Java `hashCode()`

From Wikipedia, the free encyclopedia

In the Java programming language, every class implicitly or explicitly provides a **`hashCode()`** method, which digests the data stored in an instance of the class into a single hash value (a 32-bit signed integer). This hash is used by other code when storing or manipulating the instance – the values are intended to be evenly distributed for varied inputs for use in clustering. This property is important to the performance of hash tables and other data structures that store objects in groups ("buckets") based on their computed hash values. Technically, in Java, `hashCode()` by default is a native method, meaning, it has the modifier 'native', as it is implemented directly in the native code in the JVM.

Contents

`hashCode()` in general

All the classes inherit a basic hash scheme from the fundamental base class `java.lang.Object`, but instead many override this to provide a hash function that better handles their specific data. Classes which provide their own implementation must override the object method `public int hashCode()`.

The general contract for overridden implementations of this method is that they behave in a way consistent with the same object's `equals()` method: that a given object must consistently report the same hash value (unless it is changed so that the new version is no longer considered "equal" to the old), and that two objects which `equals()` says are equal *must* report the same hash value. There's no requirement that hash values be consistent between different Java implementations, or even between different execution runs of the same program, and while two *unequal* objects having different hashes is very desirable, this is not mandatory (that is, the hash function implemented doesn't need to be a perfect hash).[1]

For example, the class `Employee` might implement its hash function by composing the hashes of its members:

```
public class Employee {
    int        employeeId;
    String     name;
    Department dept;

    // other methods would be in here

    @Override
    public int hashCode() {
        int hash = 1;
        hash = hash * 17 + employeeId;
        hash = hash * 31 + name.hashCode();
        hash = hash * 13 + (dept == null ? 0 : dept.hashCode());
        return hash;
    }
}
```

The `java.lang.String` hash function

In an attempt to provide a fast implementation, early versions of the Java `String` class provided a `hashCode()` implementation that considered at most 16 characters picked from the string. For some common data this worked very poorly, delivering unacceptably clustered results and consequently slow hashtable performance.[2]

From Java 1.2, `java.lang.String` class implements its `hashCode()` using a product sum algorithm over the entire text of the string.[2] An instance `s` of the `java.lang.String` class, for example, would have a hash code $h(s)$ defined by

$$h(s) = \sum_{i=0}^{n-1} s[i] \cdot 31^{n-1-i}$$

where terms are summed using Java 32-bit `int` addition, $s[i]$ denotes the ith character of the string, and n is the length of `s`.[3] [4]

References

- "Always override hashCode when you override equals" in Bloch, Joshua (2008), *Effective Java* (2nd ed.), Addison-Wesley, ISBN 978-0-321-35668-0

1. java.lang.Object.hashCode() documentation (http://download.oracle.com/javase/1.5.0/docs/api/java/lang/Object.html#hashCode%28%29), Java SE 1.5.0 documentation, Oracle Inc.
2. Suggested Fix by Bloch for Java 1.2 (http://bugs.java.com/bugdatabase/view_bug.do?bug_id=4045622), JDK-4045622, Oracle Inc.
3. java.lang.String.hashCode() documentation (http://docs.oracle.com/javase/1.5.0/docs/api/java/lang/String.html#hashCode%28%29), Java SE 1.5.0 documentation, Oracle Inc.
4. Choice of hash function -> The String hash function" (http://web.archive.org/web/20130703081745/http://www.cogs.susx.ac.uk/courses/dats/notes/html/node114.html), Data Structures course notes (2006), Peter M Williams, University of Sussex School of Information

External links

- "Java theory and practice: Hashing it out" (http://www.ibm.com/developerworks/library/j-jtp05273/), Brian Goetz, IBM Developer Works article, 27 May 2003
- "How the String hash function works (and implications for other hash functions)" (http://www.javamex.com/tutorials/collections/hash_function_technical.shtml), Neil Coffey, Javamex
- Why should hash functions use a prime number modulus? (https://stackoverflow.com/questions/1145217/why-should-hash-functions-use-a-prime-number-modulus) (stackoverflow)
- What is a sensible prime for hashcode calculation? (https://stackoverflow.com/questions/1835976/what-is-a-sensible-prime-for-hashcode-calculation) (stackoverflow)

Retrieved from "https://en.wikipedia.org/w/index.php?title=Java_hashCode()&oldid=731467698"

Categories: Java (programming language) | Hash function (non-cryptographic) | Checksum algorithms

---

- This page was last modified on 25 July 2016, at 15:17.
- Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. By using this site, you agree to the Terms of Use and Privacy Policy. Wikipedia® is a registered trademark