[**java枚举enum equal与==**](https://www.cnblogs.com/fswhq/p/9695961.html)

**目录**

* [问题](https://www.cnblogs.com/fswhq/p/9695961.html#_label0)
* [答案](https://www.cnblogs.com/fswhq/p/9695961.html#_label1)
* [额外答案](https://www.cnblogs.com/fswhq/p/9695961.html#_label2)
  + [能在枚举中使用 == 进行判断？](https://www.cnblogs.com/fswhq/p/9695961.html#_label2_0)
  + [什么时候 == 和 equals 不一样？](https://www.cnblogs.com/fswhq/p/9695961.html#_label2_1)
  + [== 不会抛出 NullPointerException](https://www.cnblogs.com/fswhq/p/9695961.html#_label2_2)
  + [== 在编译期检测类型兼容性](https://www.cnblogs.com/fswhq/p/9695961.html#_label2_3)
  + [什么时候使用 == ？](https://www.cnblogs.com/fswhq/p/9695961.html#_label2_4)

原文链接:https://www.cnblogs.com/xiohao/p/7405423.html

[Return Top](https://www.cnblogs.com/fswhq/p/9695961.html#_labelTop)

**问题**

我知道[Java](http://lib.csdn.net/base/java)枚举会被编译成一个包含私有构造参数和一堆静态方法的类，当去比较两个枚举的时候，总是使用equals()方法，例如：

[复制代码](javascript:void(0);)

public useEnums(SomeEnum a)

{

if(a.equals(SomeEnum.SOME\_ENUM\_VALUE))

{

...

}

...

}

[复制代码](javascript:void(0);)

除此之外，我也可以使用 == 替代equals() 方法

[复制代码](javascript:void(0);)

public useEnums2(SomeEnum a)

{

if(a == SomeEnum.SOME\_ENUM\_VALUE)

{

...

}

...

}

[复制代码](javascript:void(0);)

我有5年以上的java编程经验，并且我想我也懂得 == 和 equals() 之间的区别，但是我仍然觉得很困惑，哪一个操作符才是我该使用的。

[Return Top](https://www.cnblogs.com/fswhq/p/9695961.html#_labelTop)

**答案**

二者皆对，如果你看过枚举的源码，你会发现在源码中，equals也仅仅非常简单的 == 。 我使用 == ，因为无论如何，这个左值是可以为 null的

译者补充 java.lang.Enum 中Equals 代码：

public final boolean equals(Object other) {

return this==other;

}

[Return Top](https://www.cnblogs.com/fswhq/p/9695961.html#_labelTop)

**额外答案**

**能在枚举中使用 == 进行判断？**

答案是肯定的，因为枚举有着严格的实例化控制，所以你可以用 == 去做比较符，这个用法，在官方文档中也有明确的说明。

JLS 8.9 Enums 一个枚举类型除了定义的那些枚举常量外没有其他实例了。 试图明确地说明一种枚举类型是会导致编译期异常。在枚举中final clone方法确保枚举常量从不会被克隆，而且序列化机制会确保从不会因为反序列化而创造复制的实例。枚举类型的反射实例化也是被禁止的。总之，以上内容确保了除了定义的枚举常量之外，没有枚举类型实例。

因为每个枚举常量只有一个实例，所以如果在比较两个参考值，至少有一个涉及到枚举常量时，允许使用“==”代替equals()。（equals()方法在枚举类中是一个final方法，在参数和返回结果时，很少调用父类的equals()方法，因此是一种恒等的比较。）

**什么时候 == 和 equals 不一样？**

As a reminder, it needs to be said that generally, == is NOT a viable alternative to equals. When it is, however (such as with enum), there are two important differences to consider: 通常来说 == 不是一个 equals的一个备选方案，无论如何有2个重要的不同处需要考虑：

**== 不会抛出 NullPointerException**

[复制代码](javascript:void(0);)

enum Color { BLACK, WHITE };

Color nothing = null;

if (nothing == Color.BLACK); // runs fine

if (nothing.equals(Color.BLACK)); // throws NullPointerException

[复制代码](javascript:void(0);)

**== 在编译期检测类型兼容性**

[复制代码](javascript:void(0);)

enum Color { BLACK, WHITE };

enum Chiral { LEFT, RIGHT };

if (Color.BLACK.equals(Chiral.LEFT)); // compiles fine

if (Color.BLACK == Chiral.LEFT); // DOESN'T COMPILE!!! Incompatible types!

[复制代码](javascript:void(0);)

**什么时候使用 == ？**

Bloch specifically mentions that immutable classes that have proper control over their instances can guarantee to their clients that == is usable. enum is specifically mentioned to exemplify. 具体来说，那些提供恰当实例控制的不可变类能够保证 == 是可用的，枚举刚好符合这个条件。

考虑静态工厂方法代替构造器 它使得不可变的类可以确保不会存在两个相等的实例，即当且仅当a==b的时候才有a.equals(b)为true。如果类保证了这一点，它的客户端可以使用“==”操作符来代替equals（Object）方法，这样可以提升性能。枚举类型保证了这一点

总而言之，在枚举比较上使用 == ， 因为： 1. 能正常工作 2. 更快 3. 运行时是安全的 4. 编译期也是安全的

stackoverlfow链接：<http://stackoverflow.com/questions/1750435/comparing-java-enum-members-or-equals>