# Set Interface

A Set is a collection that cannot contain duplicate elements. It models the mathematical set abstraction.

It does not allow duplicate elements and allow one null value at most.

|  |  |
| --- | --- |
| **Sr.No.** | **Method & Description** |
| 1 | **add( )**  Adds an object to the collection. |
| 2 | **clear( )**  Removes all objects from the collection. |
| 3 | **contains( )**  Returns true if a specified object is an element within the collection. |
| 4 | **isEmpty( )**  Returns true if the collection has no elements. |
| 5 | **iterator( )**  Returns an Iterator object for the collection, which may be used to retrieve an object. |
| 6 | **remove( )**  Removes a specified object from the collection. |
| 7 | **size( )**  Returns the number of elements in the collection. |

There are three classes implementing this interface −

* **HashSet** − Set implementation based on hash table.
* **LinkedHashSet** − HashSet implementation based on linked list.
* **TreeSet** − Set implementation based on trees.

# HashSet:

* Hashset class which is implemented in the collection framework is an inherent implementation of the hash table datastructure.
* The objects that we insert into the hashset does not guarantee to be inserted in the same order.
* The objects are inserted based on their hashcode.
* This class also allows the insertion of NULL elements.

import java.util.\*; class setexam1{

public static void main(String[] args)

{

Set<String> h = new HashSet<String>();

// Adding elements into the HashSet

// using add() h.add("Nidhi");

h.add("Vidhi");

h.add("Aidhi"); h.add(null);

// Adding the duplicate

// element h.add("Nidhi");

// Displaying the HashSet System.out.println(h);

// Removing items from HashSet

// using remove() h.remove("Vidhi");

System.out.println("Set after removing "

+ "Vidhi:" + h);

// Iterating over hash set items System.out.println("Iterating over set:"); Iterator<String> i = h.iterator();

while (i.hasNext()) System.out.println(i.next());

// check items from HashSet

// using contains()

System.out.println("Does the Set contains Aidhi? " + h.contains("Aidhi"));

}

}

# LinkedHashSet:

LinkedHashSet class which is implemented in the collections framework is an ordered version of HashSet that maintains a doubly-linked List across all elements.

When the iteration order is needed to be maintained this class is used. import java.util.\*;

class setlinkl {

public static void main(String[] args)

{

Set<String> lh

= new LinkedHashSet<String>();

// Adding elements into the LinkedHashSet

// using add() lh.add("Nidhi");

lh.add("Vidhi");

lh.add("Aidhi");

// Adding the duplicate

// element lh.add("Nidhi");

// Displaying the LinkedHashSet System.out.println(lh);

// Removing items from LinkedHashSet

// using remove() lh.remove("Vidhi");

System.out.println("Set after removing "

+ "Vidhi:" + lh);

// Iterating over linked hash set items System.out.println("Iterating over set:"); Iterator<String> i = lh.iterator();

while (i.hasNext()) System.out.println(i.next());

}

}

# TreeSet:

* TreeSet class which is implemented in the collections framework an implementation of the SortedSet Interface and SortedSet extends Set Interface.
* It behaves like simple set with the exception that it stores elements in sorted format.
* TreeSet uses tree data structure for storage. Objects are stored in sorted, ascending order.
* But we can iterate in descending order using method TreeSet.descendingIterator().

import java.util.Iterator;

import java.util.TreeSet;

public class Treeeg {

public static void main(String[] args) {

// creating a TreeSet

TreeSet <Integer>t = new TreeSet<Integer>();

// adding in the tree set t.add(1);

t.add(8);

t.add(3);

t.add(12);

// Iterating over tree set items System.out.println("Iterating over set:"); Iterator i = t.iterator();

while (i.hasNext()) System.out.println(i.next());

// create descending iterator Iterator iterator;

iterator = t.descendingIterator();

// displaying the Tree set data

System.out.println("Tree set data in descending order: ");

while (iterator.hasNext()) { System.out.println(iterator.next() + " ");

}

}

}