Java理论与实践：哈希 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/252/2021_2022_Java_E7_90_86_E8_AE_BA_c104_252381.htm 每个Java对象都有hashCode()和equals()方法。许多类忽略（Override)这些方法的缺省实施,以在对象实例之间提供更深层次的语义可比性。在Java理念和实践这一部分，Java开发人员Brian Goetz向您介绍在创建Java类以有效和准确定义hashCode()和equals()时应遵循的规则和指南。虽然Java语言不直接支持关联数组 -- 可以使用任何对象作为一个索引的数组 -- 但在根Object类中使用hashCode()方法明确表示期望广泛使用HashMap(及其前辈Hashtable)。理想情况下基于散列的容器提供有效插入和有效检索；直接在对象模式中支持散列可以促进基于散列的容器的开发和使用。 定义对象的相等性Object类有两种方法来推断对象的标识：equals()和hashCode()。一般来说，如果您忽略了其中一种，您必须同时忽略这两种，因为两者之间有必须维持的至关重要的关系。特殊情况是根据equals() 方法，如果两个对象是相等的，它们必须有相同的hashCode()值(尽管这通常不是真的)。特定类的equals()的语义在Implementer的左侧定义；定义对特定类来说equals()意味着什么是其设计工作的一部分。Object提供的缺省实施简单引用下面等式：public boolean equals(Object obj) { return (this == obj). }在这种缺省实施情况下，只有它们引用真正同一个对象时这两个引用才是相等的。同样，Object提供的hashCode()的缺省实施通过将对象的内存地址对映于一个整数值来生成。由于在某些架构上，地址空间大于int值的范围，两个不同的对象有相同的hashCode()是

可能的。如果您忽略了hashCode()，您仍旧可以使用System.identityHashCode()方法来接入这类缺省值。忽略equals() -- 简单实例缺省情况下，equals()和hashCode()基于标识的实施是合理的，但对于某些类来说，它们希望放宽等式的定义。例如，Integer类定义equals() 与下面类似：public boolean equals(Object obj) {return (obj instanceof Integer amp. intValue() == ((Integer) obj).intValue()). }在这个定义中，只有在包含相同的整数值的情况下这两个Integer对象是相等的。结合将不可修改的Integer，这使得使用Integer作为HashMap中的关键字是切实可行的。这种基于值的Equal方法可以由Java类库中的所有原始封装类使用，如Integer、Float、Character和Boolean以及String(如果两个String对象包含相同顺序的字符，那它们是相等的)。由于这些类都是不可修改的并且可以实施hashCode()和equals()，它们都可以做为很好的散列关键字。 为什么忽略 equals()和hashCode()?如果Integer不忽略equals()和 hashCode()情况又将如何?如果我们从未在HashMap或其它基于散列的集合中使用Integer作为关键字的话，什么也不会发生。但是，如果我们在HashMap中使用这类Integer对象作为关键字，我们将不能够可靠地检索相关的值，除非我们在get()调用中使用与put()调用中极其类似的Integer实例。这要求确保在我们的整个程序中，只能使用对应于特定整数值的Integer对象的一个实例。不用说，这种方法极不方便而且错误频频。 Object的interface contract要求如果根据 equals()两个对象是相等的，那么它们必须有相同的hashCode()值。当其识别能力整个包含在equals()中时，为什么我们的根对象类需要hashCode() ? hashCode()方法纯粹用于提高效率。Java平

台设计人员预计到了典型Java应用程序中基于散列的集合类（Collection Class)的重要性--如Hashtable、HashMap和HashSet，并且使用equals()与许多对象进行比较在计算方面非常昂贵。使所有Java对象都能够支持 hashCode()并结合使用基于散列的集合，可以实现有效的存储和检索。 实施equals()和hashCode()的需求实施equals()和 hashCode()有一些限制，Object文件中列举出了这些限制。特别是equals()方法必须显示以下属性：☆ Symmetry：两个引用，a和 b,a.equals(b) if and only if b.equals(a)☆ Reflexivity：所有非空引用， a.equals(a)☆ Transitivity：If a.equals(b) and b.equals(c), then a.equals(c)☆ Consistency with hashCode()：两个相等的对象必须有相同的hashCode()值Object的规范中并没有明确要求equals()和 hashCode() 必须一致 -- 它们的结果在随后的调用中将是相同的，假设“ 不改变对象相等性比较中使用的任何信息。 ”这听起来象“ 计算的结果将不改变，除非实际情况如此。 ”这一模糊声明通常解释为相等性和散列值计算应是对象的可确定性功能，而不是其它。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com