12/03/11

Collection framework

→ An Array is an indexed Collection of fixed no. of homogeneous data elements.

* Limitations of object arrays :-

→(1) Arrays are fixed in size. i.e, once we created an array there is no chance of increasing or decreasing size based on our requirement. Hence, to use arrays concept compulsory we should know the size in advance, which may not possible always.

(2) Arrays can hold only homogeneous data elements. i.e, (same type)

ex!-
```
Student[] S = new Student[1000];
S[0] = new Student[]; ✓
S[1] = new Student[]; ✓
S[2] = new Customer[]; ✗ CE!- Incompatible types
                              found : Customer[]
                              required : Student.
```

↳ But we can resolve this problem by using object type arrays.

ex!-
```
Object[] a = new Object[1000];
a[0] = new Student[]; ✓
a[1] = new Customer[]; ✓
```

(3) Arrays concept not built based on some datastructure. Hence readymed method support is not available for every requirement. Compulsory programmer is responsible to write the logic.

→ To resolve the above problems SUN people introduced Collections Concept.

→ Advantages of Collections over arrays :-

(1) Collections are growable in nature. Hence based on our requirement we can increase or decrease the size.

(2) Collections can hold both homogeneous & heterogeneous objects.

(3) Every Collection class is implemented based on some datastructure. Hence readymed method support is available for every requirement.

dis. of Collections :-

→ Performance point of view Collections are not recommended to use. This is the limitation of Collections.

difference b/w arrays & Collections :-

Array	Collections(AL, VL, LL - - - ..)
1) Arrays are fixed in size	1) Collections are growable in nature
2) Memory point of view arrays Concept is not recommended to use	2) memory point of view Collections Concept is highly recommended to use.
3) Performance point of view arrays Concept is highly recommended to use.	3) Performance point of view Collections is not recommended to use.
4) Arrays can hold only homogeneous data elements.	4) Collections can hold both Homogeneous & Heterogeneous objects.
5) There is no underlying d.s for arrays. Hence readymed method support is not available	5) Underlying D.S is available for every Collection class. Hence readymed method support is available.

→ Arrays can be used to hold both primitives & objects.	→ Collections can be used to hold only objects but not for primitives.

Collection :-

→ A group of individual objects as a single entity is called Collection

Collection framework :-

→ It defines several classes & Interfaces, which can be used to represent a group of objects as a single Entity.

Terminology :-

Java	C++
Collection	→ Container
Collection framework	→ STL (Standard Template Library)

9-Key interfaces of Collection framework :-

① Collection (Interface) :

→ If we want to represent a group of individual objects as a single Entity then we should go for Collection.

→ In general Collection Interface is considered as root Interface of Collection framework.

→ Collection Interface defines the most common methods which can be applicable for any Collection object.

Collection vs Collections :-

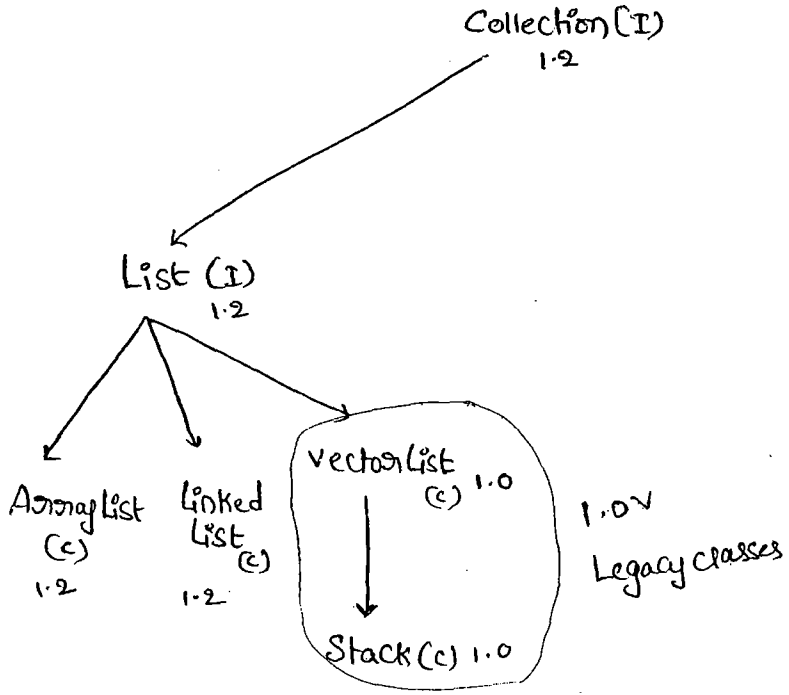
→ Collection is an interface, Can be used to represent a group of individual object as a Single Entity. where as,

→ Collections is an Utilityclass, present in java.util package, to define Several Utility methods for Collections.

3) List (Interface) :-

→ It is the child Interface of Collection.

→ If we want to represent a group of individual objects where insertion order is preserved & duplicates are allowed. Then we should go for List.

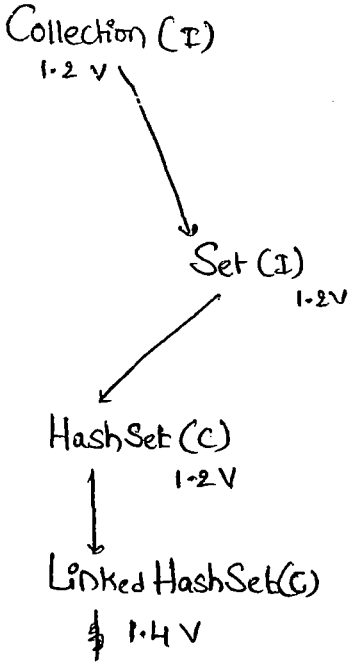


→ Vector & Stack classes are Re Engineered in 1.2 version to fit into Collection framework.

③ Set (Interface):-

→ It is the child Interface of Collection.

→ If we want to represent a group of individual objects where duplicates are not allowed & insertion order is not preserved. Then we should go for "Set".



④ SortedSet (I):-

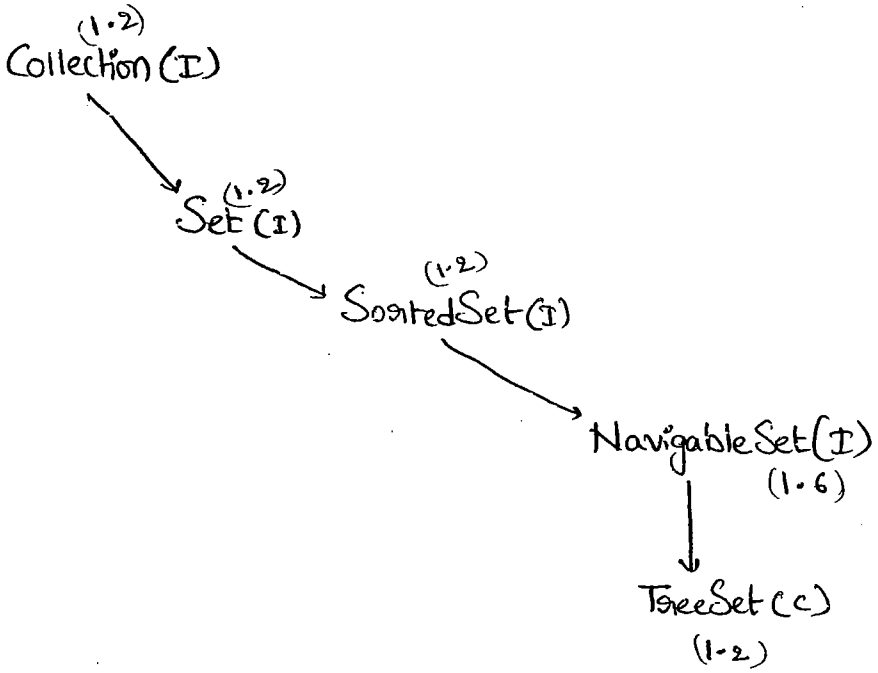
→ It is the child interface of Set.

→ if we want to represent a group of individual objects, according to some Sorting order. then we should go for SortedSet.

⑤ NavigableSet (I):

→ It is the child interface of SortedSet, to provide several methods for Navigation purposes.

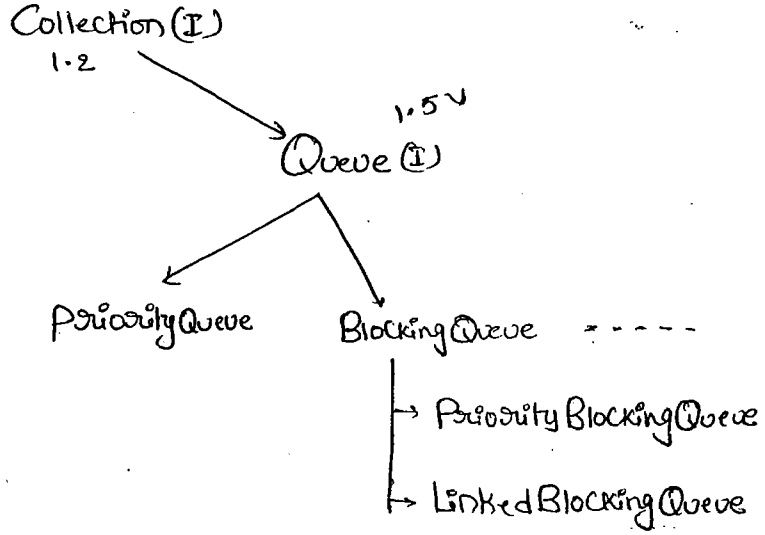
→ It is introduced in 1.6Version.



⑥ Queue (I) :- (1.5V)

→ It is the child Interface of Collection.

→ If we want to represent a group of individual objects prior to processing then we should go for Queue.



Note :-

→ all the above Interfaces (Collection, List, Set, SortedSet, NavigableSet, Queue) ment for representing a group of individual objects.

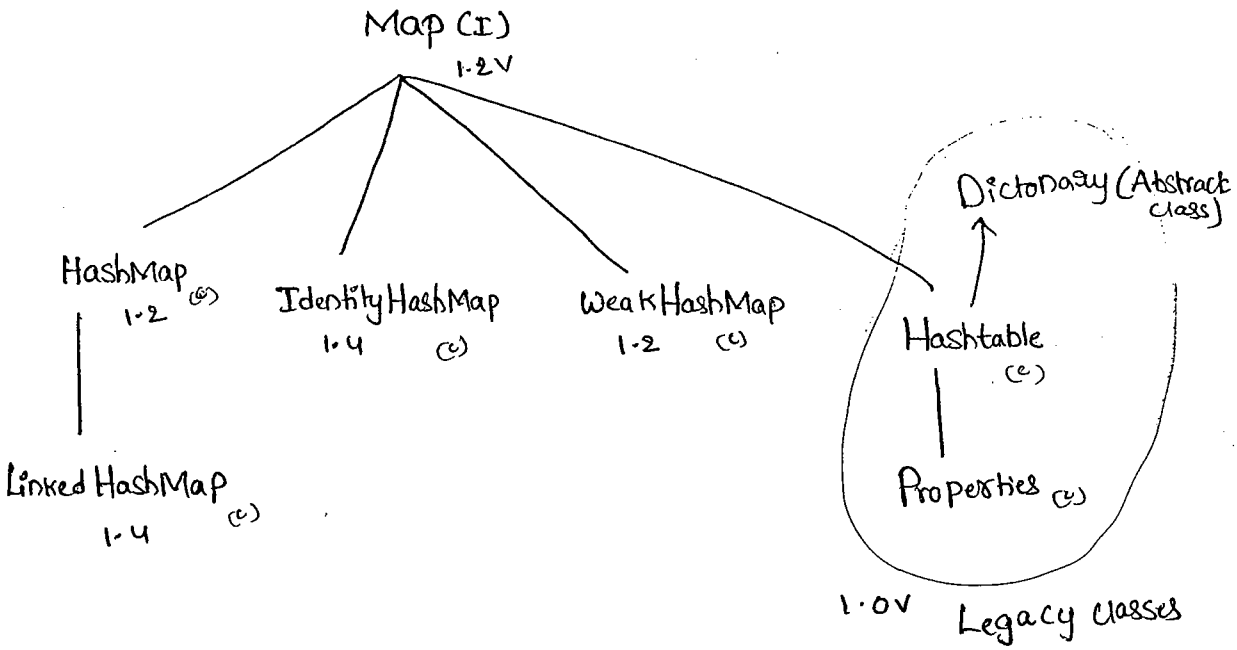
→ If we want to represent a group of objects as key-value pairs then we should go for Map.

(7) Map(I):-

→ If we want to represent a group of objects as Key-value pairs then we should go for Map.

→ Both Key & value are objects only.

→ duplicate Keys are not allowed, But values Can be duplicated.



Note:-

⟹ ** Map is not child Interface of Collection.

(8) SortedMap (I) :-

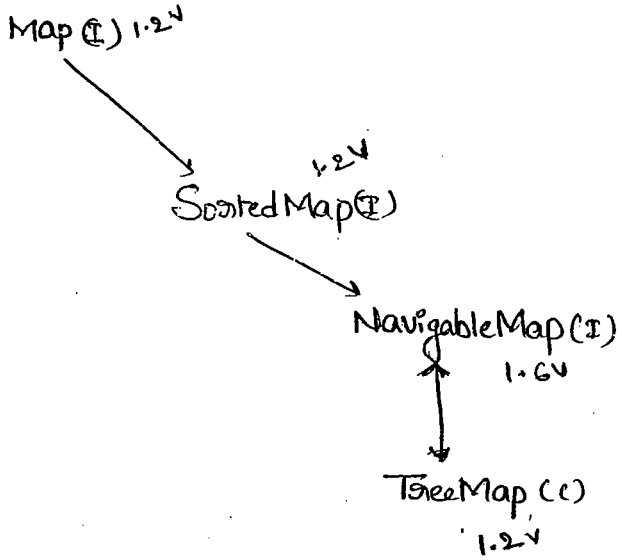
→ If we want to represent a group of objects as Key-value pairs according to Some Sorting Order. Then we Should go for SortedMap.

→ Sorting should be done only based on Keys. but not based-on values.

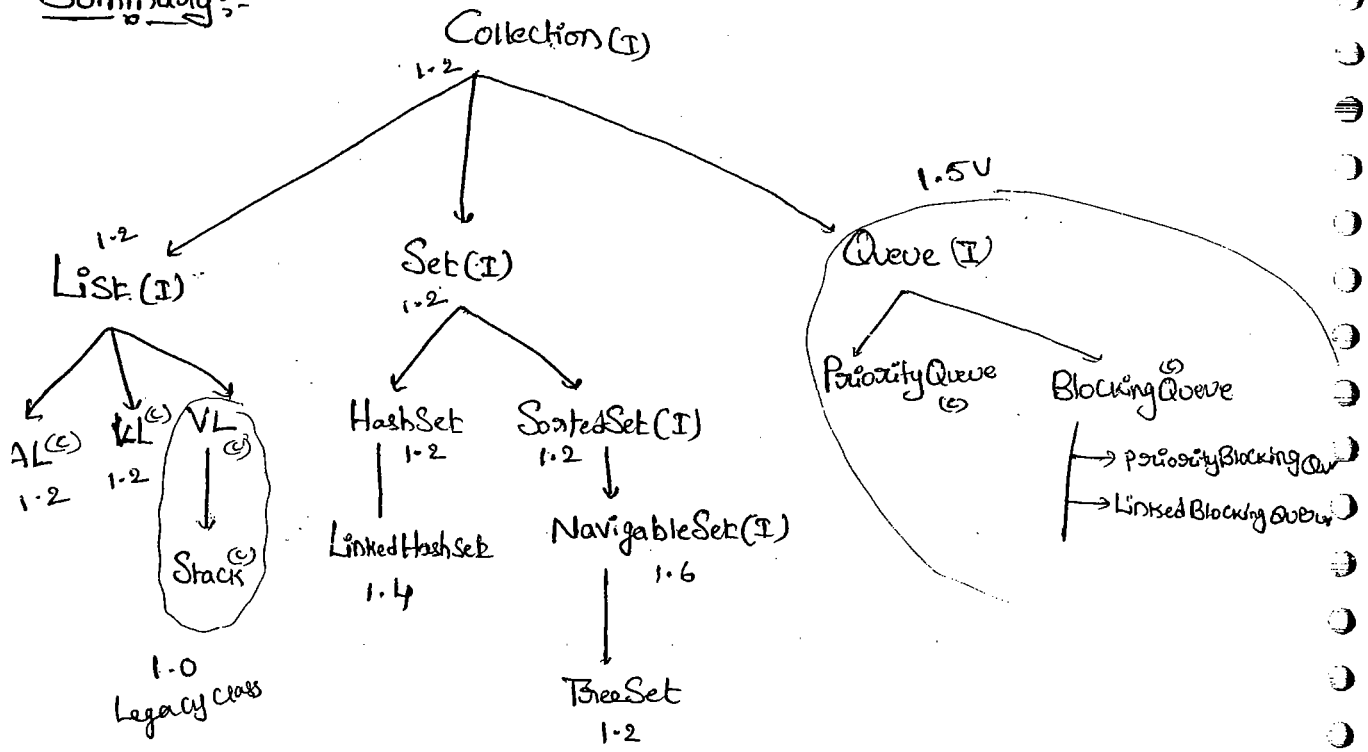
→ SortedMap is child Interface of Map.

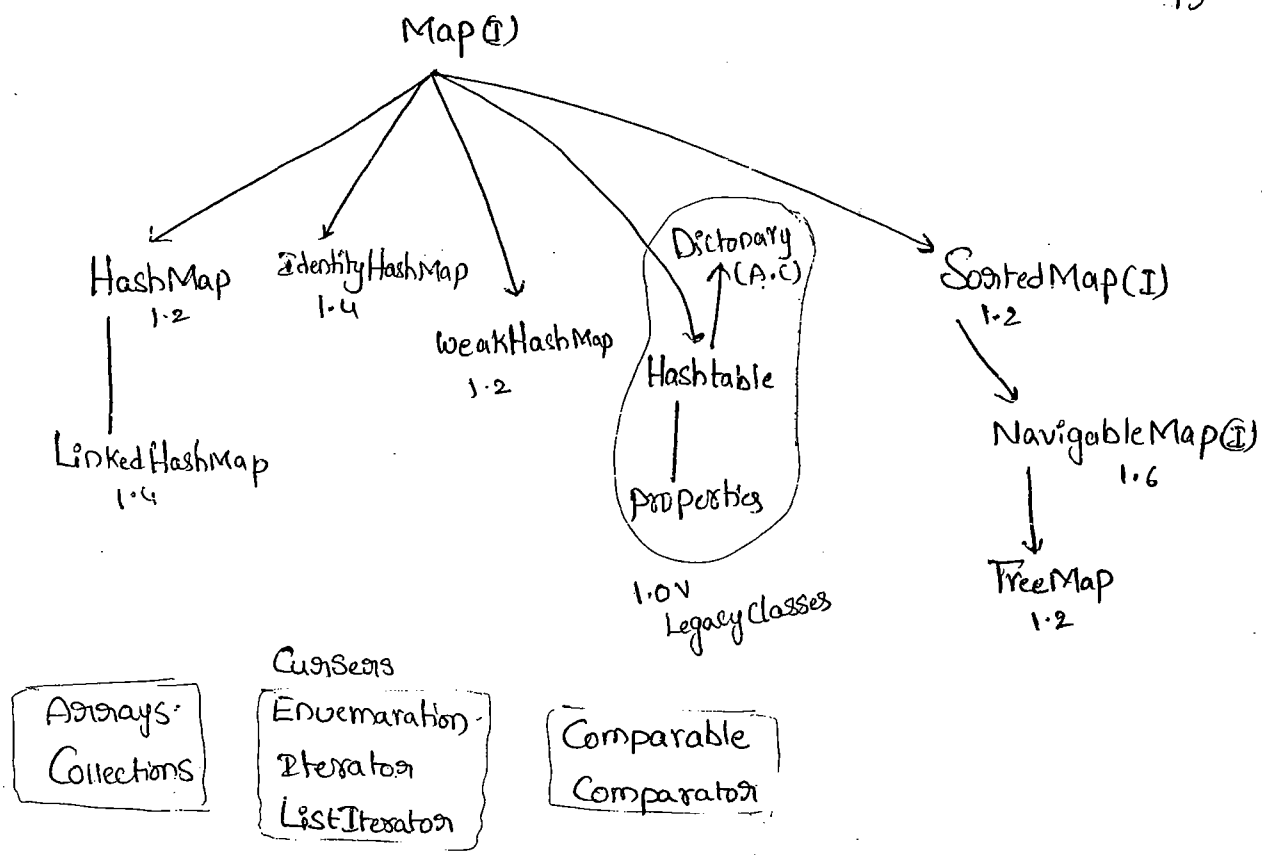
(9) NavigableMap (I):-

→ It is the child Interface of SortedMap & define Several methods for Navigation purposes.



Summary:-





→ In the Collection frame work the following are Legacy characters

(1) Enumaration (I)
(2) Dictionary (Ab c)
(3) Vector
(4) Stack
(5) Hashtable
(6) properties

(3) to (6) → classes

(1) to (6) → 1.0V

Collection framework:

Collection(I):-

→ If we want to represent a group of individual objects as a single entity then we should go for Collection.

→ Collection Interface defines the most common methods which can be applied for any Collection object.

→ The following is the list of methods present in Collection Interface.

① boolean add(Object o)

② boolean addAll(Collection c)

③ boolean remove(Object o)

④ boolean removeAll(Collection c)

⑤ boolean retainAll(Collection c)

→ To remove all objects except those present in C.

⑥ void clear()

⑦ boolean isEmpty()

⑧ int size()

⑨ boolean contains(Object o)

⑩ boolean containsAll(Collection c)

⑪ Object[] toArray()

⑫ Iterator iterator()

(2) List (I) :-

→ List is the child Interface of Collection.

→ If we want to represent a group of individual objects where duplicate objects are allowed & insertion order is preserved. Then we should go for List.

→ Insertion order will be preserved by means of Index.

→ We can differentiate duplicate objects by using Index. Hence Index place a very important role in List.

→ List Interface defines the following methods

1. boolean add(int index, Object o)
2. boolean addAll(int index, Collection c)
3. Object remove(int index)
4. Object get(int index)
5. Object set(int index, Object new)
   (old)
6. int indexOf(Object o)
7. int lastIndexOf(Object o)
8. ListIterator listIterator()

It contains 4 classes :-

(i) ArrayList (C) :- 1.2

(ii) LinkedList (C) :- 1.2

(iii) VectorList (C) :- 1.0

(iv) Stack (C) :- 1.0

(i) ArrayList (c) :-

→ The underlying datastructure for ArrayList is Resizable Array or Growable Array.

→ Insertion Order is preserved.

→ Duplicate objects are allowed.

→ Heterogeneous objects are allowed.

→ Null insertion is possible.

Constructors :-

(1) `ArrayList AL = new ArrayList();`

→ Creates an Empty ArrayList Object, with default initial Capacity 10.

→ Once AL reaches it's max. capacity then a new AL object will be Created with.

$$\text{New Capacity} = \text{CurrentCapacity} * \frac{3}{2} + 1$$

(2) `ArrayList l = new ArrayList(int initialCapacity);`

→ Creates an Empty Arraylist object with the Specified initial Capacity.

(3) `ArrayList l = new Arraylist(Collection c);`

→ Creates an Equivalent ArrayList object for the Given Collection object

ie, this Constructor is for dancing b/w Collection objects

Ex:-

```
import java.util.*;
class ArrayListDemo
{
  P.S.V.m(String[] args)
  {
    ArrayList a = new ArrayList();
    a.add("A");
    a.add(10);
    a.add("A");
    a.add(null);
    S.o.pln(a);   [A, 10, A, null]
                   0  1  2   3
    a.remove(2);
    S.o.pln(a);   [A, 10, null]
    a.add(2,"M"); [A, 10, M, null]
    a.add("N");   [A, 10, M, null, N]
    S.o.pln(a);   [A, 10, M, null, N]
  }
}
```

```
S.o.pln(a.size()); //5
a.clear(); // []
a.addAll(a); //[A,10,M,null,N,A,10,M,null,N]
```

Note:-

⊛ In Every Collection class toString() is overridden to return its Content directly in the following formatt.

[obj1, obj2, obj3 .....]

↪ Usually we Can Use Collection to Store & transfer Objects. to provide Support for this requirement Every Collection class implements Serializable & Clonable Interfaces.

→ ArrayList & Vector classes implements RandomAccess Interface, so that any random element we can access with same speed. Hence, if our frequent operation is retrivable operation then best suitable data structure is ArrayList. (Advantage)

→ If our frequent operation is Insertion or deletion operation in the middle then ArrayList is the worst choice, because it required several shift operations. (disadvantage)

Q: differences b/w ArrayList & Vector?

ArrayList	Vector
① No method is Synchronized	① Every method is Synchronized
② multiple threads can access ArrayList simultaneously. hence ArrayList object is not threadsafe	② At any point only one thread is allowed to operate on vector object at a time. Hence vector object is Thread Safe.
③ Threads are not required to wait, & Hence performance is high.	③ It increases waiting time of threads & Hence performance is Low.
④ Introduced in 1.2 version & Hence it is non-legacy	⑤ Introduced in 1.0 version & Hence it is Legacy.

Q) How to get Synchronized Version of ArrayList?

A) → By using Collections class SynchronizedList() we can get synchronized version of ArrayList.

```
Public static List SynchronizedList(List l)
```

Eg:-

```
ArrayList l = new ArrayList();
List l1 = Collections.SynchronizedList(l)
```

l1 → Synchronized

l → NON-Synchronized.

→ Similarly we can get Synchronized Version of Set & Map objects by using the following methods respectively.

① public static Set SynchronizedSet(Set s)

② public static Map SynchronizedMap(Map m)

Note:-

→ If our frequent Operation is Insertion or deletion in the middle then ArrayList is not recommended. to handle this requirement we should go for LinkedList.