All Packages   This Package   Previous   Next

Class java.util.Hashtable

```
java.lang.Object
   |
   +----java.util.Dictionary
           |
           +----java.util.Hashtable
```

public class **Hashtable**
extends Dictionary

Hashtable class. Maps keys to values. Any object can be used as a key and/or value.

To sucessfully store and retrieve objects from a hash table the object used as the key must implement the hashCode() and equals() methods.

This example creates a hashtable of numbers. It uses the names of the numbers as keys:

```
        Hashtable numbers = new Hashtable();
        numbers.put("one", new Integer(1));
        numbers.put("two", new Integer(2));
        numbers.put("three", new Integer(3));
```

To retrieve a number use:

```
        Integer n = (Integer)numbers.get("two");
        if (n != null) {
            System.out.println("two = " + n);
        }
```

See Also:
hashCode, equals
Version:
1.29, 09/28/95
Author:
Arthur van Hoff

Constructor Index

o **Hashtable**(int, float)

Constructs a new, empty hashtable with the specified initial capacity and the specified load factor.

o **Hashtable**(int)

Constructs a new, empty hashtable with the specified initial capacity.

o **Hashtable**()

Constructs a new, empty hashtable.

Method Index

o **clear**()

Clears the hash table so that it has no more elements in it.

o **clone**()

Creates a clone of the hashtable.

o **contains**(Object)

Returns true if the specified object is an element of the hashtable.

o **containsKey**(Object)

Returns true if the collection contains an element for the key.

o **elements**()

Returns an enumeration of the elements.

o **get**(Object)

Gets the object associated with the specified key in the hashtable.

o **isEmpty**()

Returns true if the hashtable contains no elements.

o **keys**()

Returns an enumeration of the hashtable's keys.

o **put**(Object, Object)

Puts the specified element into the hashtable, using the specified key.

o **rehash**()

Rehashes the content of the table into a bigger table.

o **remove**(Object)

Removes the element corresponding to the key.

o **size**()

Returns the number of elements contained in the hashtable.

o **toString**()

Converts to a rather lengthy String.

Constructors

o **Hashtable**

```
public Hashtable(int initialCapacity,
                 float loadFactor)
```

Constructs a new, empty hashtable with the specified initial capacity and the specified load factor.

Parameters:

initialCapacity – the initial number of buckets

loadFactor – a number between 0.0 and 1.0, it defines the threshold for rehashing the hashtable into a bigger one.

Throws: IllegalArgumentException

If the initial capacity is less than or equal to zero.

Throws: IllegalArgumentException

If the load factor is less than or equal to zero.

o **Hashtable**

```
public Hashtable(int initialCapacity)
```

Constructs a new, empty hashtable with the specified initial capacity.

Parameters:

initialCapacity – the initial number of buckets

o **Hashtable**

```
public Hashtable()
```

Constructs a new, empty hashtable. A default capacity and load factor is used. Note that the hashtable will automatically grow when it gets full.

Methods

o **size**

```
public int size()
```

Returns the number of elements contained in the hashtable.

Overrides:

size in class Dictionary

o **isEmpty**

```
public boolean isEmpty()
```

Returns true if the hashtable contains no elements.

Overrides:

isEmpty in class Dictionary

o **keys**

```
public synchronized Enumeration keys()
```

Returns an enumeration of the hashtable's keys.

Overrides:

keys in class Dictionary

See Also:

elements, Enumeration

o **elements**

```
public synchronized Enumeration elements()
```

Returns an enumeration of the elements. Use the Enumeration methods on the returned object to fetch the elements sequentially.

Overrides:
elements in class Dictionary

See Also:
keys, Enumeration

o **contains**

```
public synchronized boolean contains(Object value)
```

Returns true if the specified object is an element of the hashtable. This operation is more expensive than the containsKey() method.

Parameters:
value – the value that we are looking for

Throws: NullPointerException
If the value being searched for is equal to null.

See Also:
containsKey

o **containsKey**

```
public synchronized boolean containsKey(Object key)
```

Returns true if the collection contains an element for the key.

Parameters:
key – the key that we are looking for

See Also:
contains

o **get**

```
public synchronized Object get(Object key)
```

Gets the object associated with the specified key in the hashtable.

Parameters:
key – the specified key

Returns:
the element for the key or null if the key is not defined in the hash table.

Overrides:
get in class Dictionary

See Also:
put

o **rehash**

```
protected void rehash()
```

Rehashes the content of the table into a bigger table. This method is called automatically when the hashtable's size exceeds the threshold.

o put

```
public synchronized Object put(Object key,
                               Object value)
```

Puts the specified element into the hashtable, using the specified key. The element may be retrieved by doing a get() with the same key. The key and the element cannot be null.

Parameters:

key – the specified key in the hashtable

value – the specified element

Throws: NullPointerException

If the value of the element is equal to null.

Overrides:

put in class Dictionary

See Also:

get

Returns:

the old value of the key, or null if it did not have one.

o remove

```
public synchronized Object remove(Object key)
```

Removes the element corresponding to the key. Does nothing if the key is not present.

Parameters:

key – the key that needs to be removed

Returns:

the value of key, or null if the key was not found.

Overrides:

remove in class Dictionary

o clear

```
public synchronized void clear()
```

Clears the hash table so that it has no more elements in it.

o clone

```
public synchronized Object clone()
```

Creates a clone of the hashtable. A shallow copy is made, the keys and elements themselves are NOT cloned. This is a relatively expensive operation.

Overrides:

clone in class Object

o **toString**

```
 public synchronized String toString()
```

Converts to a rather lengthy String.

Overrides:

toString in class Object

All Packages   This Package   Previous   Next