Why is it useful to have null values or null keys in hash maps?

`Hashtable` does not allow null keys or values, while `HashMap` allows null values and 1 null key.

Questions:

1. Why is this so?
2. How is it useful to have such a key and values in HashMap?

java hashmap

edited Sep 1 '10 at 21:11

Shog9 ♦
123k 30 203 225

asked Sep 1 '10 at 20:43

sab
3,409 9 32 43

What do you mean by the "importance"? – Michael Mrozek Sep 1 '10 at 20:51

@Michael Mrozek: Probably significance. "Importance" isn't an AWFUL word to choose here, if you think of it as "why is `null` important enough to have those special designations". – Platinum Azure Sep 1 '10 at 21:05

6 Answers

- Hashtable calculates a hash for each key by calling `hashCode` on each key. This would fail if the key were null, so this could be a reason for disallowing nulls as keys.
- The method `Hashtable.get` returns null if the key is not present. If null were a valid value it would be ambiguous as to whether null meant that the key was present but had value null, or if the key was absent. Ambiguity is bad, so this could be a reason for disallowing nulls as values.

However it turns out that sometimes you do actually want to store nulls so the restrictions were removed in HashMap. The following warning was also included in the documentation for `HashMap.get`:

> A return value of null does not necessarily indicate that the map contains no mapping for the key; it is also possible that the map explicitly maps the key to null.

> 2. How is it useful to have such a key and values in HashMap?

It is useful to explicitly store null to distinguish between a key that you *know* exists but doesn't have an associated value and a key that doesn't exist. An example is a list of registered users and their birthdays. If you ask for a specific user's birthday you want to be able to distinguish between that user not existing and the user existing but they haven't entered their birthday.

I can't think of any (good) reason for wanting to store null as a key, and in general I'd advise against using null as a key, but presumably there is at least one person somewhere that needs that keys that can be null.

edited Mar 5 at 16:58

Nike15
364 1 13

answered Sep 1 '10 at 20:58

Mark Byers
550k 116 1294
1310

Personally, I don't think `null` values make any sense. They force you to query map twice. Instead of single `map.get()`, you have to use `map.get( )` and `map.containsKey()` to ascertain that key is known to the system. No wonder `null` values are not permitted in many new specializations of `Map` interface. – Alexander Pogrebnyak Sep 1 '10 at 21:25

You say "It is useful to explicitly store null to distinguish between a key that ...", but *how*, it seems the two situations both reply null on get, or are you thinking about the call of another method to make the check? – Chris2048 Aug 10 '13 at 1:56

Well, I think Mark Byers answered perfectly, so just a simple example where null values and keys can be useful:

Imagine you have an expensive function that always returns the same result for the same input. A map is a simple way for caching its results. Maybe sometimes the function will return null, but you need to save it anyway, because the execution is expensive. So, null values must be stored. The same applies to null key if it's an accepted input for the function.

answered Sep 1 '10 at 23:46

sinuhepop
12.5k 12 55 90

HashTable is very old class , from JDK 1.0.The classes which are in place from **JDK 1.0** are called **Legacy** classes and by default they are **synchronized**.

To understand this, first of all you need to understand comments written on this class by author. “This class implements a hashtable, which maps keys to values. Any non-null object can be used as a key or as a value. To successfully store and retrieve objects from a hashtable, the objects used as keys must implement the hashCode method and the equals method.”

HashTable class is implemented on hashing mechanism, means to store any key-value pair, its required

Hash Table were introduced like HashMap which allow one null key and multiple null values.

For HashMap, it allows one null key and there is a null check for keys, if the key is null then that element will be stored in a zero location in Entry array.

We cannot have more than one Null key in HashMap because **Keys are unique** therefor only one Null key and many Null values are allowed.

USE - Null key we can use for some default value.

Modified and better implementation of HashTable was later introduced as **ConcurrentHashMap**.

edited Feb 6 at 19:12

answered Apr 6 '17 at 12:38

shubham malik

31 3

In addition to what answered by Mark Bayers,, Null is considered as data and it has to be stored as a value for further checking. In many cases null as value can be used to check if key entry is there but no value is assigned to it, so some action can be taken accordingly. This can be done by first checking if key is there, and then getting value. There is one more case in which just put whatever data is coming(without any check). All the checks are applied to it after getting it.

Whereas null as a key, i think can be used to define some default data. Usually null as a key does not make much sense.

answered Jan 4 '14 at 1:46

Amit Agrawal

11 2

Sir HashMap is also internally uses hashCode() method for inserting an element in HashMap, so I think this will be not the proper reason for "why HashTable allow null key"

It will make the Map interface easier to use / less verbose. `null` is a legitimate value for reference types. Making the map able to handle null keys and values will eliminate the need for null checking before calling the api. So the map api create less "surprises" during runtime.

For example, it is common that map will be used to categorize a collection of homogeneous objects based a single field. When map is compatible with null, the code will be more concise as it is just a simple loop without any if statement (of course you will need to make sure the collection does not have null elements). Fewer lines of code with no branch / exception handling will be more likely to be logically correct.

On the other hand, not allowing null will not make the map interface better/safer/easier to use. It is not practical to rely on the map to reject nulls - that means exception will be thrown and you have to catch and handle it. Or, to get rid of the exception you will have to ensure nothing is null before calling the map methods - in which case you don't care if map accepts null since you filtered the input anyway.

answered Jun 3 '16 at 8:15

Xinchao

477 2 17