**Methods in HashSet**

Here, **E** is theType of elements store in HashSet

| **METHOD** | **DESCRIPTION** |
| --- | --- |
| [add(E e)](https://www.geeksforgeeks.org/hashset-add-method-in-java/) | Used to add the specified element if it is not present, if it is present then return false. |
| [clear()](https://www.geeksforgeeks.org/hashset-clear-method-in-java/) | Used to remove all the elements from set. |
| [contains(Object o)](https://www.geeksforgeeks.org/hashset-contains-method-in-java/) | Used to return true if an element is present in set. |
| [remove(Object o)](https://www.geeksforgeeks.org/hashset-remove-method-in-java/) | Used to remove the element if it is present in set. |
| [iterator()](https://www.geeksforgeeks.org/hashset-iterator-method-in-java/) | Used to return an iterator over the element in the set. |
| [isEmpty()](https://www.geeksforgeeks.org/hashset-isempty-method-in-java/) | Used to check whether the set is empty or not. Returns true for empty and false for a non-empty condition for set. |
| [size()](https://www.geeksforgeeks.org/hashset-size-method-in-java/) | Used to return the size of the set. |
| [clone()](https://www.geeksforgeeks.org/hashset-clone-method-in-java/) | Used to create a shallow copy of the set. |

**Methods inherited from class java.util.AbstractSet**

| **METHOD** | **DESCRIPTION** |
| --- | --- |
| [equals()](https://www.geeksforgeeks.org/abstractset-equals-method-in-java-with-examples/?ref=rp) | Used to verify the equality of an Object with a HashSet and compare them. The list returns true only if both HashSet contains same elements, irrespective of order. |
| [hashcode()](https://www.geeksforgeeks.org/abstractset-hashcode-method-in-java-with-examples/?ref=rp) | Returns the hash code value for this set. |
| [removeAll(collection)](https://www.geeksforgeeks.org/abstractset-removeall-method-in-java-with-examples/?ref=rp) | This method is used to remove all the elements from the collection which are present in the set.  This method returns true if this set changed as a result of the call. |

**Methods inherited from class java.util.AbstractCollection**

| **METHOD** | **DESCRIPTION** |
| --- | --- |
| [addAll(collection)](https://www.geeksforgeeks.org/abstractcollection-addall-method-in-java-with-examples/) | This method is used to append all of the elements from the mentioned collection to the existing set.  The elements are added randomly without following any specific order. |
| [containsAll(collection)](https://www.geeksforgeeks.org/abstractcollection-containsall-method-in-java-with-examples/) | This method is used to check whether the set contains all the elements present in the given collection or not.  This method returns true if the set contains all the elements and returns false if any of the elements are missing. |
| [retainAll(collection)](https://www.geeksforgeeks.org/abstractcollection-retainall-method-in-java-with-examples/) | This method is used to retain all the elements from the set which are mentioned in the given collection.  This method returns true if this set changed as a result of the call. |
| [toArray()](https://www.geeksforgeeks.org/abstractcollection-toarray-method-in-java-with-examples/) | This method is used to form an array of the same elements as that of the Set. |
| [toString()](https://www.geeksforgeeks.org/abstractcollection-tostring-method-in-java-with-examples/) | The toString() method of Java HashSet is used to return a string representation of the elements of the HashSet Collection. |

**Methods declared in interface java.util.Collection**

| **METHOD** | **DESCRIPTION** |
| --- | --- |
| parallelStream() | Returns a possibly parallel Stream with this collection as its source. |
| removeIf​(Predicate<? super E> filter) | Removes all of the elements of this collection that satisfy the given predicate. |
| stream() | Returns a sequential Stream with this collection as its source. |
| toArray​(IntFunction<T[]> generator) | Returns an array containing all of the elements in this collection, using the provided generator function to allocate the returned array. |

**Methods declared in interface java.lang.Iterable**

| **METHOD** | **DESCRIPTION** |
| --- | --- |
| [forEach​(Consumer<? super T> action)](https://www.geeksforgeeks.org/iterable-foreach-method-in-java-with-examples/) | Performs the given action for each element of the Iterable until all elements have been processed or the action throws an exception. |

**Methods declared in interface java.util.Set**

| **METHOD** | **DESCRIPTION** |
| --- | --- |
| [addAll​(Collection<? extends E> c)](https://www.geeksforgeeks.org/set-addall-method-in-java-with-examples/) | Adds all of the elements in the specified collection to this set if they’re not already present (optional operation). |
| [containsAll​(Collection<?> c)](https://www.geeksforgeeks.org/set-containsall-method-in-java-with-examples/) | Returns true if this set contains all of the elements of the specified collection. |
| [equals​(Object o)](https://www.geeksforgeeks.org/set-equals-method-in-java-with-examples/) | Compares the specified object with this set for equality. |
| [hashCode()](https://www.geeksforgeeks.org/set-hashcode-method-in-java-with-examples/) | Returns the hash code value for this set. |
| [removeAll​(Collection<?> c)](https://www.geeksforgeeks.org/set-removeall-method-in-java-with-examples/) | Removes from this set all of its elements that are contained in the specified collection (optional operation). |
| [retainAll​(Collection<?> c)](https://www.geeksforgeeks.org/set-retainall-method-in-java-with-example/) | Retains only the elements in this set that are contained in the specified collection (optional operation). |
| [toArray()](https://www.geeksforgeeks.org/set-toarray-method-in-java-with-example/) | Returns an array containing all of the elements in this set. |
| toArray​(T[] a) | Returns an array containing all of the elements in this set; the runtime type of the returned array is that of the specified array. |

**HashSet vs HashMap**

| **BASIS** | **HashSet** | **HashMap** |
| --- | --- | --- |
| *Implementation* | HashSet implements Set interface. | HashMap implements Map interface. |
| *Duplicates* | HashSet doesn’t allow duplicate values. | HashMap store key, value pairs and it does not allow duplicate keys. If key is duplicate then the old key is replaced with the new value. |
| *Number of objects during storing objects* | HashSet requires only one object add(Object o). | HashMap requires two objects put(K key, V Value) to add an element to the HashMap object. |
| *Dummy value* | HashSet internally uses HashMap to add elements. In HashSet, the argument passed in add(Object) method serves as key K. Java internally associates dummy value for each value passed in add(Object) method. | HashMap does not have any concept of dummy value. |
| *Storing or Adding mechanism* | HashSet internally uses the HashMap object to store or add the objects. | HashMap internally uses hashing to store or add objects |
| *Faster* | HashSet is slower than HashMap. | HashMap is faster than HashSet. |
| *Insertion* | HashSet uses the add() method for add or storing data. | HashMap uses the put() method for storing data. |
| *Example* | HashSet is a set, e.g. {1, 2, 3, 4, 5, 6, 7}. | HashMap is a key -> value pair(key to value) map, e.g. {a -> 1, b -> 2, c -> 2, d -> 1}. |

**HashSet vs TreeSet**

| **BASIS** | **HashSet** | **TreeSet** |
| --- | --- | --- |
| *Speed and internal implementation* | For operations like search, insert and delete. It takes constant time for these operations on average. HashSet is faster than TreeSet. HashSet is Implemented using a hash table. | TreeSet takes O(Log n) for search, insert and delete which is higher than HashSet. But TreeSet keeps sorted data. Also, it supports operations like higher() (Returns least higher element), floor(), ceiling(), etc. These operations are also O(Log n) in TreeSet and not supported in HashSet. TreeSet is implemented using a Self Balancing Binary Search Tree (Red-Black Tree). TreeSet is backed by TreeMap in Java. |
| *Ordering* | Elements in HashSet are not ordered. | TreeSet maintains objects in Sorted order defined by either Comparable or Comparator method in Java. TreeSet elements are sorted in ascending order by default. It offers several methods to deal with the ordered set like first(), last(), headSet(), tailSet(), etc. |
| *Null Object* | HashSet allows the null object. | TreeSet doesn’t allow null Object and throw NullPointerException, Why, because TreeSet uses compareTo() method to compare keys and compareTo() will throw java.lang.NullPointerException. |
| *Comparison* | HashSet uses equals() method to compare two objects in Set and for detecting duplicates. | TreeSet uses compareTo() method for same purpose. If equals() and compareTo() are not consistent, i.e. for two equal object equals should return true while compareTo() should return zero, then it will break the contract of the Set interface and will allow duplicates in Set implementations like TreeSet |