Boolean源码

位置：java.lang

Boolean类是对原生类型boolean的包装类，这个类提供了很多可以将boolean转换为String、String转换为boolean的方法，实现了Serializable类和Comparable类，Boolean对象可以进行比较和序列化。

```
public final class Boolean implements java.io.Serializable,Comparable<Boolean>
```

Boolean类被final修饰符修饰，说明Boolean类不能被继承。同样，Boolean的属性都是被final修饰的，这些属性也是不能被改变的，这些属性如下：

```
//对原生类型值true的包装
public static final Boolean TRUE = new Boolean(true);
//对原生类型值false的包装
public static final Boolean FALSE = new Boolean(false);
//Boolean的class类型
 public static final Class<Boolean> TYPE = (Class<Boolean>)
Class.getPrimitiveClass("boolean");
//Boolean的值
 private final boolean value;
```

TRUE和FALSE属性是用原生类型包装好的，表示原生类型true和false的包装类，这两个属性除了用final修饰，还用了static修饰，属于静态的类属性，提供这两个类属性的好处是，由于原生类型只有true和false，只需要包装一次就可以处处使用，减少创建新对象带来的性能消耗。在开发的过程中，如果需要使用到Boolean类，直接使用类静态属性TRUE和FALSE，在Boolean类的方法中，会多次用到这两个类属性。

value属性是保存Boolean类的原生类型的值，也是被修饰为final，不可改变。只有有参构造器，没有无参构造器：

```
public Boolean(boolean value) {
        this.value = value;
}

public Boolean(String s) {
        this(parseBoolean(s));
}
```

构造器的参数为boolean类型和String类型，最后都是将boolean值保存在value属性中。

Boolean类的方法都比较简单，不用分析。

```
//返回包装类型的原生类型
public boolean booleanValue() {
        return value;
 }
//返回原生类型的包装类型
public static Boolean valueOf(boolean b) {
        return (b ? TRUE : FALSE);
}
```

```
//根据字符串返回包装类
public static Boolean valueOf(String s) {
        return parseBoolean(s) ? TRUE : FALSE;
}

//根据原生类型的值返回字符串“true”和“false”
 public static String toString(boolean b) {
        return b ? "true" : "false";
 }

public String toString() {
        return value ? "true" : "false";
}
//equals方法
 public boolean equals(Object obj) {
        if (obj instanceof Boolean) {
            return value == ((Boolean)obj).booleanValue();
        }
        return false;
 }
//比较方法
public static int compare(boolean x, boolean y) {
        return (x == y) ? 0 : (x ? 1 : -1);
}

// 返回hashCode，重点关注
public int hashCode() {
        return Boolean.hashCode(value);
}

public static int hashCode(boolean value) {
        return value ? 1231 : 1237;
}
```

上面的方法是Boolean比较常用的方法，逻辑比较简单，一看就懂的代码，只分析equals方法和hashCode方法，子类在继承Object类时，只要重写equals方法，那么必须重写hashCode。equals方法中用instanceof判断传入的类型是否是Boolean类型，如果是，将传入类型的原生类型与value进行比较，否则，返回false。**hashCode方法中，如果value为true，则返回1231，否则返回1237，所有值为true的Boolean hashcode都是1231 ，为false的都是1237**。为什么是返回这两个数？返回其他两个数可以吗？

1231和1237仅仅因为是任意的质数，这两个质数可以是任何两个大质数都可以。采用质数的原因是可以减少哈希冲突，假设使用1000 和2000来替换1231和1237，当Boolean类作为哈希表的元素时，true和false将分别插入哈希表的 1000% N 和2000% N的位置上(其中N是bucket的数量)。

- 当N等于8时，1000 % 8 等于 2000 % 8；
- 当N等于10时，1000 % 10 等于 2000 % 10；
- 当N等于20时，1000 % 20 等于 2000 % 20；
- .....

当使用1000 和2000作为Boolean类hashCode方法返回的值，那么将会产生大量的冲突。1000和2000有很多因子，因此选择质数，因为它们不太可能与桶大小有任何公因数。

为什么是大的质数，2和3可以吗？

这两个数使用较小的质数时，当哈希表中存在大量的Boolean对象时，就有可能导致对象分布不均匀，哈希查找性能也会降低。

上述hashCode方法的解释参考网址为：https://stackoverflow.com/questions/3912303/boolean-hashcode