HashMap.get() != null   vs   HashMap.containsKey()

I want to retrieve value of a key in hash map and throw exception if key not found, which one would be better..

```
value = map.get(key)
if (value != null)
  proceed with rest of logic
else
  throw exception..
```

OR

```
if (!map.containsKey(key))
  throw exception
value = map.get(key)
proceed with rest of logic
```

ANSWER

Only the second one actually works, so there is no point in considering the first one. The documentation for the "get" method says:

"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. The containsKey method may be used to distinguish these two cases."

What is the use of adding a null key or value to a HashMap in Java?

Answer 1 (null key):

I use them often in maps to represent the default case (i.e. the value that should be used if a given key isn't present):

```
Map<A, B> foo;
A search;
B val = foo.containsKey(search) ? foo.get(search) : foo.get(null);
```

`HashMap` handles null keys specially (since it can't call `.hashCode()` on a null object), but null values aren't anything special, they're stored in the map like anything else

Answer 2 (null value):

One example would be for modeling tree nodes.
If you are using a HashMap to encapsulate a tree structure.
Where the key is the parent and the value is list of children.
Then the children of the null key would be all the top level nodes.

Answer 3 (null value):

One example of usage for `null` *values* is when using a `HashMap` as a cache for results of an expensive operation (such as a call to an external web service) which may return `null`.
Putting a `null` value in the map then allows you to distinguish between the case where the operation has not been performed for a given key (`cache.containsKey(someKey)` returns `false`), and where the operation has been performed but returned a `null` value (`cache.containsKey(someKey)` returns `true`, `cache.get(someKey)` returns `null`).
Without `null` values, you would have to either put some special value in the cache to indicate a `null` response, or simply not cache that response at all and perform the operation every time.

Answer 4 (null key):

Another example: I use it to group Data by date. But some data doesn't have a date. I can group it with the header "NoDate"