java.util

Class Hashtable

java.lang.Object
    java.util.Hashtable

```
public class Hashtable
extends Object
```

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.

An instance of `Hashtable` has two parameters that affect its efficiency: its *capacity* and its *load factor*. The load factor in the CLDC implementation of the hashtable class is always 75 percent. When the number of entries in the hashtable exceeds the product of the load factor and the current capacity, the capacity is increased by calling the `rehash` method.

If many entries are to be made into a `Hashtable`, creating it with a sufficiently large capacity may allow the entries to be inserted more efficiently than letting it perform automatic rehashing as needed to grow the table.

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 the following code:

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

Since:

JDK1.0, CLDC 1.0

Version:

12/17/01 (CLDC 1.1)

See Also:

Object.equals(java.lang.Object), Object.hashCode(), rehash()

Constructor Summary

Constructors

Constructor and Description
Hashtable() Constructs a new, empty hashtable with a default capacity and load factor.
Hashtable(int initialCapacity) Constructs a new, empty hashtable with the specified initial capacity.

Method Summary

Methods

Modifier and Type	Method and Description
void	**clear**() Clears this hashtable so that it contains no keys.
boolean	**contains**(**Object** value) Tests if some key maps into the specified value in this hashtable.
boolean	**containsKey**(**Object** key) Tests if the specified object is a key in this hashtable.
Enumeration	**elements**() Returns an enumeration of the values in this hashtable.
Object	**get**(**Object** key) Returns the value to which the specified key is mapped in this hashtable.
boolean	**isEmpty**() Tests if this hashtable maps no keys to values.
Enumeration	**keys**() Returns an enumeration of the keys in this hashtable.
Object	**put**(**Object** key, **Object** value) Maps the specified key to the specified value in this hashtable.
protected void	**rehash**() Rehashes the contents of the hashtable into a hashtable with a larger capacity.
Object	**remove**(**Object** key) Removes the key (and its corresponding value) from this hashtable.
int	**size**() Returns the number of keys in this hashtable.
String	**toString**() Returns a rather long string representation of this hashtable.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

Hashtable

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

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

Parameters:

`initialCapacity` - the initial capacity of the hashtable.

Throws:

`IllegalArgumentException` - if the initial capacity is less than zero

Since:

JDK1.0

Hashtable

```
public Hashtable()
```

Constructs a new, empty hashtable with a default capacity and load factor.

Since:

JDK1.0

Method Detail

size

```
public int size()
```

Returns the number of keys in this hashtable.

Returns:

the number of keys in this hashtable.

Since:

JDK1.0

isEmpty

```
public boolean isEmpty()
```

Tests if this hashtable maps no keys to values.

Returns:

`true` if this hashtable maps no keys to values; `false` otherwise.

Since:

JDK1.0

keys

```
public Enumeration keys()
```

Returns an enumeration of the keys in this hashtable.

Returns:

an enumeration of the keys in this hashtable.

Since:

JDK1.0

See Also:

`Enumeration`, `elements()`

elements

```
public Enumeration elements()
```

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

Returns:

an enumeration of the values in this hashtable.

Since:

JDK1.0

See Also:

`Enumeration`, `keys()`

contains

```
public 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.

Parameters:

`value` - a value to search for.

Returns:

`true` if some key maps to the `value` argument in this hashtable; `false` otherwise.

Throws:

`NullPointerException` - if the value is `null`.

Since:

JDK1.0

See Also:

`containsKey(java.lang.Object)`

containsKey

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

Tests if the specified object is a key in this hashtable.

Parameters:

`key` - possible key.

Returns:

`true` if the specified object is a key in this hashtable; `false` otherwise.

Since:

JDK1.0

See Also:

`contains(java.lang.Object)`

get

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

Returns the value to which the specified key is mapped in this hashtable.

Parameters:

`key` - a key in the hashtable.

Returns:

the value to which the key is mapped in this hashtable; `null` if the key is not mapped to any value in this hashtable.

Since:

JDK1.0

See Also:

`put(java.lang.Object, java.lang.Object)`

rehash

```
protected void rehash()
```

Rehashes the contents of the hashtable into a hashtable with a larger capacity. This method is called automatically when the number of keys in the hashtable exceeds this hashtable's capacity and load factor.

Since:

JDK1.0

put

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

Maps the specified `key` to the specified `value` in this hashtable. Neither the key nor the value can be `null`.

The value can be retrieved by calling the `get` method with a key that is equal to the original key.

Parameters:

`key` - the hashtable key.

`value` - the value.

Returns:

the previous value of the specified key in this hashtable, or `null` if it did not have one.

Throws:

`NullPointerException` - if the key or value is `null`.

Since:

JDK1.0

See Also:

`Object.equals(java.lang.Object)`, `get(java.lang.Object)`

remove

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

Removes the key (and its corresponding value) from this hashtable. This method does nothing if the key is not in the hashtable.

Parameters:

`key` - the key that needs to be removed.

Returns:

the value to which the key had been mapped in this hashtable, or `null` if the key did not have a mapping.

Since:

JDK1.0

clear

```
public void clear()
```

Clears this hashtable so that it contains no keys.

Since:

JDK1.0

toString

```
public String toString()
```

Returns a rather long string representation of this hashtable.

Overrides:

`toString` in class `Object`

Returns:

a string representation of this hashtable.

Since:

JDK1.0