Why does Hashmap allows a null key?

Ask Question

Actually I read lots of post for this question, but did not get the exact reason/answer for **Why does Hashmap allows a null key?**. Please can anyone explain me the exact answer with an example. Thanks in advance.

java collections hashmap

edited Dec 11 '17 at 8:33
nmkyuppie
623 1 5 24

asked Dec 11 '17 at 5:57
hp36
2 1

Because they didn't feel a need to disallow it. – Andreas Dec 11 '17 at 5:59

You can use a null key when you can get a default value from the map when there is no key maybe.. Asking why only one allowance does not seem a reasonable question. – Dogukan Zengin Dec 11 '17 at 6:02

I agree, I don't get the exact reason for above question either. Why did you ask this question? What problem do you have with the functionality? – Andreas Dec 11 '17 at 6:03

See also: stackoverflow.com/a/9912908/1441122 and stackoverflow.com/a/9298113/1441122 – Stuart Marks Dec 11 '17 at 21:37

4 Answers

Ask yourself: if HashMap allowed more than one `null` key, how would the map object distinguish between them?

Hint: there is only one `null` value.

Alternative interpretation of your question

> why hashmap allowed [a] null key?

Because it is useful in some circumstances, and because there is no real *semantic* need to disallow it[1, 2].

By contrast, with `TreeMap` `null` keys are disallowed because supporting them would be difficult given the implications of orderings involving `null`.

- Given that the specified semantics for `Comparable` is to throw NPE.
- `Comparator` is *allowed* to order `null`, but it is not required to. And many common implementations don't.

So if `null` was allowed with `TreeMap`, then the behavior of a map could be different depending on whether a `Comparator` or a `Comparable` is used. Messy.

1 - At least, that was the view when they specified `HashMap` in Java 1.2 back in 1998. Some of the designers may have changed their minds since then, but since the behavior is *clearly specified* it cannot be changed without messing up compatibility. It won't happen ...

2 - Support for `null` keys requires some special case code in `HashMap` which at least adds complexity to the implementation. It is not clear if it is a performance overhead for `HashMap`, since there would still need to be an implicit test for a `null` keys even if `null` keys were not allowed. This is most likely down in the noise.

edited Dec 11 '17 at 10:03

answered Dec 11 '17 at 5:59

Stephen C

Here the question is why it is allowed. Your answer is for "Why one key is allowed" – nmkyuppie Dec 11 '17 at 6:01

@nmkyuppie - To a native English speaker "one" implies "only one" in this context. There is a strong difference in meaning between "one" and "a". Granted, the OP's command of English is not good. But that doesn't allow you or anyone else to say definitively what he actually meant. – Stephen C Dec 11 '17 at 6:09

@StephenC if that's the case, then why is `CHM` disallowing it? I hardly agree with this answer - as I've heard Stuart Marks saying that the decision to have a null key is *now* viewed as a mistake that was done initially. – Eugene Dec 11 '17 at 9:11

@Eugene See this answer, that cites some release notes from JDK 1.2, presumably written by Joshua Bloch, mentioning null support was for consistency with `TreeMap` (which I don't understand) and that also mentions customer requests (which is understandable). It also links to an email message from Doug Lea where he explain's CHM's behavior and where he says he has disagreed with Bloch on this issue regarding non-concurrent collections. However.... – Stuart Marks Dec 11 '17 at 21:45

Java engineers must have realized that having a null key and values has its uses like using them for default cases. So, they provided HashMap class with collection framework in Java 5 with capability of storing null key and values.

The put method to insert key value pair in HashMap checks for null key and stores it at the first location of the internal table array. It isn’t afraid of the null values and does not throw NullPointerException like Hashtable does.

Now, there can be only one null key as keys have to be unique although we can have multiple null values associated with different keys.

this link can answer more hashmap null key explained

answered Dec 11 '17 at 6:04

The javadoc for HashMap.put clearly states:

> Associates the specified value with the specified key in this map. If the map previously contained a mapping for the key, the old value is replaced.

It clearly states what happens when you do a put with a key which was already in the map. The specific case of key == null behaves in the same way: you can't have two different mappings for the null key (just like you can't for any other key).

answered Dec 11 '17 at 6:05
Amol Raje
314 11

This is still viewed as a **mistake** done in early implementations of `HashMap` that unfortunately people where (are?) relying on (as well as some particular order until java-8). The thing is, having this `null` key you could always pass some metadata along with your actual data "for free". Notice that all new collections `ConcurrentHashMap`, `Map.of` (in java-9), etc - all prohibit nulls to start with.

answered Dec 11 '17 at 9:09
Eugene
50.6k 8 66 114
