Linked lists in Java:

LinkedList implements the Collection interface. It can store the duplicate elements. It maintains the insertion order.

LinkedList<type>linkedlist=new LinkedList();

Here, type indicates the type of a linked list for example

Integer type linked list

LinkedList<Integer> linkedList = new LinkedList();

String type linked list

LinkedList<String> linkedList = new LinkedList();

Methods of Java LinkedList

LinkedList provides various methods that allow us to perform different operations in linked lists. Four commonly used LinkedList Operators

1.Add elements—add()
2.Access elements---get(int index)
3.Change elements—set(int index,Object element)
4.Remove elements---remove(int index)

Other Methods of LinkedList

Methods	Description
contains()	checks if the LinkedList contains the element
indexOf()	returns the index of the first occurrence of the element
lastIndexOf()	returns the index of the last occurrence of the element
clear()	removes all the elements of the LinkedList
iterator()	returns an iterator to iterate over LinkedList

Linked list

Example:

```
public class Demo{
public static void main(String args[]){
LinkedList<String> al=new LinkedList();
al.add("Ravi");
al.add("Vijay");
al.add("Ravi");
al.add("Ajay");
Iterator<String> itr=al.iterator();
while(itr.hasNext()){
                System.out.println(itr.next());
            }
}
 }
```

When to chose ArrayList and LinkedList??

- It is better to use ArrayList when searching is more frequent operation i.e storing and fetching becomes easy in ArrayList then add and remove operation.
- LinkedList is used when insertion/deletion i.e manipulation is required.
- In short, ArrayList is better to access data whereas LinkedList is better to manipulate data.

Set Interface

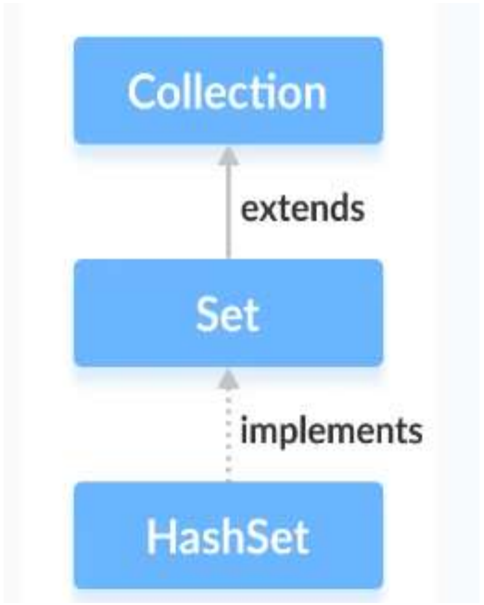
- Set Interface in Java is present in java.util package.
- It extends the Collection interface. It represents the unordered set of elements which doesn't allow us to store the duplicate items.
- Set does not have indexing. Therefore, we cannot access, insert or remove based on index.
- We can access the elements of set only by using
- ✓ iterator()
- ✓ for each loop
- Set is implemented by HashSet, LinkedHashSet, and TreeSet.

Set can be instantiated as:

- Set<data-type> s1 = **new** HashSet<data-type>();
- Set<data-type> s2 = **new** LinkedHashSet<data-type>();
- Set<data-type> s3 = **new** TreeSet<data-type>();

Characteristics of HashSet

- It is a concrete implementing class of set interface.
- It has all the methods inherited from collection interface.
- It is unordered collection of objects.
- We cannot have duplicate objects.
- No indexing possible in HashSet.



1.Insert Elements to HashSet

•add() - inserts the specified element to the set
•addAll() - inserts all the elements of the specified collection to the set

2.Access HashSet Elements

- To access the elements of a hash set, we can use the iterator() method. In order to use this method, we must import the java.util.Iterator package.

3.Remove HashSet Elements

•remove() - removes the specified element from the set
•removeAll() - removes all the elements from the set

Other Methods Of HashSet

Method	**Description**
clone()	Creates a copy of the HashSet
contains()	Searches the HashSet for the specified element and returns a boolean result
isEmpty()	Checks if the HashSet is empty
size()	Returns the size of the HashSet
clear()	Removes all the elements from the HashSet

Hashset

Example:

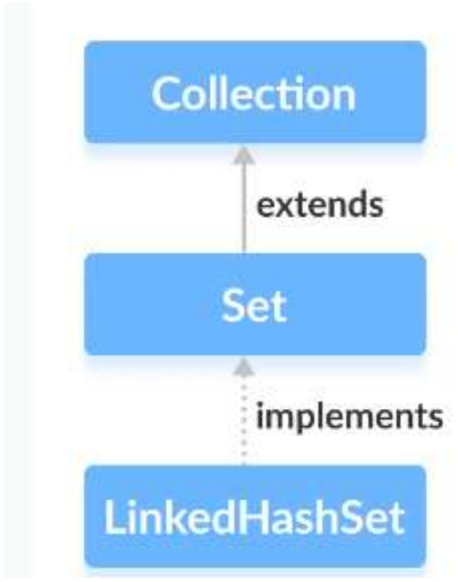
```
import java.util.*;
public class TestJavaCollection7{
public static void main(String args[]){
//Creating HashSet and adding elements
HashSet set=new HashSet();
set.add(“Karthik");
set.add(“Alok");
set.add("Ravi");
set.add("Ajay");
//Traversing elements
Iterator itr=set.iterator();
while(itr.hasNext()){
System.out.println(itr.next());
}  }}
```

Why HashSet?

- Note: In Java, HashSet is commonly used if we have to access elements randomly. It is because elements in a hash table are accessed using hash codes.
- The hashCode of an element is a unique identity that helps to identify the element in a hash table.
- HashSet cannot contain duplicate elements. Hence, each hash set element has a unique hashCode

Characteristics of LinkedHashSet

- It is implementing class of set interface, it is defined in java.util package.
- Insertion order is maintained.
- Duplicate elements are not allowed.
- No indexing
- Only one null value is allowed.
- It is a combination of list and HashSet



1.Insert Elements to LinkedHashSet

•add() - inserts the specified element to the linked hash set

•addAll() - inserts all the elements of the specified collection to

2.Access LinkedHashSet Elements

- To access the elements of a linked hash set, we can use the iterator() method. In order to use this method, we must import the java.util.Iterator package.

Note:

•hasNext() returns true if there is a next element in the linked hash set

•next() returns the next element in the linked hash set

3.Remove Elements from HashSet

•remove() - removes the specified element from the linked hash set

•removeAll() - removes all the elements from the linked hash set

Other Methods Of LinkedHashSet

Method	***Description***
clone()	Creates a copy of the LinkedHashSet
contains()	Searches the LinkedHashSet for the specified element and returns a boolean result
isEmpty()	Checks if the LinkedHashSet is empty
size()	Returns the size of the LinkedHashSet
clear()	Removes all the elements from the LinkedHashSet

LinkedHashSet

Example:

```
import java.util.*;
public class TestJavaCollection8{
public static void main(String args[]){
LinkedHashSet<String> set=new LinkedHashSet<String>();
set.add("Ravi");
set.add("Vijay");
set.add("Ravi");
set.add("Ajay");
Iterator<String> itr=set.iterator();
while(itr.hasNext()){
System.out.println(itr.next());
} } }
```

LinkedHashSet Vs. HashSet

- Both LinkedHashSet and HashSet implements the Set interface. However, there exist some differences between them.

- LinkedHashSet maintains a linked list internally. Due to this, it maintains the insertion order of its elements.

- The LinkedHashSet class requires more storage than HashSet. This is because LinkedHashSet maintains linked lists internally.

- The performance of LinkedHashSet is slower than HashSet. It is because of linked lists present in LinkedHashSet.

Characteristics of TreeSet

- It is a concrete implementing class of set interface.
- TreeSet will also have all those methods present in collection interface.
- The elements are sorted by default in ascending order.
- The elements to be added in TreeSet should be of comparable type, else we get ClassCastException
- The elements to be added in TreeSet should be of same type (Homogenous)if not we get ClassCastException.
- Duplicate elements not allowed.
- No indexing, therefore we cannot add, remove elements using index.

1.Insert Elements to TreeSet

- add() - inserts the specified element to the set
- addAll() - inserts all the elements of the specified collection to the set

2.Access TreeSet Elements

- To access the elements of a tree set, we can use the iterator() method. In order to use this method, we must import the java.util.Iterator package.

3.Remove TreeSet Elements

•remove() - removes the specified element from the set
•removeAll() - removes all the elements from the set

Other Methods Of TreeSet

Method	Description
clone()	Creates a copy of the TreeSet
contains()	Searches the TreeSet for the specified element and returns a boolean result
isEmpty()	Checks if the TreeSet is empty
size()	Returns the size of the TreeSet
clear()	Removes all the elements from the TreeSet

TreeSet

```
public class Demo{
public static void main(String args[]){
//Creating and adding elements
TreeSet<String> set=new TreeSet<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());
} }}
```