**Collection Framework :-**

**Framework**:-`The readymade architecture. It is collection of class & interface.

**Collection**:-It is the single unit of group of object and provide many interface and classes.

Collection

List

Vector

ArrayList

LinkedList

Stack

Set

Hash Set

TreeSet

Sorted Set

LinkedHashSet

Queue

Priority Queue

Blocking Queue

Priority Blocking Queue

Linked Blocking Queue

**Methods of Collection interface:**

|  |  |  |
| --- | --- | --- |
| **No.** | **Method** | **Description** |
| 1. | add(Object element) | It is used to insert the element in this collection. |
| 2. | booleanaddAll(Collection c) | It is used to insert the specified collection elements in the invoking collection. |
| 3 | boolean remove(Object element) | It is used to delete the element from this collection. |
| 4 | booleanremoveAll(Collection c) | It is used to delete all the elements of specified collection from the invoking collection. |
| 5 | booleanretainAll(Collection c) | It is used to delete all the elements of invoking collection except the specified collection. |
| 6 | int size() | Show the total number of elements in the collection. |
| 7 | void clear() | Removes the total no of element from the collection. |
| 8 | boolean contains(Object element) | It is used to search the element. |
| 9 | booleancontainsAll(Collection c) | It is used to search the specified collection in this collection. |
| 10 | Iterator iterator() | This is returns an iterator. |
| 11 | Object[] toArray() | Converts collection into array. |
| 12 | booleanisEmpty() | Checks if collection is empty. |
| 13 | boolean equals(Object element) | Compare two collection. |
| 14 | inthashCode() | This is returns thehashcode number for collection. |

**Iterator interface:**

There are only three methods of the Iterator interface. That are follows :

1. **booleanhasNext()** : it returns true if iterator has more elements.
2. **object next()** : it returns the element and moves the cursor pointer to the next element.
3. **void remove()** : it removes the last elements returned by the iterator. It is rarely used.

**List** :-

* Its child interface of collection.
* Its want represent a group of individual object where duplicate are allow and Insertion order is preserved.

**List interface define the following methods :-**

|  |  |  |
| --- | --- | --- |
| **No.** | **Method** | **Description** |
|  | boolean add( intindex,object o) | Add the object. |
|  | booleanaddAll( intindex,Collection c) | Add all the object in List |
|  | object remove(int index) | Remove the object in List |
|  | object get(int index) | Get the object by index value. |
|  | object set(intindex,object new) | Replace object in list |
|  | intindexof(object o) | Find the index of Object. |
|  | intlastIndexOf(object o) | Find last index of Object. |
|  | ListIteratorlistIterator() | This use for ListIterator. |

**List contain 4 Classes : -**

1. **ArrayList(c)**
2. **LinkedList**
3. **VectorList**
4. **Stack**

**1. ArrayList** :-

* ArrayList maintains the insertion order.
* It contains duplicate elements .
* It uses the iterator interface to travers the a list element.
* NULL insertion is possible.
* It is non-Synchronized & not Thread Safe.
* It increase the array current size.

**Constructor** :-

1. ArrayList Al = new ArrayList(); // Create empty Arraylist.
2. ArrayList l = new ArrayList(intintialCapcity); // Specified Arraylist size.
3. ArrayList l = new ArrayList(Collection c); // Create equivalent ArrayList.

**ArrayListExample :-**

*import java.util.\*;*

*class MyArrayList*

*{*

*public static void main(String args[])*

*{*

*ArrayList list=new ArrayList();//Creating arraylist*

*list.add("Ravi");//Adding object in arraylist*

*list.add("Vijay");*

*list.add("Ravi");*

*list.add("Ajay");*

*//Traversing list through Iterator*

*Iterator itr=list.iterator();*

*while(itr.hasNext())*

*{*

*System.out.println(itr.next());*

*}*

*}*

*}*

**Output** : -

*Ravi*

*Vijay*

*Ravi*

*Ajay*

**2. Linked List :-**

* Linked List uses list as well as deque.
* LinkList internally uses douplylinkedlist structure.
* In case of linkedlist there is no more shifting operation.
* But LinkedList is basically use for manipulation purpose(object).

**Constructor** :-

1. LinkedListll = new LinkedList(); // Create empty Linkedlist.
2. LinkedList l = new LinkedList(Collection c); // Create equivalent LinkedList.

**LinkedListExample :-**

*import java.util.\*;*

*class MyLinkedList*

*{*

*public static void main(String args[])*

*{*

*LinkedList al=****new****LinkedList ();*

*al.add("Ravi");*

*al.add("Vijay");*

*al.add("Ravi");*

*al.add("Ajay");*

*//Traversing list through Iterator*

*Iterator itr=list.iterator();*

*while(itr.hasNext())*

*{*

*System.out.println(itr.next());*

*}*

*}*

*}*

**Output**: -

*Ravi*

*Vijay*

*Ravi*

*Ajay*

**3.Vector :-**

* It is Synchronized & Thread Safe.
* Vector also Contain duplicate value.
* It increase double of array current size.
* Vector also contains duplicate element.
* Vector uses the Enumeration interface to travers the a Vector element.

**Constructor :-**

1. Vector v = new Vector(); // Create empty Vector.
2. Vector v = new Vector(intintialCapcity); // Specified Vector size.
3. Vector v = new Vector(Collection c); // Create equivalent Vector.

**VectorExample :-**

*import java.util.\*;*

*class MyVector*

*{*

*public static void main(String args[])*

*{*

*Vector al=****new****Vector ();*

*al.addElement("Ravi");*

*al.addElement("Vijay");*

*al.addElement("Ravi");*

*al.addElement("Ajay");*

*//Traversing list through Iterator*

*Enumeration vEnum = v.elements();*

*System.out.println("\nElements in vector:");*

*while(vEnum.hasMoreElements())*

*{*

*System.out.print(vEnum.nextElement() + " ");*

*}*

*}*

*}*

**Output**: -

*Ravi*

*Vijay*

*Ravi*

*Ajay*

**Set**:-The Set Contain single object . The Set Interface. A Set is a Collection that cannot contain duplicate elements. It models the mathematical set abstraction.

**Set contain 4 Classes : -**

1. **HashSet**
2. **LinkedHashSet**
3. **SortedSet**
4. **TreeSet**

**1.HashSet :-**

* HashSet doesn`t allow duplicate value.
* HashSetdoesn`t maintains the insertion order.

**Constructor :-**

1. HashSeths = new HashSet(); // Create empty HashSet.
2. HashSeths = new HashSet(intintialCapcity); // Specified HashSet size.
3. HashSeths = new HashSet(Collection c); // Create equivalent HashSet.

**HashSetExample :-**

*import java.util.\*;*

*class MyHashSet*

*{*

*public static void main(String args[])*

*{*

*HashSet<String> set=****new****HashSet<String>();*

*set.add("Ravi");*

*set.add("Vijay");*

*set.add("Ravi");*

*set.add("Ajay");*

*//Traversing elements*

*Iterator<String> itr=set.iterator();*

***while****(itr.hasNext())*

*{*

*System.out.println(itr.next());*

*}*

*}*

*}*

**Output**: -

*Ajay*

*Vijay*

*Ravi*

**2.LinkedHashSet** :-

The LinkedHashSet class extends HashSet class which implements Set interface. The Set interface inherits Collection and Iterable interfaces in hierarchical order.

* HashSet doesn`t allow duplicate value.
* HashSet does maintains the insertion order.

**Constructor :-**

1. LinkedHashSeths = new LinkedHashSet(); // Create empty LinkedHashSet.
2. LinkedHashSeths = new LinkedHashSet(intintialCapcity); // Specified LinkedHashSet size.
3. LinkedHashSeths = new LinkedHashSet(Collection c);// Create equivalent LinkedHashSet.

**LinkedHashSetExample :-**

*import java.util.\*;*

*class MyLinkedHashSet*

*{*

*public static void main(String args[])*

*{*

*LinkedHashSet<String> set=****new****LinkedHashSet<String>();*

*set.add("Ravi");*

*set.add("Vijay");*

*set.add("Ravi");*

*set.add("Ajay");*

*//Traversing elements*

*Iterator<String> itr=set.iterator();*

*while(itr.hasNext())*

*{*

*System.out.println(itr.next());*

*}*

*}*

*}*

**Output**: -

*Ravi*

*Ajay*

*Vijay*

**3. SortedSet :-**

* It`s Child Interface of Set.
* If we want to represent a group of individual object ascending to some Sorting Order.

**SortedSetExample :-**

*importjava.util.\*;*

*classSortedSetTest*

*{*

*public static void main(String[] args)*

*{*

*// Create the sorted set*

*SortedSet set = new SortedSet();*

*// Add elements to the set*

*set.add("b");*

*set.add("c");*

*set.add("a");*

*// Iterating over the elements in the set*

*Iterator it = set.iterator();*

*while (it.hasNext())*

*{*

*// Get element*

*Object element = it.next();*

*System.out.println(element.toString());*

*}*

*}*

*}*

**Output :-**

*a*

*b*

*c*

**4.TreeSet**:-

* TreeSet implements set interface and navigable set interface.
* TreeSet doesn`t allow duplicate value.
* TreeSet doesn`t allow NULL values if you pass NULL into Treeset it generate Null Pointer Exception.
* TreeSet elements are stored in nature by default.

**Constructor :-**

1. TreeSetts = new TreeSet (); // Create empty TreeSet.
2. TreeSetts = new TreeSet (intintialCapcity); // Specified TreeSet size.
3. TreeSetts = new TreeSet (Collection c); // Create equivalent TreeSet.
4. TreeSetts = new TreeSet (SortedSet c); // Create equivalent TreeSet.

**TreeSetExample :-**

*importjava.util.\*;*

*public class SortedSetTest*

*{*

*public static void main(String[] args)*

*{*

*// Create the sorted set*

*TreeSet set = new TreeSet();*

*// Add elements to the set*

*set.add("b");*

*set.add("c");*

*set.add("a");*

*// Iterating over the elements in the set*

*Iterator it = set.iterator();*

*while (it.hasNext())*

*{*

*// Get element*

*Object element = it.next();*

*System.out.println(element.toString());*

*}*

*}*

*}*

**Output :-**

*a*

*b*

*c*