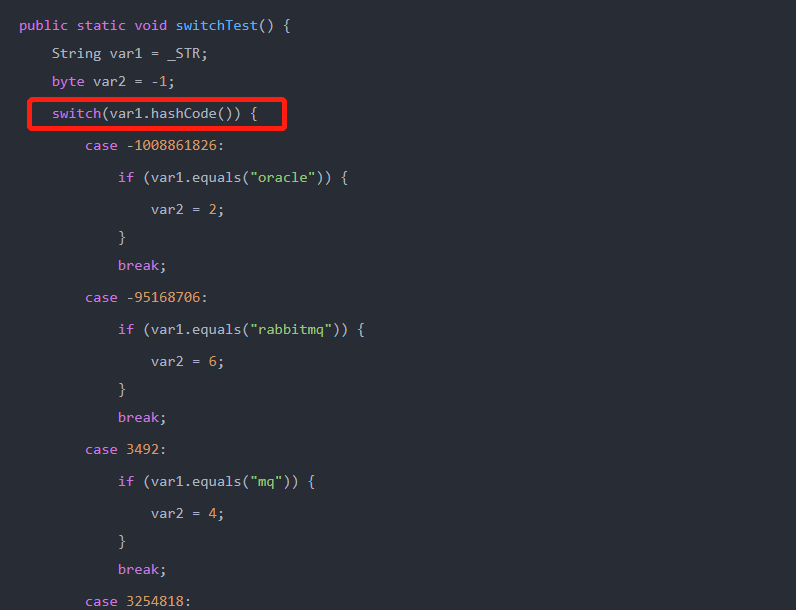
httpsmp.weixin.qq.comsicjByl5g3GvkgqLGIYakgQ

# SWITCH语句性能优化

## **switch 性能优化**

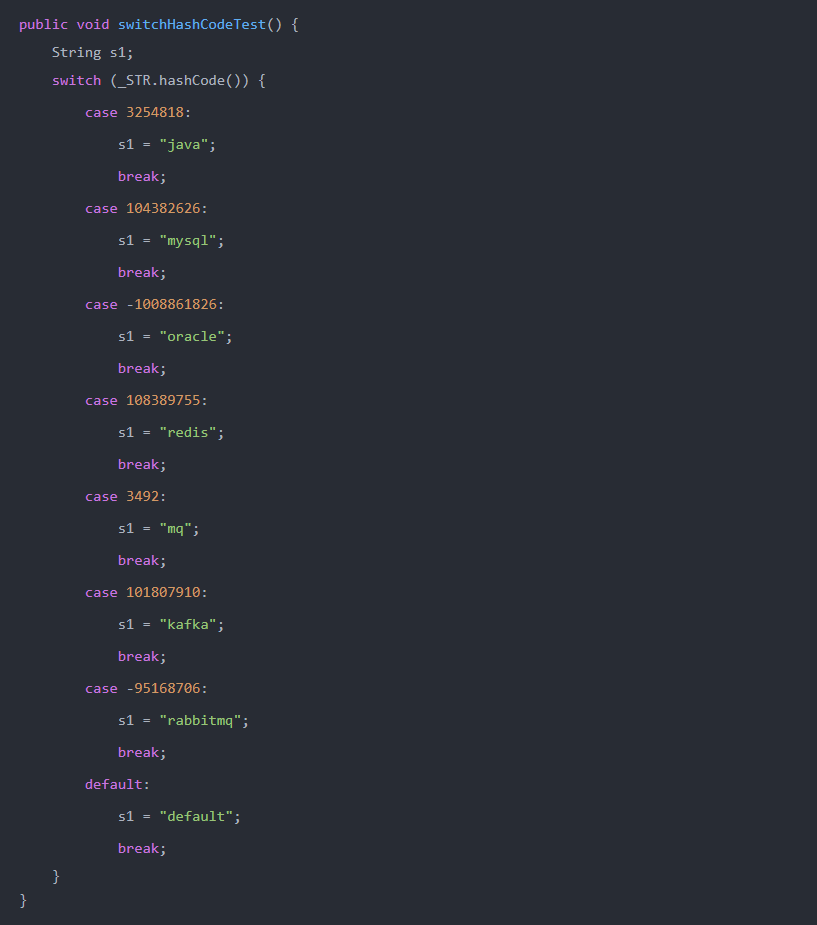
我们知道在 JDK 1.7 之前 switch 是不支持 String 的，**实际上 switch 只支持 int 类型**。

在 JDK 1.7 中的 String 类型，其实在编译的时候会使用 hashCode 来作为 switch 的实际值，以上 switch 判断字符串的代码，编译为字节码实际结果如下：



知道了 switch 实现的本质，那么优化就变得比较简单了。

从以上的字节码可以看出，如果要优化 switch 只需要把 String 类型变成 int 类型就可以了，这样就剩了每个 case 中进行 if 判断的性能消耗，最终的优化代码如下：



## **注意事项**

以上的 switch 优化是基于 String 类型的，同时我们需要注意 hashCode 重复的问题，例如对于字符串“Aa”和“BB”来说，他们的 hashCode 都是 2112，因此在优化是需要注意此类问题，也就是说我们使用 hashCode 时，必须保证判断添加的值是已知的，并且最好不要出现 hashCode 重复的问题，如果出现此类问题，我们的解决方案是在 case 中进行判断并赋值。

## **总结**

通过本文我们知道 **switch 本质上只支持 int 类型的条件判断**，即使是 JDK 1.7 中的 String 类型，最终编译的时候还是会被转化为 hashCode（int）进行判断。但因为编译成字节码后会在 case 中使用 if equals 进行比较，所以性能并不算太高（只比 if 高一点点），因此我们可以直接把 String 转化成 int 类型进行比较，从而避免在 case 中进行 if equals 判断的性能消耗，这样就大大的提升 switch 的性能，但需要注意的是，有些 key 值的 hashCode 是相同的，因此在优化时需要提前规避。