小弟在不知情的前提下,将static关键字删掉,问题又会出现了!

古往今来,使用匿名类已经带来够多的麻烦了,而匿名内部类,那简直更危险!因为无心的人根本不会注意到他们通过一个匿名内部类将外部容器类的引用暴露了出去!

结论

SO,我们的结论是 没有蛀牙————————— 永远不要使用双花括号初始化实例!!!

参考文章:Don’ t be “Clever”: The Double Curly Braces Anti Pattern