HashSet

1. Duplicates are not allowed,if we try to add it would not throw any error rather it would return false.
2. Internal DataStructure: Hashtable
3. null insertion is possible.
4. heterogeneous data elements can be added.
5. If our frequent operation is search, then the best choice is HashSet.
6. It implements Serializable,Cloneable, but not random access.

Constructors

HashSet s=new HashSet(); Default initial capacity is 16
Default FillRation/load factor is 0.75

Note: In case of ArrayList, default capacity is 10, after filling the complete capacity then a new ArrayList would be created.
In the case of HashSet, after filling 75% of the ratio only new HashSet will be created.

HashSet s=new HashSet(int initialCapacity);//specified capacity with default fill ration=0.75
HashSet s=new HashSet(int initaliCapacity,float fillRatio)
HashSet s=new HashSet(Collection c);

LinkedHashSet

- It is the child class of "HashSet".
- DataStructure: HashTable + linkedlist
- duplicates : not allowed
- insertion order: preserved
- null allowed : yes

All the constructors and methods which are a part of HashSet will be a part of "LinkedHashSet",but except "insertion order will be preserved".

Difference b/w HashSet and LinkedHashSet

HashSet => underlying data structure is "HasTable"
LinkedHashSet => underlying data structure is a combination of "Hashtable + "linkedlist" .
HashSet=> Duplicates are not allowed and insertion order is not preserved
LinkedHashSet => Duplicates are not allowed,but insertion order is preserved.
HashSet => 1.2V
LinkedHashSet => 1.4v

The 3 Cursors of Java

- If we want to get Objects One by One from the Collection then we should go for Cursors.
- There are 3 Types of Cursors Available in Java.

1. Enumeration
2. Iterator
3. ListIterator

1. Enumeration:

We can Use Enumeration to get Objects One by One from the Collection.
We can Create Enumeration Object by using elements().

public Enumeration elements();
Eg: Enumeration e = v.elements(); //v is Vector Object.

Methods:

1. public boolean hasMoreElements();
2. public Object nextElement();

```
import java.util.*;
public class EnumerationDemo {
 public static void main(String[] args) {
  Vector v = new Vector();
  for(int i=0; i<=10; i++) {
   v.addElement(i);
  }
  System.out.println(v);//[0,1,2,3,4,5,6,7,8,9,10]
  Enumeration e = v.elements();
  while(e.hasMoreElements()) {
   Integer I = (Integer)e.nextElement();
   if(I%2 == 0)
    System.out.println(I);//0 2 4 6 8 10
  }
  System.out.println(v);//[0,1,2,3,4,5,6,7,8,9,10]
 }
}
```

Limitations of Enumeration:

- Enumeration Concept is Applicable Only for Legacy Classes and it is Not a Universal Cursor.
- By using Enumeration we can Perform Read Operation and we can't Perform Remove Operation.

To Overcome Above Limitations we should go for Iterator.

2. Iterator

- We can use an Iterator to get Objects One by One from Collection.
- We can Apply Iterator Concept for any Collection Object. Hence it is a Universal Cursor.
- By using Iterator we can Able to Perform Both Read and Remove Operations.
- We can Create Iterator Object by using the iterator() of Collection Interface.
  public Iterator iterator();
  Eg:Iterator itr = c.iterator(); //c Means any Collection Object.

Methods:

1) public booleanhasNext()
2) public Object next()
3) public void remove()

Limitations:

- By using Enumeration and Iterator we can Move Only towards Forward Direction and we can't Move Backward Direction. That is, these are Single Direction Cursors but NotBiDirection.
- By using Iterator we can Perform Only Read and Remove Operations and we can't Perform Addition of New Objects and Replacing Existing Objects.

To Overcome these Limitations we should go for ListIterator.

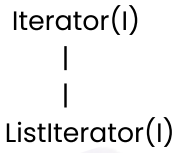
3. ListIterator:

- ListIterator is the Child Interface of Iterator.
- By using ListIterator we can Move Either to the Forward Direction OR to the Backward Direction. That is it is a Bi-Directional Cursor.
- By using ListIterator we can Able to Perform Addition of New Objects andReplacing existing Objects. In Addition to Read and Remove Operations.
- We can Create ListIterator Object by using listIterator().

public ListIteratorlistIterator();
Eg:ListIteratorItr = l.listIterator(); //l is Any List Object

Methods:

- ListIteratoris the Child Interface of Iterator and Hence All Iterator Methods by Default Available to the ListIterator.



ListIteratorDefines the following 9 Methods.

```
public boolean hasNext()
public Object next()
public int nextIndex()

public boolean hasPrevious()
public Object previous()
public int previousIndex()

public void remove()
public void set(Object new)
public void add(Object new)
```

Legacy classes

- Legacy classes refers to the older classes that were included in the early versions of Java and have since been replaced by newer, more efficient classes. One such class is Enumeration, which is a legacy interface that was used to traverse collections before the introduction of the Iterator interface.

The Legacy classes are Dictionary, Hashtable, Properties, Stack, and Vector. The Legacy interface is the Enumeration interface.