**Collection Framework**

**Main Limitations of Array over Collection**

1. Array used to store similar type of data.

2. Size of array should be known before performing operations.

3. Each method must be defined for performing any operations search etc.

But In collection store Homogeneous and Heterogeneous type of data, Size problem solved and Interface in Java refers to the **abstract data types,** means structure in which method are predefined for performing search ,insert etc. and classes implements interfaces.

**What is Java collections?**

1. Java collections refer to a single unit of individual objects.

2. A Java collection framework provides an architecture to store and manipulate a group of objects.

3. The Java collection framework provides the developers to access prepackaged data structures as well as algorithms to manipulate data.

4. A Java collection framework includes the following:

* Interfaces
* Classes
* Algorithm

**A. Interfaces**: Interface in Java refers to the **abstract data types,** means structure in which method are predefined for performing search ,insert etc. They allow Java collections to be manipulated independently from the details of their representation.

**B. Classes:**Classes in Java are the **implementation** of the **collection interface**. It basically refers to the **data structures** that are used again and again.

**C. Algorithm:** Algorithm refers to the methods which are used to perform operations such as searching and sorting, on objects that implement collection interfaces.  You can perform all operations on data such as searching, sorting, insertion, manipulation, deletion, etc. by Java collections.

**HashMap:(unsynchronized)**

* **HashMap** allows one null key and multiple null values.
* It is unsynchronized. It has better performance. It is not-thread safe

**Map l= new HashMap();**

**l.put(1,null);**

**l.put(null,null);**

**l.put(null,20);**

**System.out.println(l);**

**Output:-** **{null=20, 1=null}//no guarantee what sequence of output**

As I is told that HashMap is unsynchronized to make synchronized it refer

Map m = Collections.synchronizedMap(new HashMap(...));

: <https://www.geeksforgeeks.org/java-util-hashmap-in-java-with-examples/>

**Hashtable:-**

**HashTable(**synchronized)

* It didn’t allow any null for key and value.
* It is synchronized. Only one thread can access in one time

**Map l= new Hashtable();**

**l.put(1,null); //error**

**l.put(2,45);**

**l.put(3,20);**

**System.out.println(l);**

**HashSet:-**

HashSet does not allow duplicate values.(not null more than one i.e also duplicate null value).

It provides **add** method rather **put** method.

**HashSet**  implements **Set** Interface.

HashSet stores its element by a process called **hashing**. In hashing, hash code of every element is computed internally.

**Set l= new HashSet();**

**l.add(null);**

**l.add(null);**

**l.add(45);**

**l.add(20);**

**System.out.println(l);**

**OutPut:** **[null, 20, 45]**

Both HashMap and HashSet are not synchronized

 HashMap is non synchronized. It is not-thread safe and can’t be shared between many threads without proper synchronization code whereas Hashtable is synchronized. It is thread-safe and can be shared with many threads.

**Comparable VS Comparator**

Comparable provide single way of sorting.(compareTo()); store in java.lang

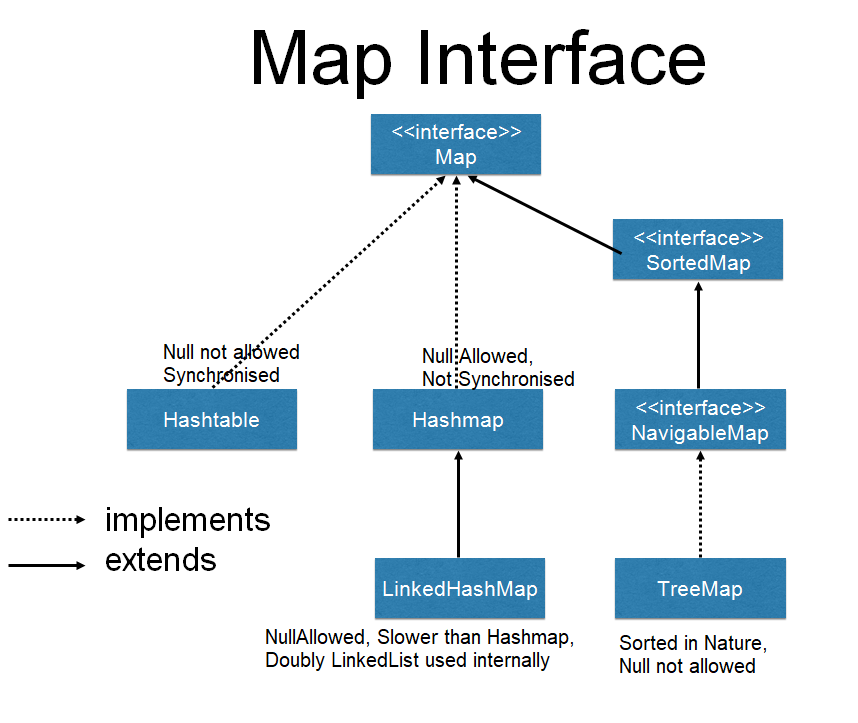
Comparator provide different ways of sorting.(compare()); store in java.util\*

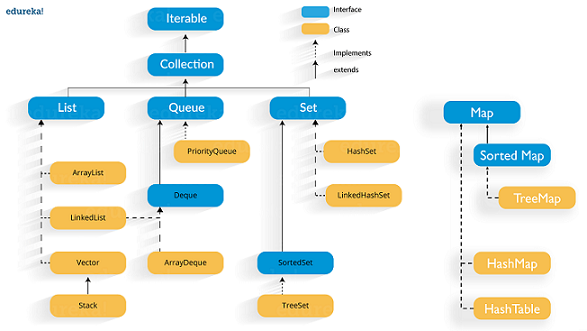
**Default Natural Ordering possible**

1. Object must be homogeneous type (TreeSet and Map)

2. Object must be comparable type (String and Wrapper Class).

|  |  |
| --- | --- |
| **HashMap** | **TreeMap** |
| 1) HashMap can contain one null key. | TreeMap cannot contain any null key. |
| 2) HashMap maintains no order. | TreeMap maintains ascending order. |





A Map doesn't allow duplicate keys, but you can have duplicate values. **HashMap and LinkedHashMap** allow null keys and values, but **TreeMap** doesn't allow any null key or value

|  |  |
| --- | --- |
|  |  |

Difference: Vector and ArrayList:

Vector increases the capacity twice of its initial size while ArrayList increases its Array size by 50%.

List allows any number of null values.

Set allows single null value at most.

Map can have single null key at most and any number of null values.

**Methods of Iterator:**

hasNext()

next()

remove()

**Methods of ListIterator:**

add(E e)

hasNext()

hasPrevious()

next()

nextIndex()

previous()

previousIndex()

remove()

set(E e)