compact1, compact2, compact3

java.util

Class HashSet<E>

java.lang.Object
  java.util.AbstractCollection<E>
    java.util.AbstractSet<E>
      java.util.HashSet<E>

Type Parameters:

`E - the type of elements maintained by this set`

All Implemented Interfaces:

`Serializable, Cloneable, Iterable<E>, Collection<E>, Set<E>`

Direct Known Subclasses:

`JobStateReasons, LinkedHashSet`

```
public class HashSet<E>
extends AbstractSet<E>
implements Set<E>, Cloneable, Serializable
```

This class implements the `Set` interface, backed by a hash table (actually a `HashMap` instance). It makes no guarantees as to the iteration order of the set; in particular, it does not guarantee that the order will remain constant over time. This class permits the `null` element.

This class offers constant time performance for the basic operations (`add`, `remove`, `contains` and `size`), assuming the hash function disperses the elements properly among the buckets. Iterating over this set requires time proportional to the sum of the `HashSet` instance's size (the number of elements) plus the "capacity" of the backing `HashMap` instance (the number of buckets). Thus, it's very important not to set the initial capacity too high (or the load factor too low) if iteration performance is important.

Note that this implementation is not synchronized. If multiple threads access a hash set concurrently, and at least one of the threads modifies the set, it *must* be synchronized externally. This is typically accomplished by synchronizing on some object that naturally encapsulates the set. If no such object exists, the set should be "wrapped" using the `Collections.synchronizedSet` method. This is best done at creation time, to prevent accidental unsynchronized access to the set:

```
Set s = Collections.synchronizedSet(new HashSet(...));
```

The iterators returned by this class's `iterator` method are *fail-fast*: if the set is modified at any time after the iterator is created, in any way except through the iterator's own `remove` method, the Iterator throws a `ConcurrentModificationException`. Thus, in the face of concurrent

modification, the iterator fails quickly and cleanly, rather than risking arbitrary, non-deterministic behavior at an undetermined time in the future.

Note that the fail-fast behavior of an iterator cannot be guaranteed as it is, generally speaking, impossible to make any hard guarantees in the presence of unsynchronized concurrent modification. Fail-fast iterators throw `ConcurrentModificationException` on a best-effort basis. Therefore, it would be wrong to write a program that depended on this exception for its correctness: *the fail-fast behavior of iterators should be used only to detect bugs.*

This class is a member of the Java Collections Framework.

Since:

1.2

See Also:

`Collection, Set, TreeSet, HashMap, Serialized Form`

Constructor Summary

Constructors

Constructor and Description
`HashSet()` Constructs a new, empty set; the backing `HashMap` instance has default initial capacity (16) and load factor (0.75).
`HashSet(Collection<? extends E> c)` Constructs a new set containing the elements in the specified collection.
`HashSet(int initialCapacity)` Constructs a new, empty set; the backing `HashMap` instance has the specified initial capacity and default load factor (0.75).
`HashSet(int initialCapacity, float loadFactor)` Constructs a new, empty set; the backing `HashMap` instance has the specified initial capacity and the specified load factor.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`boolean`	**`add(E e)`** Adds the specified element to this set if it is not already present.
`void`	**`clear()`** Removes all of the elements from this set.

Object	**clone**() Returns a shallow copy of this HashSet instance: the elements themselves are not cloned.
boolean	**contains**(**Object** o) Returns true if this set contains the specified element.
boolean	**isEmpty**() Returns true if this set contains no elements.
Iterator<E>	**iterator**() Returns an iterator over the elements in this set.
boolean	**remove**(**Object** o) Removes the specified element from this set if it is present.
int	**size**() Returns the number of elements in this set (its cardinality).
Spliterator<E>	**spliterator**() Creates a ***late-binding*** and *fail-fast* **Spliterator** over the elements in this set.

Methods inherited from class java.util.AbstractSet

equals, hashCode, removeAll

Methods inherited from class java.util.AbstractCollection

addAll, containsAll, retainAll, toArray, toArray, toString

Methods inherited from class java.lang.Object

finalize, getClass, notify, notifyAll, wait, wait, wait

Methods inherited from interface java.util.Set

addAll, containsAll, equals, hashCode, removeAll, retainAll, toArray, toArray

Methods inherited from interface java.util.Collection

parallelStream, removeIf, stream

Methods inherited from interface java.lang.Iterable

forEach

Constructor Detail

HashSet

```
public HashSet()
```

Constructs a new, empty set; the backing `HashMap` instance has default initial capacity (16) and load factor (0.75).

HashSet

```
public HashSet(Collection<? extends E> c)
```

Constructs a new set containing the elements in the specified collection. The `HashMap` is created with default load factor (0.75) and an initial capacity sufficient to contain the elements in the specified collection.

Parameters:

`c - the collection whose elements are to be placed into this set`

Throws:

`NullPointerException - if the specified collection is null`

HashSet

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

Constructs a new, empty set; the backing `HashMap` instance has the specified initial capacity and the specified load factor.

Parameters:

`initialCapacity - the initial capacity of the hash map`

`loadFactor - the load factor of the hash map`

Throws:

`IllegalArgumentException - if the initial capacity is less than zero, or if the load factor is nonpositive`

HashSet

```
public HashSet(int initialCapacity)
```

Constructs a new, empty set; the backing `HashMap` instance has the specified initial capacity and default load factor (0.75).

Parameters:

`initialCapacity - the initial capacity of the hash table`

Throws:

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

Method Detail

iterator

```
public Iterator<E> iterator()
```

Returns an iterator over the elements in this set. The elements are returned in no particular order.

Specified by:

`iterator` in interface `Iterable<E>`

Specified by:

`iterator` in interface `Collection<E>`

Specified by:

`iterator` in interface `Set<E>`

Specified by:

`iterator` in class `AbstractCollection<E>`

Returns:

an Iterator over the elements in this set

See Also:

`ConcurrentModificationException`

size

```
public int size()
```

Returns the number of elements in this set (its cardinality).

Specified by:

`size` in interface `Collection<E>`

Specified by:

`size` in interface `Set<E>`

Specified by:

`size` in class `AbstractCollection<E>`

Returns:

the number of elements in this set (its cardinality)

isEmpty

```
public boolean isEmpty()
```

Returns `true` if this set contains no elements.

Specified by:

`isEmpty` in interface `Collection<E>`

Specified by:

`isEmpty` in interface `Set<E>`

Overrides:

`isEmpty` in class `AbstractCollection<E>`

Returns:

`true` if this set contains no elements

contains

```
public boolean contains(Object o)
```

Returns `true` if this set contains the specified element. More formally, returns `true` if and only if this set contains an element `e` such that `(o==null ? e==null : o.equals(e))`.

Specified by:

`contains` in interface `Collection<E>`

Specified by:

`contains` in interface `Set<E>`

Overrides:

`contains` in class `AbstractCollection<E>`

Parameters:

`o` - element whose presence in this set is to be tested

Returns:

`true` if this set contains the specified element

add

```
public boolean add(E e)
```

Adds the specified element to this set if it is not already present. More formally, adds the specified element `e` to this set if this set contains no element `e2` such that `(e==null ? e2==null : e.equals(e2))`. If this set already contains the element, the call leaves the set unchanged and returns `false`.

Specified by:

`add` in interface `Collection<E>`

Specified by:

`add` in interface `Set<E>`

Overrides:

`add` in class `AbstractCollection<E>`

Parameters:

`e` - element to be added to this set

Returns:

`true` if this set did not already contain the specified element

remove

```
public boolean remove(Object o)
```

Removes the specified element from this set if it is present. More formally, removes an element `e` such that `(o==null ? e==null : o.equals(e))`, if this set contains such an element. Returns `true` if this set contained the element (or equivalently, if this set changed as a result of the call). (This set will not contain the element once the call returns.)

Specified by:

`remove` in interface `Collection<E>`

Specified by:

`remove` in interface `Set<E>`

Overrides:

`remove` in class `AbstractCollection<E>`

Parameters:

`o` - object to be removed from this set, if present

Returns:

`true` if the set contained the specified element

clear

```
public void clear()
```

Removes all of the elements from this set. The set will be empty after this call returns.

Specified by:

`clear` in interface `Collection<E>`

Specified by:

`clear` in interface `Set<E>`

Overrides:

`clear` in class `AbstractCollection<E>`

clone

```
public Object clone()
```

Returns a shallow copy of this `HashSet` instance: the elements themselves are not cloned.

Overrides:

`clone` in class `Object`

Returns:

a shallow copy of this set

See Also:

`Cloneable`

spliterator

```
public Spliterator<E> spliterator()
```

Creates a *late-binding* and *fail-fast* `Spliterator` over the elements in this set.

The `Spliterator` reports `Spliterator.SIZED` and `Spliterator.DISTINCT`. Overriding implementations should document the reporting of additional characteristic values.

Specified by:

`spliterator` in interface `Iterable<E>`

Specified by:

`spliterator` in interface `Collection<E>`

Specified by:

`spliterator` in interface `Set<E>`

Returns:

a Spliterator over the elements in this set

Since:

1.8