再识 Java 的 final 关键字

作者：88250
原文链接：https://ld246.com/article/1473562584429
来源网站：链滴

这是 2011 年时候的一篇博文，大家如果好奇 Solo / Sym 中为何那么多 final 修饰，请看此文。

在 Java 中，我们一般用 final 关键字来定义类字段常量，防止类继承、方法覆写。但其实我们也应该可能地使用 final 关键字来修饰方法参数与局部变量。

因为这样做可以使代码更易读，能让阅读者清楚地知道该参数 /变量是不会在被重赋值的，也可以让译器更好地帮助我们优化生成的字节码。

代码可读性

方法的输入参数是不应该被重新赋值的，细节见重构 Remove Assignments to Parameters。

对于局部变量也应该尽可能地使用 final 来修饰，例如折半查找返回中间索引的方法（摘自 Sun JDK：

```
private static <T> int indexedBinarySearch(List<? extends Comparable<? super T>> list, T key

{
    int low = 0;
    int high = list.size()-1;

    while (low <= high) {
        int mid = (low + high) >>> 1;
        Comparable<? super T> midVal = list.get(mid);
        int cmp = midVal.compareTo(key);

        if (cmp < 0)
            low = mid + 1;
        else if (cmp > 0)
            high = mid - 1;
        else
            return mid; // key found
    }
    return -(low + 1);  // key not found
}
```

其中 mid 标记了中间值索引，midValue 是中间值，cmp 是中间值与待查找值 key 的比较结果，这个变量在 while 迭代中都是不会被重新赋值的。

另外，入参也没有在方法内部被重赋值。所以改成下面这样是更好的选择：

```
private static <T> int indexedBinarySearch(final List<? extends Comparable<? super T>> list, f
nal T key)
{
    int low = 0;
    int high = list.size()-1;

    while (low <= high) {
        final int mid = (low + high) >>> 1;
        final Comparable<? super T> midVal = list.get(mid);
        final int cmp = midVal.compareTo(key);
```

```
        if (cmp < 0)
            low = mid + 1;
        else if (cmp > 0)
            high = mid - 1;
        else
            return mid; // key found
    }
    return -(low + 1);  // key not found
}
```

这样能够给予代码阅读者更多的上下文信息，能更精确地表达实现语义，也有利于避免实现时由于疏造成的错误。

当然，有人也许会说太多 final 会造成视觉干扰，影响阅读。其实，这只是个视觉习惯而已 ;-)

编译内联优化

通常，编译器会内联常量表达式，比如方法：

```
private void test() {
    long i = 88250;
    long j = 219;

    long k = i + j;

    System.out.println(k);
}
```

编译生成的字节码：

```
 private void test();
   Code:
     0:   ldc2_w  #2; //long 88250l
     3:   lstore_1
     4:   ldc2_w  #4; //long 219l
     7:   lstore_3
     8:   lload_1
     9:   lload_3
     10:  ladd
     11:  lstore  5
     13:  getstatic       #6; //Field java/lang/System.out:Ljava/io/PrintStream;
     16:  lload   5
     18:  invokevirtual   #7; //Method java/io/PrintStream.println:(J)V
     21:  return
```

很显然，long i = 88250，long j = 219 以及 long k = i + j 这三句语句可以写为常量表达式。改写的 test 方法：

```
private void test() {
    final long i = 88250;
    final long j = 219;

    final long k = i + j;
```

```
    System.out.println(k);
}
```

编译生成的字节码：

```
private void test();
  Code:
   0:   getstatic       #2; //Field java/lang/System.out:Ljava/io/PrintStream;
   3:   ldc2_w  #3; //long 88469l
   6:   invokevirtual   #5; //Method java/io/PrintStream.println:(J)V
   9:   return
```

从上面我们可以看出，编译器生成字节码时做了内联处理，即

- 在引用某常量的地方进行值替换
- 移除原常量

当然，除了上面提到的使用 final 修饰变量外，final 也可用于类、方法和字段。

修饰方法时要注意内联条件，不是加了 final 编译器就一定会对该方法进行内联，因为至少要考虑二制兼容性。

结论

final 关键字能更有效地表达设计与实现意图，能够有助于减少编码错误，所以我们应该尽可能地使用 final。

参考

- The Java Language Specification, Third Edition
- DIP, Dependency Inversion Principle
- Robert Simmons Jr. 2004, The Final Story, Hardcore Java
- 重构——改善既有代码的设计
- 敏捷软件开发——原则、模式与实践