A Set is a Collection that cannot contain duplicate elements. There are three main implementations of Set interface: HashSet, TreeSet, and LinkedHashSet. HashSet, which stores its elements in a hash table, is the best-performing implementation; however it makes no guarantees concerning the order of iteration. TreeSet, which stores its elements in a red-black tree, orders its elements based on their values; it is substantially slower than HashSet. LinkedHashSet, which is implemented as a hash table with a linked list running through it, orders its elements based on the order in which they were inserted into the set (insertion-order).

**HashSet Class in Java with example**

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