第三章 对于所有对象都通用的方法

2018年7月10日

17:57

**第8条：覆盖equals时请准守通用的约定**

**不覆盖equals方法的情况：**

* 类的每个实例本质上都是唯一的。
* 不关心类是否提供了“逻辑相等的测试功能“。
* 超类已经覆盖了equals，从超类继承过来的行为对于子类也是合适的。
* 类时私有的或是包级私有的，可以确定它的的equals方法永远不会被调用。

**覆盖equals方法时的通用约定(equals方法实现了等价关系)：**

* 自反性
  + 违背自反性，可能导致该类无法使用集合。
* 对称性
* 传递性
  + 无法在增加新的组件，同时又保留equals预定。
  + 用getClass测试代替instanceof测试并不适应大多数场景。
  + 里氏替换原则：一个类型的任何重要属性也将适用于它的子类型。
* 一致性
  + 无论类是否是不可变，都不要使equals方法依赖于不可靠的资源。（java.nte.URL是反例，会依赖于主机可变的IP）
* 对于任何非null的引用值，x.equals(null)必须返回false（非空性）
  + 进行转换之前，equals方法必须使用instanceof操作符，检查器参数是否为正常的类型：

**实现高质量equals方法的诀窍：**

1. 使用==操作符检查”参数是否为这个对象的引用“
2. 使用instanceof操作符检查“参数是否为这个对象的引用”。
3. 参数转换成正确的类型。
4. 对与该类中的每个“关键”域，检查参数中的域是否与该对象中对应域向匹配。

* 既不是float也不是double类型的基本类型域，可以使用==操作符进行比较。
* 对象引用域，可以递归调用equals方法
* float域，使用Float.compare方法比较
* double域，使用Double.compare方法比较。

1. 当你编写完成equals方法之后，应该问自己三个问题：他是否是对称的、传递的、一致的？

* 覆盖equals时总要覆盖hashCode
* 不要企图让equals方法过于智能
* 不要将equals声明中的Object对象替换为其他类型。

**第9条：覆盖equals时总要覆盖hashCode**

每个覆盖了equals方法的类中，必须覆盖hashCode方法。

* 在应用程序的执行时期，在未修改情况下，没下hashCode方法都始终如一地返回同一个整数。
* 相等的对象必须具有相等的散列码(hash code)。
* 如果两个对象根据equals(Object)方法比较是不相等的，他们的hashCode最好不相等。因为这样有利于提高散列表的性能。

散列函数应该把集合中不相等的实例均匀地分布到所有可能的散列值上:

1. 非零常数值用int类型保存在result。
2. 对于对象中每个关键域f(指equals方法中涉及的每个域)，完成以下步骤：
   1. 为该域计算int类型的散列码c:
      1. Boolan -> (f?1:0)
      2. Byte, char, short, int -> (int) f
      3. Long -> (int)(f^(f>>>32))
      4. Float -> Float.floatToLong(f)
      5. Double -> Double.doubleTOLongBits(f)
      6. 对象引用 -> 递归调用hashCode方法，如果为null,则返回零。
      7. 数组 -> 把每个元素的域单独递归处理。如果每个元素都很重要，使用Arrays.hashCode方法
   2. 按照下面的公式，把步骤2.a中计算得到的散列码c合并到result中：

result = 31 \* result + c

1. 返回result

如果类不可变，且计算散列码的开销比较大，可以考虑把散列码缓存在对象内部。

可以选择“延迟初始化”散列码。

不要试图从散列码计算中排除掉一个对象的关键部分来提高性能。

**第10条：始终要覆盖toString**

在实际应用中，toString方法应该返回对象中包含的所有值得关注的信息。

无论你是否决定指定格式，都应该在文档中明确表明的意图。如果要指定格式，更要严格执行。

无论是否指定了格式，都为toString返回值中包含的所有信息，提供一种编程式的访问途径。

**第11条：谨慎地覆盖clone**

cloneale接口的目的是作一个mixin接口，表明这样的对象允许克隆。

clone方法就是另一个构造器；你必须确保它不会伤害到原始对象，并确保正确地创建被克隆对象中的约束条件。

clone架构与引用可变对象的final域的正常用法是不相兼容的。

另一个实现对象拷贝的好办法是提供一个拷贝构造器或拷贝工厂。

**第12条：考虑实现Comparable接口**

compareTo方法中的域的比较是顺序的比较，而不是同性的比较。

浮点域用Double.compare或者Float.compare.,而不同关系操作符。

使用相减方法是，如果最小和最大的可能值域之差小于或等于INTEGER>MAX\_VALUE((231)-1), 否者就不要使用这种方法。