How Set/HashSet works internally in Java

Posted by: [Ima Miri](https://examples.javacodegeeks.com/author/ima-miri/) in [Core Java](https://examples.javacodegeeks.com/category/core-java/) November 19th, 2015 [0](https://examples.javacodegeeks.com/core-java/sethashset-works-internally-java/#respond) 9746 Views

This is one of frequently asked java questions in interviews. However, not many programmers know that HashSet uses HashMap internally in java. Here we discuss in details how HashSet works inside java.

1. Set and HashSet

Set is a collection that contains no duplicate elements. So, it can contain at most one null.

HashSet implements Set interface in java. It is not synchronized and is not thread safe.

Here is an example that how to use HashSet in java:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10 | public class Exercise {     public static void main(String[] args) throws IOException {            HashSet hashSet = new HashSet();          hashSet.add(20);          hashSet.add("HashSet Tutorial");          hashSet.add("JavaCodeGeeks");          System.out.println("Set contains " + hashSet);     }  } |

Output:

|  |  |
| --- | --- |
| 1 | Set contains [20, Hash Tutorial, JavaCodeGeeks] |

Lets see what is the result, if we add duplicate elements to the above set:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12 | public class Exercise {     public static void main(String[] args) throws IOException {            HashSet hashSet = new HashSet();          hashSet.add(20);          hashSet.add("HashSet Tutorial");          hashSet.add("JavaCodeGeeks");          hashSet.add(20);          hashSet.add("HashSet Tutorial");          System.out.println("Set contains " + hashSet);     }  } |

If we look at the output, we will see the same result as before.

|  |  |
| --- | --- |
| 1 | Set contains [20, Hash Tutorial, JavaCodeGeeks] |

Here is the question, what happened internally when the duplicate elements are passed to the HashSet? The add(e) method in HashSet returns false when the element exists in the HashSet, otherwise it returns true.

Lets see in more details that how the add(e) method returns false.

2. HashSet implementation internally in Java

When we have a look at the HashSet.java in java API, we can see the following code:

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22 | public class HashSet      extends AbstractSet      implements Set, Cloneable, java.io.Serializable  {        private transient HashMap<E,Object> map;        // Dummy value to associate with an Object in the backing Map      private static final Object PRESENT = new Object();        public HashSet() {          map = new HashMap<>();      }        public boolean add(E e) {          return map.put(e, PRESENT)==null;      }        /\*\*      \* Some code      \*/  } |

Set achieves the uniqueness in its elements through HashMap. In HashMap, each key is unique. So, when an object of HashSet is created, it will create an object of HashMap. When an element is passed to Set, it is added as a key in the HashMap in the add(Element e) method. Now, a value needs to be associated to the key. Java uses a Dummy value (new Object) which is called PRESENT in HashSet.

In HashMap, the put(Key k,Value V) method returns:

1. null, if the key is unique. The key will be added to the map.
2. old value of the key, if the key is duplicated.

|  |  |
| --- | --- |
| 1  2  3 | public V put(K key, V value) {  /\* Some code \*/  } |

In HashSet add(e) method, the return value of map.put(key,value) method will be checked with null value.

|  |  |
| --- | --- |
| 1  2  3 | public boolean add(E e) {  return map.put(e, PRESENT)==null;  } |

If map.put(key,value) returns null, then map.put(e, PRESENT)==null will return true and element is added to the HashSet.

If map.put(key,value) returns the old value of the key, then map.put(e, PRESENT)==nullwill return false and element wont be added to the HashSet.

remove() method also works in the same way.

|  |  |
| --- | --- |
| 1  2  3 | public boolean remove(Object o) {      return map.remove(o)==PRESENT;  } |

If you know how HashMap works internally in Java, you can explain working of HashSet provided. As you know HashSet uses same values for all keys.

It is really important to override equals() and hashCode() for any object you are going to store in HashSet. Because the object is used as key in map, must override those method. So, make the object Immutable or effective immutable if possible.