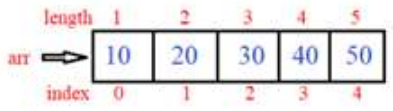
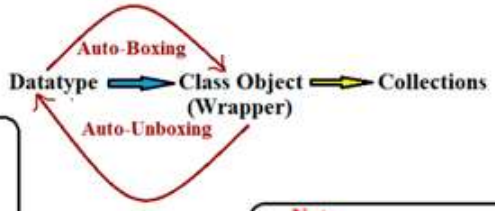
Java Array: Array collects multiple elements of **similiar** datatypes in a continous block of memory

```
int arr[]=new int[5];
  arr[0]=10;
  arr[1]=20;
  arr[2]=30;
  arr[3]=40;
  arr[4]=50;
```



Drawbacks

1) Array length is fixed.
2) Array allows only homogeneous data
3) Array does not have any method support



int ⇒ Integer
byte ⇒ Byte
short ⇒ Short
long ⇒ Long
float ⇒ Float
double ⇒ Double
char ⇒ Character
boolean ⇒ Boolean

Note

All Collection classes will NOT accepts datatypes, they accepts ONLY **Objects.**

Collection Framework

ArrayList
LinkedLIst
Vector
HashSet
LinkedHashSet
TreeSet
HashMap
Hashtable ....etc

java.util

primitive data types — Wrapper classes

Wrapper classes present in java.lang package

Collection framework is a group of classes as above

Drawbacks

1) Array length is fixed.
2) Array allows only homogeneous data
3) Array does not have any method support

```
int arr[]=new int[3];
  arr[0]=10;
  arr[1]=20;
  arr[2]=30;
  arr[3]=40;// AIOB Exception

Vector v=new Vector();
```

11	"hi"	22	'A'	44	5.5	88	true	88	99	50									

Introduction

- In java if we want to store multiple items of homogenous data types we can use "Arrays".
- Arrays can hold both primitive data types and objects.

Example:

```
int i[]=new int[100];
Object o[]=new Object[10];
```

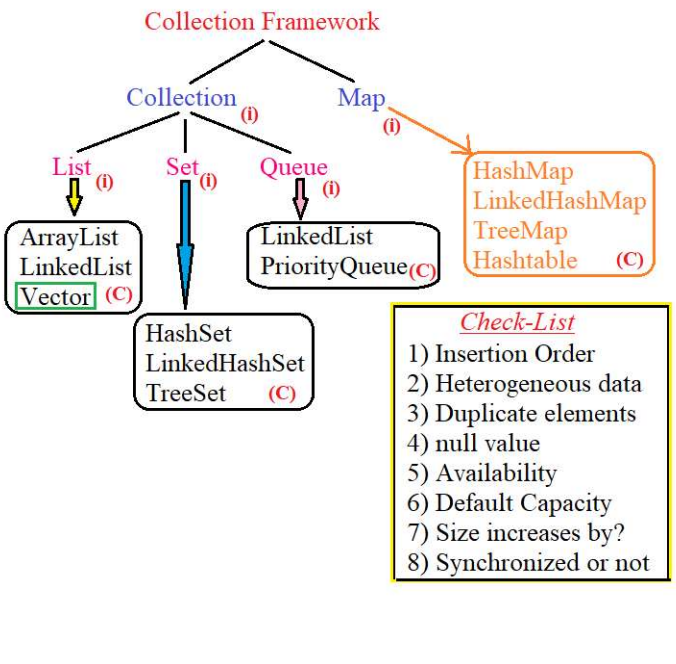
- Arrays don't have underlying data structures and algorithms.
- The Length of the arrays is fixed.
- Arrays can hold duplicate elements also.
- We can't store heterogeneous elements in an array.
- Inserting element at the end of array is easy but at the middle is difficult.
- After retrieving the elements from the array, in order to process the elements we don't have any methods.

- To overcome the above mentioned limitations we prefer Collections Framework.
- Collections size is not fixed based on our requirement, We can increase or decrease the size of a collection.
- Collections can hold both homogeneous and heterogeneous objects
- Every collection class is implemented based on some standard data structure & algorithms so predefined method support is available.

Array Vs Collections

Array	Collections
Arrays are fixed in size	Collections are growable in nature
Arrays can hold only homogeneous data types elements.	Collections can hold both homogeneous and heterogeneous elements.
There is no underlying data structure.	Every collection class is implemented based on some standard data structure.
Arrays can hold both object and primitive	Collection can hold only object types
Memory wise-> Recommended Performance Wise---> Not Recommended	Memory wise->Not Recommended Performance Wise---> Recommended

No underlying data structure means no method support.



List

In List implemented classes elements will be stored just like an array and ***List allows duplicates***

Set

In Set implemented classes elements will be stored just like an array and ***Set does not allows duplicates***

Queue

In Dueue implemented classes elements will be stored in the form of First-In-First-Out order (FIFO)

Map

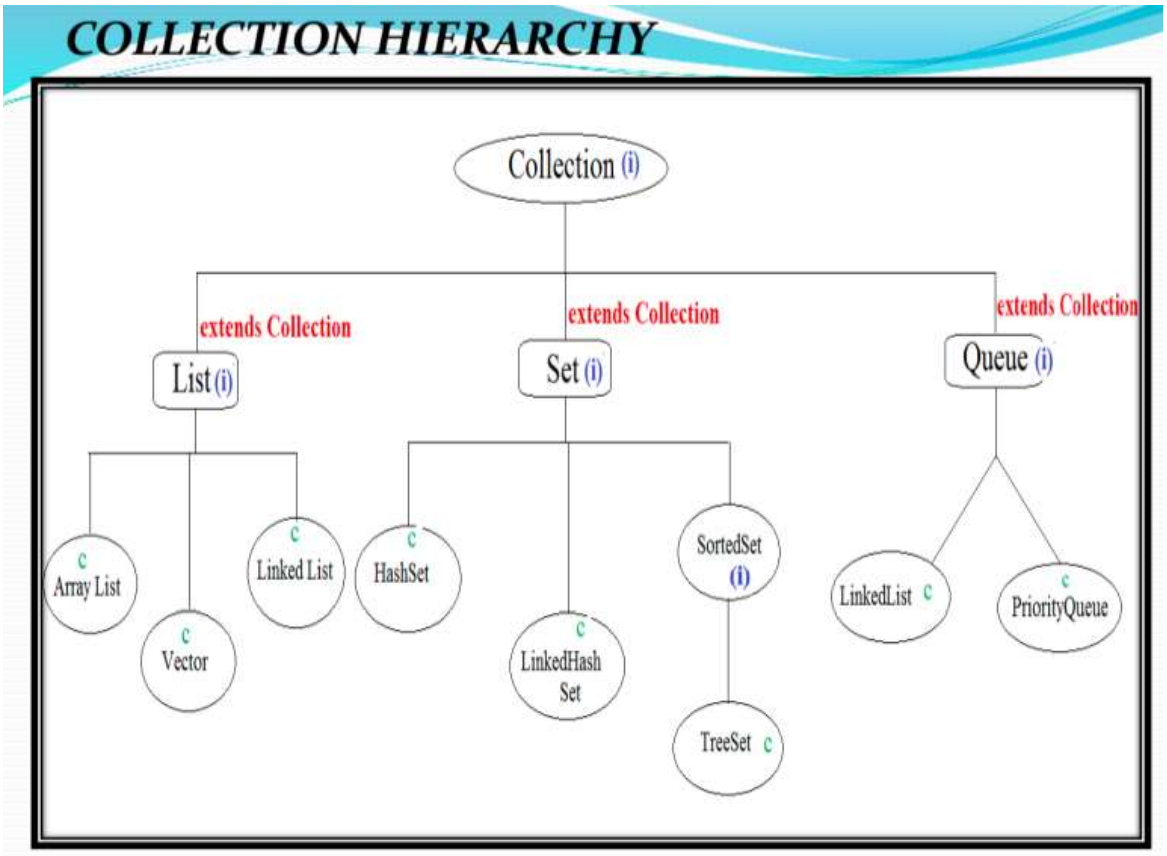
In Map implemented classes data will be stored in the form of **Key-Value** pairs. ex: Kishan - 101
(Key) (Value)

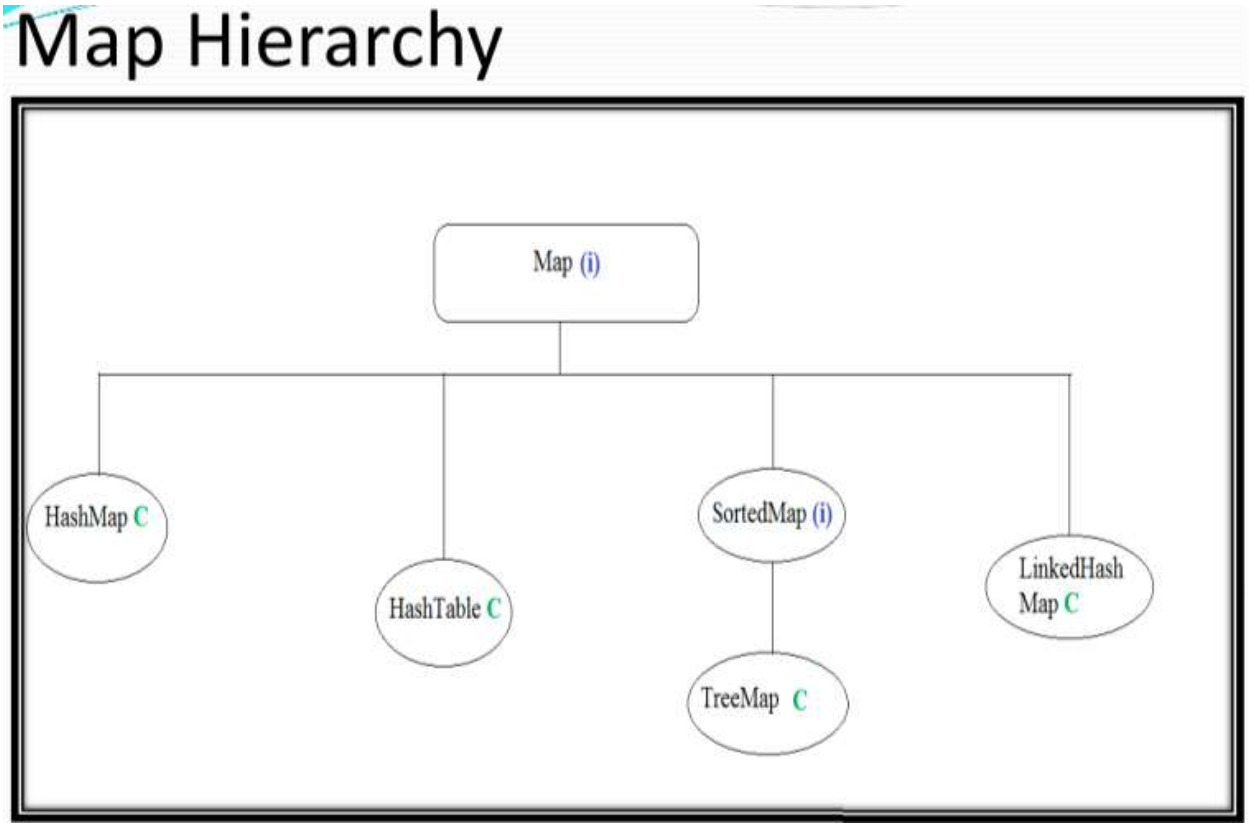
Q) How to reterive the data from Collection classes?

A)

1) By using for-loop
2) By using for-each loop

3) Iterator Interface : Forward Direction
4) ListIterator Interface : Forward & Backward Directions
5) Enumeration Interface : Used to reterive the data from legacy classes


COLLECTION HIERARCHY
Collection (i)
extends Collection
extends Collection
extends Collection
List (i)
Set (i)
Queue (i)
C Array List
C Vector
C Linked List
C HashSet
C LinkedHash Set
SortedSet (i)
TreeSet C
LinkedList C
C PriorityQueue


Map Hierarchy
Map (i)
HashMap C
HashTable C
SortedMap (i)
TreeMap C
LinkedHash Map C

Collection Object

- A collection object is an object which can store group of other objects.
- A collection object has a class called Collection class or Container class.
- All the collection classes are available in the package called 'java.util' (util stands for utility).
- Group of collection classes is called a Collection Framework.
- All the collection classes in java.util package are the implementation classes of different interfaces.
- In General Collection Framework consists of '3' parts
  1. Algorithms (Rules)
  2. Interfaces (abstract datatypes)
  3. Implementations (Concrete versions of these Interfaces)

Overview of Set, List, Queue, Map

- **Set:** A Set represents a group of elements (objects) arranged just like an array. The set will grow dynamically when the elements are stored into it. A set will not allow duplicate elements.

- **List:** Lists are like sets but allow duplicate values to be stored.

- **Queue:** A Queue represents arrangement of elements in FIFO (First In First Out) order. This means that an element that is stored as a first element into the queue will be removed first from the queue.

- **Map:** Maps store elements in the form of key value pairs. If the key is provided its corresponding value can be obtained.

Retrieving Elements from Collections:

- Following are the ways to retrieve any element from a collection object:
  - ✓Using Iterator interface.
  - ✓Using ListIterator interface.
  - ✓Using Enumeration interface.
- **Iterator Interface:** Iterator is an interface that contains methods to retrieve the elements one by one from a collection object. It retrieves elements only in forward direction. It has 3 methods:

Method	Description
boolean hasNext()	returns true if the iterator has more elements.
element next()	returns the next element in the iterator.
void remove()	removes the last element from the collection returned by the iterator.

ListIterator Interface:

- ListIterator is an interface that contains methods to retrieve the elements from a collection object, both in forward and reverse directions. It can retrieve the elements in forward and backward direction.
- It has the following important methods:

Method	Description
boolean hasNext()	returns true if the ListIterator has more elements when traversing the list in forward direction.
element next()	returns the next element.
void remove()	removes the list last element that was returned by the next () or previous () methods.
boolean hasPrevious()	returns true if the ListIterator has more elements when traversing the list in reverse direction
element previous()	returns the previous element in the list.

Enumeration Interface:

- This interface is useful to retrieve elements one by one like Iterator. [Mainly used in Legacy Classes]
- It has 2 methods.

Method	Description
boolean hasMoreElements()	This method tests Enumeration has any more elements.
element nextElement()	This returns the next element that is available in Enumeration.

Understanding Collection Interface

- The root interface in the *collection hierarchy*.
- A collection represents a group of objects, known as its *elements*.
- Some collections allow duplicate elements and others do not.
- Some are ordered and others unordered.
- The JDK does not provide any *direct* implementations of this interface.

Important Collection Interface methods

Method Name	Description
int size()	Returns the number of elements in this collection
boolean isEmpty()	Returns true if this collection contains no elements
int hashCode()	Returns the hash code value for this collection.
void clear()	Removes all of the elements from this collection. (Collection will be empty)
boolean contains(Object o)	Returns true if this collection contains the specified element
boolean containsAll(Collection c)	Returns true if this collection contains all of the elements in the specified collection.
boolean add(Object o)	Returns true if this collection changed as a result of the call. Returns false if this collection does not permit duplicates and already contains the specified element.

Method Name	Description
boolean addAll(Collection c)	Adds all of the elements in the specified collection to this collection
boolean remove(Object o)	true if an element was removed as a result of this call.
boolean removeAll(Object c)	Removes all of this collection's elements that are also contained in the specified collection
boolean retainAll(Collection c)	Retains only the elements in this collection that are contained in the specified collection

All the above mentioned methods are the most common general methods which can be applicable for any Collection object