Collections in Java

The **Collection in Java** is a framework that provides an architecture to store and manipulate the group of objects.

Java Collections can achieve all the operations that you perform on a data such as searching, sorting, insertion, manipulation, and deletion.

Java Collection means a single unit of objects. Java Collection framework provides many interfaces (Set, List, Queue, Deque) and classes ([ArrayList](https://www.javatpoint.com/java-arraylist), Vector, [LinkedList](https://www.javatpoint.com/java-linkedlist), [PriorityQueue](https://www.javatpoint.com/java-priorityqueue), HashSet, LinkedHashSet, TreeSet).

### **Hierarchy of Collection Framework**



## **List Interface**

List interface is the child interface of Collection interface. It inhibits a list type data structure in which we can store the ordered collection of objects. It can have duplicate values.

List interface is implemented by the classes ArrayList, LinkedList, Vector, and Stack.

To instantiate the List interface, we must use :

1. List <data-type> list1= **new** ArrayList();
2. List <data-type> list2 = **new** LinkedList();
3. List <data-type> list3 = **new** Vector();
4. List <data-type> list4 = **new** Stack();

## **ArrayList**

The ArrayList class implements the List interface. It uses a dynamic array to store the duplicate element of different data types. The ArrayList class maintains the insertion order and is non-synchronized. The elements stored in the ArrayList class can be randomly accessed. Consider the following example.

**import** java.util.\*;

**class** TestJavaCollection1{

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

ArrayList<String> list=**new** ArrayList<String>();

list.add("Ravi");

list.add("Vijay");

list.add("Ravi");

list.add("Ajay");

Iterator itr=list.iterator();

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

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

}

}

}

## **LinkedList**

LinkedList implements the Collection interface. It uses a doubly linked list internally to store the elements. It can store the duplicate elements. It maintains the insertion order and is not synchronized. In LinkedList, the manipulation is fast because no shifting is required.

Consider the following example.

**import** java.util.\*;

**public** **class** TestJavaCollection2{

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

LinkedList<String> al=**new** LinkedList<String>();

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());

}

}

}

## **Vector**

Vector uses a dynamic array to store the data elements. It is similar to ArrayList. However, It is synchronized and contains many methods that are not the part of Collection framework.

Consider the following example.

**import** java.util.\*;

**public** **class** TestJavaCollection3{

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

Vector<String> v=**new** Vector<String>();

v.add("Ayush");

v.add("Amit");

v.add("Ashish");

v.add("Garima");

Iterator<String> itr=v.iterator();

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

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

}

}

}

## **Stack**

The stack is the subclass of Vector. It implements the last-in-first-out data structure, i.e., Stack. The stack contains all of the methods of Vector class and also provides its methods like boolean push(), boolean peek(), boolean push(object o), which defines its properties.

Consider the following example.

**import** java.util.\*;

**public** **class** TestJavaCollection4{

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

Stack<String> stack = **new** Stack<String>();

stack.push("Ayush");

stack.push("Garvit");

stack.push("Amit");

stack.push("Ashish");

stack.push("Garima");

stack.pop();

Iterator<String> itr=stack.iterator();

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

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

}

}

}

## **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. We can store at most one null value in Set. Set is implemented by HashSet, LinkedHashSet, and TreeSet.

Set can be instantiated as:

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

## **HashSet**

HashSet class implements Set Interface. It represents the collection that uses a hash table for storage. Hashing is used to store the elements in the HashSet. It contains unique items.

Consider the following example.

**import** java.util.\*;

**public** **class** TestJavaCollection7{

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

HashSet<String> set=**new** HashSet<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**

LinkedHashSet class represents the LinkedList implementation of Set Interface. It extends the HashSet class and implements Set interface. Like HashSet, It also contains unique elements. It maintains the insertion order and permits null elements.

Consider the following 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());

}

}

}

## **TreeSet**

Java TreeSet class implements the Set interface that uses a tree for storage. Like HashSet, TreeSet also contains unique elements. However, the access and retrieval time of TreeSet is quite fast. The elements in TreeSet stored in ascending order.

Consider the following example:

**import** java.util.\*;

**public** **class** TestJavaCollection9{

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

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

}

}

}