**HashSet Class in Java with example**

By Chaitanya Singh | Filed Under: [Java Collections](https://beginnersbook.com/category/java-collections/)

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 is not synchronized. However it can be synchronized explicitly like this: Set s = Collections.synchronizedSet(new HashSet(...));

**Points to Note about HashSet:**

1. HashSet doesn’t maintain any order, the elements would be returned in any random order.
2. HashSet doesn’t allow duplicates. If you try to add a duplicate element in HashSet, the old value would be overwritten.
3. HashSet allows null values however if you insert more than one nulls it would still return only one null value.
4. HashSet is non-synchronized.
5. The iterator returned by this class is fail-fast which means iterator would throw ConcurrentModificationException if HashSet has been modified after creation of iterator, by any means except iterator’s own remove method.

**HashSet Example**

import java.util.HashSet;

public class HashSetExample {

public static void main(String args[]) {

// HashSet declaration

HashSet<String> hset =

new HashSet<String>();

// Adding elements to the HashSet

hset.add("Apple");

hset.add("Mango");

hset.add("Grapes");

hset.add("Orange");

hset.add("Fig");

//Addition of duplicate elements

hset.add("Apple");

hset.add("Mango");

//Addition of null values

hset.add(null);

hset.add(null);

//Displaying HashSet elements

System.out.println(hset);

}

}

Output:

[null, Mango, Grapes, Apple, Orange, Fig]

As you can see there all the duplicate values are not present in the output including the duplicate null value.

**HashSet tutorials**

* [Delete all elements from HashSet](https://beginnersbook.com/2014/08/delete-all-the-elements-from-hashset/)
* [How to iterate through a HashSet](https://beginnersbook.com/2014/08/how-to-iterate-over-a-sethashset/)
* [Convert a HashSet to an array](https://beginnersbook.com/2014/08/converting-a-hashset-to-an-array/)
* [Convert a HashSet to a TreeSet](https://beginnersbook.com/2014/08/how-to-convert-a-hashset-to-a-treeset/)
* [Convert HashSet to a List/ArrayList](https://beginnersbook.com/2014/08/convert-hashset-to-a-list-arraylist/)
* [HashSet vs HashMap](https://beginnersbook.com/2014/08/hashset-vs-hashmap-java/)
* [HashSet vs TreeSet](https://beginnersbook.com/2014/08/difference-between-hashset-and-treeset/)

**HashSet Methods:**

1. **boolean add(Element  e)**: It adds the element e to the list.
2. **void clear()**: It removes all the elements from the list.
3. **Object clone()**: This method returns a shallow copy of the HashSet.
4. **boolean contains(Object o)**: It checks whether the specified Object o is present in the list or not. If the object has been found it returns true else false.
5. **boolean isEmpty()**: Returns true if there is no element present in the Set.
6. **int size()**: It gives the number of elements of a Set.
7. **boolean(Object o)**: It removes the specified Object o from the Set.

# Delete all the elements from HashSet

By Chaitanya Singh | Filed Under: [Java.util package](https://beginnersbook.com/category/java-util-package/)

Here we are gonna see how to remove all the elements of HashSet in one go. We can do so by calling clear() method of HashSet class.

#### Example

import java.util.HashSet;

class EmptyHashSetExample{

public static void main(String[] args) {

// Create a HashSet

HashSet<String> hset = new HashSet<String>();

//add elements to HashSet

hset.add("Element1");

hset.add("Element2");

hset.add("Element3");

hset.add("Element4");

hset.add("Element5");

// Display HashSet elements

System.out.println("Before: HashSet contains: "+ hset);

/\* public void clear(): It removes all the elements

\* from HashSet. The set becomes empty after this

\* method gets called.

\*/

hset.clear();

// Display HashSet content again

System.out.println("After: HashSet contains: "+ hset);

}

}

**Output:**

Before: HashSet contains: [Element1, Element2, Element3, Element4, Element5]

After: HashSet contains: []

**Java – How to Sort a HashSet?**

By Chaitanya Singh | Filed Under: [Java.util package](https://beginnersbook.com/category/java-util-package/)

As we know [HashSet](https://beginnersbook.com/2013/12/hashset-class-in-java-with-example/) doesn’t sort elements, in fact it displays them in random order. While dealing with HashSet we may come across a situation where we need to sort it explicitly. we need to write a logic to sort them when required. In this article we are going to see an example where we are sorting a HashSet using two different methods.

/\* Program to Sort a HashSet using two different

 \* methods.

 \* Method 1:By using List interface

 \* Method 2:By using TreeSet

 \*/

import java.util.Collections;

import java.util.ArrayList;

import java.util.List;

import java.util.HashSet;

import java.util.TreeSet;

public class SortingHashSetDemo {

   public static void main(String args[]) {

       HashSet<String> fruits = new HashSet<String>();

       fruits.add("Orange");

       fruits.add("Apple");

       fruits.add("Banana");

       fruits.add("Guava");

       fruits.add("Pear");

       System.out.println("HashSet elements before sorting: "+fruits);

       // Method 1: Sorting HashSet using List interface

       List<String> fruitList = new ArrayList<String>(fruits);

       Collections.sort(fruitList);

       // Displaying list

       System.out.println("HashSet elements after sorting: "+fruitList);

       // Method 2: Sorting using TreeSet

       TreeSet<String> tset = new TreeSet<String>(fruits);

       System.out.println("HashSet elements after using TreeSet: "+tset);

   }

}

**Output:**

HashSet elements before sorting: [Pear, Guava, Apple, Banana, Orange]

HashSet elements after sorting: [Apple, Banana, Guava, Orange, Pear]

HashSet elements after using TreeSet: [Apple, Banana, Guava, Orange, Pear]

In method 1, we created a list and get the HashSet elements copied to it, then we have used the Collections.sort() method that sorts the elements of a list.  
In method 2, we created a [TreeSet](https://beginnersbook.com/2013/12/treeset-class-in-java-with-example/) and got the HashSet elements in TreeSet by providing the HashSet reference while creating TreeSet. As we are aware that TreeSet sorts its elements in ascending order, we ultimately got our HashSet elementns sorted out.  
As you can see, we got the HashSet sorted out using above mentioned methods. The TreeSet method is preferred one because TreeSet is created for this purpose only.

**Enjoyed this post? Try these related posts**

1. [Adding element to front of LinkedList in Java](https://beginnersbook.com/2014/08/adding-element-to-front-of-linkedlist-in-java/)
2. [How to convert a HashSet to a TreeSet](https://beginnersbook.com/2014/08/how-to-convert-a-hashset-to-a-treeset/)
3. [Adding an element to LinkedList using add(E e) method – Java](https://beginnersbook.com/2014/08/adding-an-element-to-linkedlist-using-adde-e-method-java/)
4. [How to Iterate over a Set/HashSet](https://beginnersbook.com/2014/08/how-to-iterate-over-a-sethashset/)
5. [Difference between HashSet and TreeSet](https://beginnersbook.com/2014/08/difference-between-hashset-and-treeset/)
6. [Clone a HashMap in Java](https://beginnersbook.com/2014/08/clone-a-hashmap-in-java/)

# How to convert a HashSet to a TreeSet

By Chaitanya Singh | Filed Under: [Java.util package](https://beginnersbook.com/category/java-util-package/)

#### Description

Program to convert a HashSet to a TreeSet

#### Program

Here is the complete code for HashSet to TreeSet conversion. We have a HashSet of Strings and we are creating a TreeSet of strings by copying all the elements of HashSet to TreeSet.

import java.util.HashSet;

import java.util.TreeSet;

import java.util.Set;

class ConvertHashSettoTreeSet{

public static void main(String[] args) {

// Create a HashSet

HashSet<String> hset = new HashSet<String>();

//add elements to HashSet

hset.add("Element1");

hset.add("Element2");

hset.add("Element3");

hset.add("Element4");

// Displaying HashSet elements

System.out.println("HashSet contains: "+ hset);

// Creating a TreeSet of HashSet elements

Set<String> tset = new TreeSet<String>(hset);

// Displaying TreeSet elements

System.out.println("TreeSet contains: ");

for(String temp : tset){

System.out.println(temp);

}

}

}

**Output:**

HashSet contains: [Element1, Element2, Element3, Element4]

TreeSet contains:

Element1

Element2

Element3

Element4