Hashtable in java with example

BY CHAITANYA SINGH | FILED UNDER: JAVA COLLECTIONS

This class implements a hash table, which maps keys to values. Any non-null object can be used as a key or as a value. Hashtable is similar to HashMap except it is synchronized. There are few more differences between HashMap and Hashtable class, you can read them in detail at: Difference between HashMap and Hashtable.

In this tutorial we will see how to create a Hashtable, how to populate its entries and then we will learn how to display its key-value pairs using Enumeration. At the end of this article we will see Hashtable tutorials and methods of Hashtable class.

Example

```
import java.util.Hashtable;
import java.util.Enumeration;

public class HashtableExample {

 public static void main(String[] args) {

   Enumeration names;
   String key;

   // Creating a Hashtable
   Hashtable<String, String> hashtable =
              new Hashtable<String, String>();

   // Adding Key and Value pairs to Hashtable
   hashtable.put("Key1","Chaitanya");
   hashtable.put("Key2","Ajeet");
   hashtable.put("Key3","Peter");
   hashtable.put("Key4","Ricky");
   hashtable.put("Key5","Mona");

   names = hashtable.keys();
   while(names.hasMoreElements()) {
      key = (String) names.nextElement();
      System.out.println("Key: " +key+ " & Value: " +
      hashtable.get(key));
   }
 }
}
```

Output:

```
Key: Key4 & Value: Ricky
Key: Key3 & Value: Peter
Key: Key2 & Value: Ajeet
Key: Key1 & Value: Chaitanya
```

```
Key: Key5 & Value: Mona
```

Hashtable tutorials

- Hashtable example
- Sort Hashtable
- Hashtable Iterator example
- Check key-value existence in Hashtable
- Remove mapping from Hashtable
- Remove all mappings from Hashtable
- Get size of Hashtable
- Hashtable vs HashMap

Methods of Hashtable class:

1) `void clear()`: Removes all the key-value mappings from Hashtable and makes it empty. Clears this hashtable so that it contains no keys..

2) `Object clone()`: Creates a shallow copy of this hashtable. All the structure of the hashtable itself is copied, but the keys and values are not cloned. This is a relatively expensive operation.

3) `boolean contains(Object value)`: Tests if some key maps into the specified value in this hashtable. This operation is more expensive than the containsKey method.
Note that this method is identical in functionality to containsValue, (which is part of the Map interface in the collections framework).

4) `boolean isEmpty()`: Tests if this hashtable maps no keys to values.

5) `Enumeration keys()`: Returns an enumeration of the keys contained in the hash table.

6) `Object put(Object key, Object value)`: Maps the specified key to the specified value in this hashtable.

7) `void rehash()`: Increases the size of the hash table and rehashes all of its keys.

8) `Object remove(Object key)`: Removes the key (and its corresponding value) from this hashtable.

9) `int size()`: Returns the number of key-value mappings present in Hashtable.

10) `String toString()`: Returns the string equivalent of a hash table.

11) `boolean containsKey(Object key)`: Tests if the specified object is a key in this hashtable.

12) boolean containsValue(Object value): Tests if the specified object is a value in this hashtable. Returns true if some value equal to value exists within the hash table. Returns false if the value isn’t found.

13) `Enumeration elements()`: Returns an enumeration of the values contained in the hash table.

14) `Object get(Object key)`: Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key.