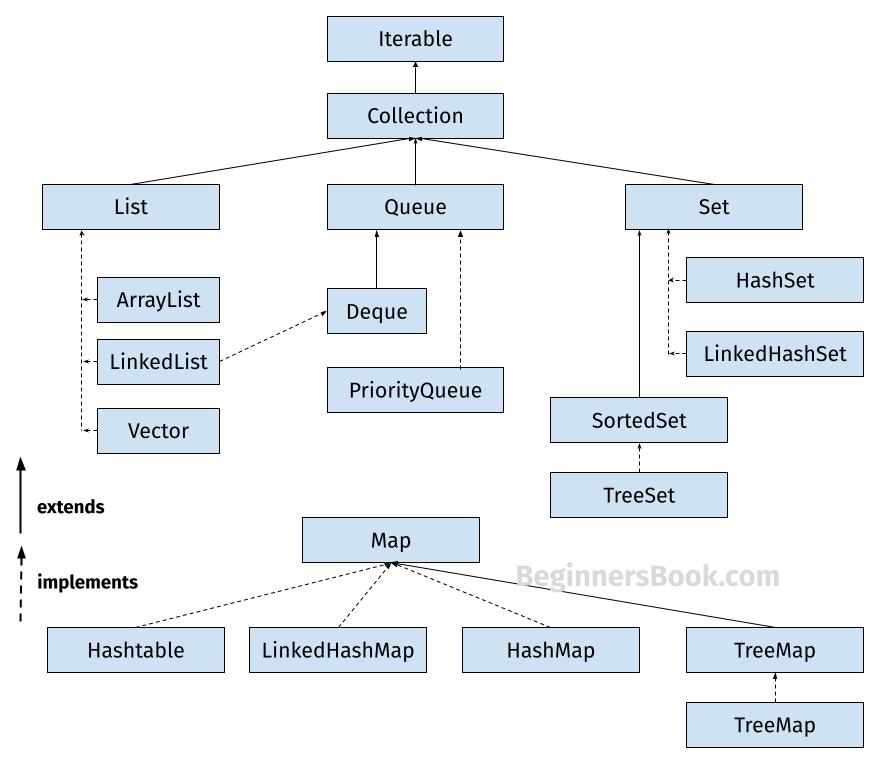
Collections in java:

The **Java Collections Framework** is a collection of interfaces and classes, which helps in storing and processing the data efficiently.



## 1. List

A List is an ordered Collection (sometimes called a sequence). Lists may contain duplicate elements. Elements can be inserted or accessed by their position in the list, using a zero-based index. The classes that implements List interface are:

* ArrayList
* LinkedList
* Vector
* Stack

### 1.1 ArrayList

ArrayList is a popular alternative of [**arrays in Java**](https://beginnersbook.com/2013/05/java-arrays/). It is based on an Array **data structure**. ArrayList is a resizable-array implementation of the List interface. It, and permits all elements, including null

import java.util.\*;

class JavaExample{

public static void main(String args[]){

//creating ArrayList of string type

ArrayList<String> arrList=new ArrayList<>();

//adding few elements

arrList.add("Cricket"); //list: ["Cricket"]

arrList.add("Hockey"); //list: ["Cricket", "Hockey"]

//inserting element at first position, index 0

//represents first element because ArrayList is based

//on zero based indexing system

arrList.add(0, "BasketBall"); //list: ["BasketBall", "Cricket", "Hockey"]

System.out.println("ArrayList Elements: ");

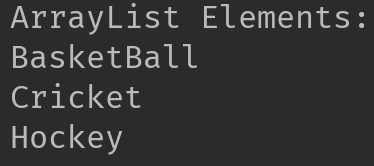
//Traversing ArrayList using enhanced for loop

for(String str:arrList)

System.out.println(str);

}

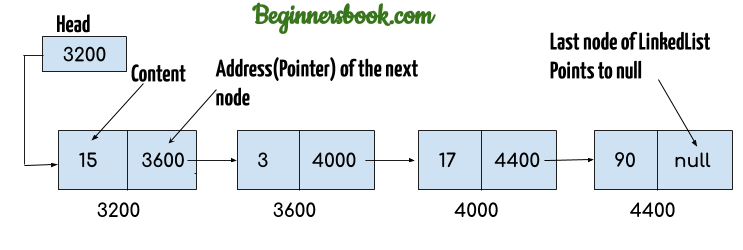
}

**Output:**  
  
**Refer the following guides to learn More about ArrayList:**

* [**ArrayList in Java**](https://beginnersbook.com/2013/12/java-arraylist/)
* [**Array vs ArrayList**](https://beginnersbook.com/2022/08/difference-between-array-and-arraylist/)
* [**Iterate ArrayList**](https://beginnersbook.com/2013/12/how-to-loop-arraylist-in-java/)
* [**Sort ArrayList**](https://beginnersbook.com/2013/12/how-to-sort-arraylist-in-java/)
* [**Convert Array to ArrayList**](https://beginnersbook.com/2013/12/how-to-convert-array-to-arraylist-in-java/)

Refer [**this collection**](https://beginnersbook.com/tag/java-arraylist/), which contains all the articles related to ArrayList published on this website. It is **regularly updated** whenever a new article on ArrayList topic is published on this site.

### 1.2 LinkedList

LinkedList is a linear data structure. However LinkedList elements are not stored in contiguous locations like arrays, they are linked with each other using pointers. Each element of the LinkedList has the reference(address/pointer) to the next element of the LinkedList.  


import java.util.\*;

public class JavaExample{

public static void main(String args[]){

LinkedList<String> linkList=new LinkedList<>();

linkList.add("Apple"); //["Apple"]

linkList.add("Orange"); //["Apple", "Orange"]

//inserting element at first position

linkList.add(0, "Banana"); ////["Banana", "Apple", "Orange"]

System.out.println("LinkedList elements: ");

//iterating LinkedList using iterator

Iterator<String> it=linkList.iterator();

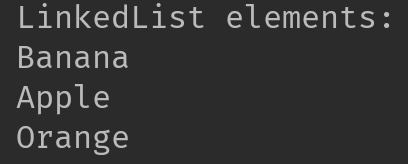
while(it.hasNext()){

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

}

}

}

**Output:**  
  
**Refer these article to learn More about LinkedList:**

* [**LinkedList in Java**](https://beginnersbook.com/2013/12/linkedlist-in-java-with-example/)
* [**How to iterate LinkedList**](https://beginnersbook.com/2013/12/how-to-loop-linkedlist-in-java/)
* [**Add element at the beginning and end of LinkedList**](https://beginnersbook.com/2014/07/java-add-elements-at-beginning-and-end-of-linkedlist-example/)
* [**Search element in LinkedList**](https://beginnersbook.com/2014/07/java-search-elements-in-linkedlist-example/)
* [**Convert LinkedList to ArrayList**](https://beginnersbook.com/2014/07/java-convert-a-linkedlist-to-arraylist/)

Refer [**this collection**](https://beginnersbook.com/tag/java-linkedlist/), which contains all the articles related to LinkedList published on this website.

### 1.3 Vector

Here is the list of all the tutorials published on the Vector.

import java.util.\*;

public class JavaExample{

public static void main(String args[]){

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

v.add("item1"); //["item1"]

v.add("item2"); //["item1", "item2"]

v.add("item3"); //["item1", "item2", "item3"]

//removing an element

v.remove("item2"); //["item1", "item3"]

System.out.println("Vector Elements: ");

//iterating Vector using iterator

Iterator<String> it=v.iterator();

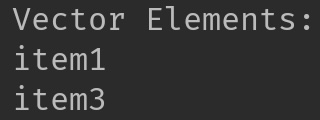
while(it.hasNext()){

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

}

}

}

**Output:**  
  
Refer [**this article**](https://beginnersbook.com/2013/12/vector-in-java/) for more **guides on Vector**.

### 1.4 Stack

Stack class extends Vector class, which means it is a subclass of Vector. Stack works on the concept of Last In First Out (LIFO). The elements are inserted using push() method at the end of the stack, the pop() method removes the element which was inserted last in the Stack.

import java.util.\*;

public class JavaExample{

public static void main(String args[]){

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

//push() method adds the element in the stack

//and pop() method removes the element from the stack

stack.push("Chaitanya"); //["Chaitanya"]

stack.push("Ajeet"); //["Chaitanya", Ajeet]

stack.push("Hari"); //["Chaitanya", "Ajeet", "Hari"]

stack.pop(); //removes the last element

stack.push("Steve"); //["Chaitanya", "Ajeet", "Steve"]

stack.push("Carl"); //["Chaitanya", "Ajeet", "Steve", "Carl"]

stack.pop(); //removes the last element

System.out.println("Stack elements: ");

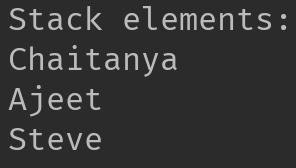
for(String str: stack){

System.out.println(str);

}

}

}

**Output:**  


## 2. Set

A Set is a Collection that cannot contain duplicate elements. There are three main implementations of Set interface: HashSet, TreeSet, and LinkedHashSet.

### 2.1 HashSet

[**HashSet**](https://beginnersbook.com/2013/12/hashset-class-in-java-with-example/) which stores its elements in a hash table, is the best-performing implementation. HashSet allows only unique elements. It doesn’t maintain the insertion order which means element inserted last can appear at first when traversing the HashSet.

import java.util.\*;

public class JavaExample{

public static void main(String args[]){

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

set.add("Paul");

set.add("Ram");

set.add("Aaron");

set.add("Leo");

set.add("Becky");

Iterator<String> it=set.iterator();

while(it.hasNext()){

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

}

}

}

**Output:**

Aaron

Leo

Paul

Ram

Becky

### 2.2 LinkedHashSet

Unlike HashSet, the **[LinkedHashSet](https://beginnersbook.com/2013/12/linkedhashset-class-in-java-with-example/)** maintains insertion order.

import java.util.\*;

public class JavaExample{

public static void main(String args[]){

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

set.add("Paul");

set.add("Ram");

set.add("Aaron");

set.add("Leo");

set.add("Becky");

Iterator<String> it=set.iterator();

while(it.hasNext()){

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

}

}

}

**Output:**

Paul

Ram

Aaron

Leo

Becky

### 2.3 TreeSet

[**TreeSet**](https://beginnersbook.com/2013/12/treeset-class-in-java-with-example/) stores elements in a red-black tree. It is substantially slower than HashSet. TreeSet class implements SortedSet interface, which allows TreeSet to order its elements based on their values, which means TreeSet elements are sorted in ascending order.

import java.util.\*;

public class JavaExample{

public static void main(String args[]){

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

set.add("Paul");

set.add("Ram");

set.add("Aaron");

set.add("Leo");

set.add("Becky");

Iterator<String> it=set.iterator();

while(it.hasNext()){

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

}

}

}

**Output:**

Aaron

Becky

Leo

Paul

Ram

## 3. Map

A Map is an object that maps keys to values. A map cannot contain duplicate keys. There are three main implementations of Map interfaces: HashMap, TreeMap, and LinkedHashMap.

### 3.1 HashMap

**HashMap:** HashMap is like HashSet, it doesn’t maintain insertion order and doesn’t sort the elements in any order. Refer [**this guide**](https://beginnersbook.com/2013/12/hashmap-in-java-with-example/) to **learn HashMap in detail**.

import java.util.\*;

public class JavaExample{

public static void main(String args[]){

HashMap<Integer, String> hmap = new HashMap<>();

//key and value pairs

hmap.put(101, "Chaitanya");

hmap.put(105, "Derick");

hmap.put(111, "Logan");

hmap.put(120, "Paul");

//print HashMap elements

Set set = hmap.entrySet();

Iterator iterator = set.iterator();

while(iterator.hasNext()) {

Map.Entry m = (Map.Entry)iterator.next();

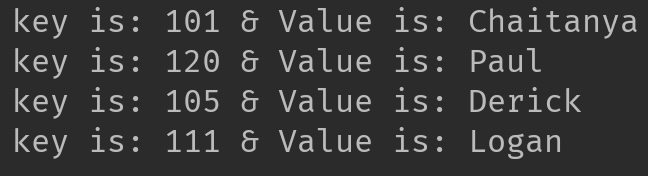
System.out.print("key is: "+ m.getKey() + " & Value is: ");

System.out.println(m.getValue());

}

}

}

**Output:**  


### 3.2 TreeMap

**TreeMap:** It stores its elements in a red-black tree. The elements of TreeMap are sorted in ascending order. It is substantially slower than HashMap. Refer [**this guide**](https://beginnersbook.com/2013/12/treemap-in-java-with-example/) to **learn TreeMap with examples**.

This is the same example that we have seen above in HashMap. Here, elements are sorted based on keys.

import java.util.\*;

public class JavaExample{

public static void main(String args[]){

TreeMap<Integer, String> hmap = new TreeMap<>();

//key and value pairs

hmap.put(101, "Chaitanya");

hmap.put(105, "Derick");

hmap.put(111, "Logan");

hmap.put(120, "Paul");

//print HashMap elements

Set set = hmap.entrySet();

Iterator iterator = set.iterator();

while(iterator.hasNext()) {

Map.Entry m = (Map.Entry)iterator.next();

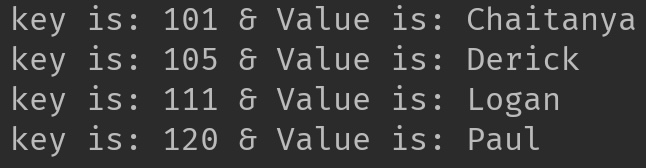
System.out.print("key is: "+ m.getKey() + " & Value is: ");

System.out.println(m.getValue());

}

}

}

**Output:**  


### 3.3 LinkedHashMap

**LinkedHashMap:** It maintains insertion order. Refer [**this guide**](https://beginnersbook.com/2013/12/linkedhashmap-in-java/), to learn LinkedHashMap in detail. As you can see: In the following example, the key & value pairs maintained the insertion order.

import java.util.\*;

public class JavaExample{

public static void main(String args[]){

LinkedHashMap<Integer, String> hmap = new LinkedHashMap<>();

//key and value pairs

hmap.put(100, "Chaitanya");

hmap.put(120, "Paul");

hmap.put(105, "Derick");

hmap.put(111, "Logan");

//print LinkedHashMap elements

Set set = hmap.entrySet();

Iterator iterator = set.iterator();

while(iterator.hasNext()) {

Map.Entry m = (Map.Entry)iterator.next();

System.out.print("key is: "+ m.getKey() + " & Value is: ");

System.out.println(m.getValue());

}

}

}

**Output:**

key is: 100 & Value is: Chaitanya

key is: 120 & Value is: Paul

key is: 105 & Value is: Derick

key is: 111 & Value is: Logan