Collections

Collection Programs

1. What is Collection?

▼ Ans

- A group of individual objects represented as a single entity is called has collection.. *Collection Framework* has been introduced in JDK 1.2 which holds all the collection classes and interface in it.
- Collection Interface implements Iterable Interface.

2. What is Framework ?

▼ Ans

- A framework is a set of classes and interface which provides a ready-made architecture.

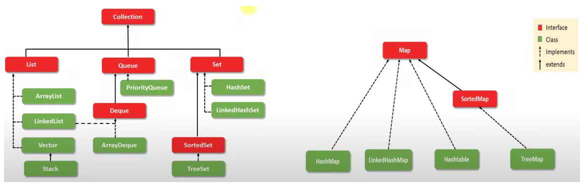
3. Why we use collection and framework ? What is the purpose of collection and frameworks ?

▼ Ans

- With respect to variables in order to store one data one container is needed, Hence to store n number of data n number of container is needed this becomes costly operation. Hence we go for Arrays where it is possible to store n number data in one container. but still the array size is fixed and array is homogenous and does not have inbuilt method to perform data operation so hence we go for collection frameworks.
- With respect to collection frameworks the size is dynamic and it can be homogenous as well as heterogenous and it contains inbuilt method to perform data operation.
- collection frameworks related classes and interface are present in java.util packages.

4. What are the Classes and Interface present in Collection ? What is collection Hiererchy ?

- And



5. What are methods present in collections ?

- Ans

  There are 11 abstract methods present in collection Interface:

  - add() :
    - method is used to insert the object into the Collections.
  - addAll() :
    - method used to fetches all the object from one Collection and insert it into the another Collection.
  - contains() :
    - method is used to check if an object is present within the collection or not.
  - containsAll() :
    - method is used to check if all the object are present from a particular nested Collection or not.
  - remove() :
    - method is used to delete a particular object from the Collection.
  - removeAll() :
    - method is used to delete all the object from a particular nested Collection.
  - clear() :
    - method is used to delete all object from the Collection.
  - isEmpty() :

- method is used to check if the Collection is empty or not.

▼ size() :

- method is used to define current number of object present in Collection.

▼ iterator() :

- method is used to returns an iterator over the elements in this Collection.

▼ toArray() :

- method is used to return an Array containing all the elements in this collection.

6. What is List ?

▼ Ans

- List is a child interface of C**ollection**. It is an ordered collection of objects in which we can store duplicate values as well as Null value. List Interface is implemented by ArrayList class, Vector class, LinkedList class.

7. What is ArrayList ?

▼ Ans

- ArrayList is a part of Collections and it is a child class of List Interface and present in java.util.ArrayList package since JDK1.2.
- ArrayList internally stores the object in the form of object Array.
- Default capacity is 10.
- incremental capacity is (current capacity*3/2 +1).
- Data-Structure Growable/Resizable Array.
- 3 Overloaded Constructors:
[1.ArrayList(), 2.ArrayList(int initialcapacity),   3.ArrayList(collection c).
- Duplicate Insertion is Allowed.
- Null Insertion is Allowed.
- Insertion Order is maintained.

- ArrayList impleaments List, RandomAccess, Cloneable and Serializable Interfaces.

▼ Where to use Array-List:

  - Array-List is good for data retrieval & search operation . Because time taken to search any data in entire List is same.
  - we use for better performance but it is not thread safe because of mult-threaded

▼ Where not to use Array-List:

  - If we try to add the data in between the list then all the existing data gets shifted to next position , so because of this shift operation the performance becomes slow

▼ Example Program

```
import java.util.*;
public class Arraylist1
{
  public static void main(String[] args)
  {
    ArrayList a1=new ArrayList();

    a1.add("Rahul");
    a1.add("Virat");
    a1.add("Rohit");
    a1.add("MSD");
    a1.add("Rishab");

    System.out.println(a1);
    System.out.println(a1.contains("Virat"));
    a1.remove("Rahul");
    System.out.println(a1);
    System.out.println(a1.size());
    System.out.println(a1.isEmpty());
    a1.clear();
    System.out.println(a1);
  }
}

OUTPUT:
[Rahul, Virat, Rohit, MSD, Rishab]
true
[Virat, Rohit, MSD, Rishab]
4
```

```
false
[]
```

8. What is Vector ?

▼ Ans

- Vector is a part of Collections and it is a child class of List Interface and present in java.util.Vector package since JDK1.0
- Vector internally stores the object in the form of object Array.
- Default capacity is 10.
- incremental capacity is (current capacity*2).
- Data-Structure Growable/Resizable Array.
- 4 Overloaded Constructors: [1.Vector(),2.Vector(int initialcapacity),3.Vector(int initialcapacity,int increamentalcapacity)] 4.Vector(collection c).
- Duplicate Insertion is Allowed.
- Null Insertion is Allowed.
- Insertion Order is maintained.
- Thread safe because of Single Threaded.
- Vector impleaments List, RandomAccess, Cloneable and Serializable Interfaces.

▼ Where to use Vector:

- Vector is good for data retrieval & search operation . Because time taken to search any data in entire List is same.
- For security we will use vector but slow in progress

▼ Where not to use Vector:

- If we try to add the data in between the list then all the existing data gets shifted to next position , so because of this shift operation the performance becomes slow

▼ Example Program

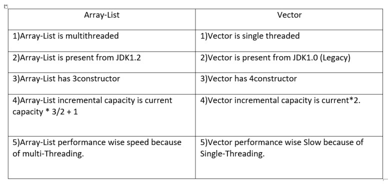
```
import java.util.*;
public class vector1
{
  public static void main(String[] args)
  {
    Vector a1=new Vector();

    a1.add("Rahul");
    a1.add("Virat");
    a1.add("Rohit");
    a1.add("MSD");
    a1.add("Rishab");

    System.out.println(a1);
    System.out.println(a1.contains("Virat"));
    a1.remove("Rahul");
    System.out.println(a1);
    System.out.println(a1.size());
    System.out.println(a1.isEmpty());
    a1.clear();
    System.out.println(a1);
  }
}
OUTPUT:
[Rahul, Virat, Rohit, MSD, Rishab]
true
[Virat, Rohit, MSD, Rishab]
4
false
[]
```

9. Difference Between Array-List and Vector ?

▼ Ans



Array-List	Vector
1)Array-List is multithreaded	1)Vector is single threaded
2)Array-List is present from JDK1.2	2)Vector is present from JDK1.0 (Legacy)
3)Array-List has 3constructor	3)Vector has 4constructor
4)Array-List incremental capacity is current capacity * 3/2 + 1	4)Vector incremental capacity is current*2.
5)Array-List performance wise speed because of multi-Threading.	5)Vector performance wise Slow because of Single-Threading.

10. What is the major problem we are going to face in Array-List and Vector ?

▼ Ans

- Shifting operation problem

- Whenever we add or delete in between two elements with respect to ArrayList and Vector Shifting operation takes place where it will leads to decrease in the performance hence the solution is LinkedList.

11. What is LinkedList ?

▼ Ans

- LinkedList is a part of Collections and it is a child class of List Interface and present in java.util.LinkedList package since JDK1.2.
- LinkedList internally stores the object in the form of node.
- LinkedList has no default capacity and incremental capacity.
- Data-Structure used is Doubly-linked.
- 2 Overloaded construtors. 1.LinkedList() and 2.LinkedList(collection).
- Duplicate Insertion is Allowed.
- Null Insertion is Allowed.
- Insertion Order is maintained.
- Not index based
- sort method is not present. because it is not index based
- we cannot use get() method.
- LinkedList implements List, Deque, Cloneable and Serializable Interfaces.

▼ Where to use LinkedList:

- LinkedList doesn't have any shift operation , hence it is suitable for insertion or removal of data in between.

▼ Where not use LinkedList:

- LinkedList is not suitable for any searching operation. because the control has to start through first node

▼ Example Program

```
import java.util.*;
public class Linkedlist1 {
  public static void main(String[] args)
  {
    LinkedList a1=new LinkedList();

    a1.add("Rahul");
    a1.add("Virat");
    a1.add("Rohit");
    a1.add("MSD");
    a1.add("Rishab");

    System.out.println(a1);
    System.out.println(a1.contains("Virat"));
    a1.remove("Rahul");
    System.out.println(a1);
    System.out.println(a1.size());
    System.out.println(a1.isEmpty());
    a1.clear();
    System.out.println(a1);
  }
}
OUTPUT:
[Rahul, Virat, Rohit, MSD, Rishab]
true
[Virat, Rohit, MSD, Rishab]
4
false
[]
```

12. What is Node ?

▼ Ans

- Node is a combination of data and the address.

13. What is Singly LinkedList and Doubly LinkedList ?

▼ Ans

▼ Singly LinkedList

- consists of object followed by address of next node.

▼ Doubly LinkedList

- consists of address of previous node followed by address of next node.
- the first node does not have previous node address and last node does not have next node address.

14. What is Generics ?

▼ Ans

- Generics is basically used to define a datatype for a collection.
- For a collection generics can be specified within angular braces. (<,>)
- For a collection if the generic are not specified then by default the generics will be object.
- with respect to list get() method is used in order to fetch the data.

▼ Example Program

```
import java.util.*;
public class Generic_example
{
  public static void main(String[] args)
  {
    ArrayList <String> al=new ArrayList();
    al.add("Java");
    al.add("SQL");
    al.add("Programimg");
    al.add("Web-Script");
    for(int i=0; i<al.size(); i++)
    {
      System.out.println(al.get(i));
    }
  }
}
OUTPUT:
Java
SQL
Programimg
Web-Script
```

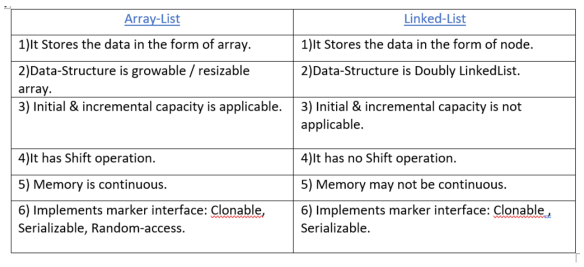
15. Why we use Generics ?

▼ Ans

- Generics are used to make Collection as homogeneous.

16. Difference Between ArrayList and LinkedList ?

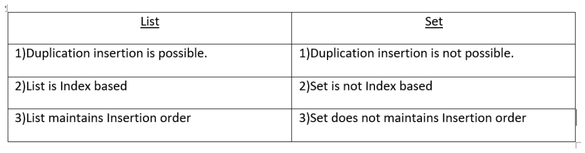
▼ Ans



Array-List	Linked-List
1)It Stores the data in the form of array.	1)It Stores the data in the form of node.
2)Data-Structure is growable / resizable array.	2)Data-Structure is Doubly LinkedList.
3) Initial & incremental capacity is applicable.	3) Initial & incremental capacity is not applicable.
4)It has Shift operation.	4)It has no Shift operation.
5) Memory is continuous.	5) Memory may not be continuous.
6) Implements marker interface: Clonable, Serializable, Random-access.	6) Implements marker interface: Clonable , Serializable.

17. Difference Between List and Set?

▼ Ans



List	Set
1)Duplication insertion is possible.	1)Duplication insertion is not possible.
2)List is Index based	2)Set is not Index based
3)List maintains Insertion order	3)Set does not maintains Insertion order

18. What is Set ?

▼ Ans

- Set is a child interface of C**ollection**. It is an unordered collection of objects in which we cannot store duplicate values and we can only store one null value. List Interface is implemented by HashSet class, LinkedHashSet class, TreeSet class.

19. What is HashSet ?

▼ Ans

- HashSet is a part of Collections and it is a child class of Set Interface and present in java.util.HashSet package since JDK1.2.
- HashSet internally stores the object in the form of Hashtable+Singly Linkedlist.
- Default capacity is 16.
- Fill ratio or load factor is 75%.
- There are 4 overloaded constructors
  HashSet(int initialCapacity)
  HashSet ()
  HashSet (Collection c)
  HashSet (int initialCapacity, float fillratio)

- Duplicate Insertion is not Allowed.
- Only One Null Insertion is Allowed.
- Insertion Order is not maintained.
- Implements Set, Clonable , Serializable Interfaces.
- Search operation works based on hash code

▼ Example program

```
import java.util.*;
public class Hashset1
{
  public static void main(String[] args)
  {
    HashSet hs=new HashSet();

    hs.add("Rahul");
    hs.add("Virat");
    hs.add("Rohit");
    hs.add("MSD");
    hs.add("Rishab");

    System.out.println(hs);
    System.out.println(hs.contains("Virat"));
    hs.remove("Rahul");
    System.out.println(hs);
    System.out.println(hs.size());
    System.out.println(hs.isEmpty());
    hs.clear();
    System.out.println(hs);
  }
}
OUTPUT:
[Rahul, Rohit, Rishab, Virat, MSD]
true
[Rohit, Rishab, Virat, MSD]
4
false
[]
```

20. What is LinkedHashSet ?

▼ Ans

- LinkedHashSet is a part of Collections and it is a child class of HashSet class and present in java.util.LinkedHashSet package since JDK1.4.

- LinkedHashSet internally stores the object in the form of Hashtable+7Doubly Linkedlist
- Default capacity is 16.
- Fill ratio or load factor is 75%.
- There are 4 overloaded constructors
  Linked HashSet(int initialCapacity)
  Linked HashSet ()
  Linked HashSet (Collection c)
  Linked HashSet (int initialCapacity, float fillratio)
- Duplicate Insertion is not Allowed.
- Only One Null Insertion is Allowed.
- Insertion Order is maintained.
- Extends HashSet and implements Set, Cloneable, Serializable Interfaces.

▼ Example program

```
import java.util.*;
public class Linkedhashset2 {
  public static void main(String[] args)
  {
    LinkedHashSet hs=new LinkedHashSet();

    hs.add("Rahul");
    hs.add("Virat");
    hs.add("Rohit");
    hs.add("MSD");
    hs.add("Rishab");

    System.out.println(hs);
    System.out.println(hs.contains("Virat"));
    hs.remove("Rahul");
    System.out.println(hs);
    System.out.println(hs.size());
    System.out.println(hs.isEmpty());
    hs.clear();
    System.out.println(hs);
  }
}
OUTPUT:
[Rahul, Virat, Rohit, MSD, Rishab]
true
[Virat, Rohit, MSD, Rishab]
```

```
4
false
[]
```

21. What is TreeSet ?

▼ Ans

- TreeSet is a part of Collections and it is a child class of Set Interface and present in java.util.TreeSet package since JDK1.2.
- TreeSet internally stores the data in the form of node.
- Data-Structure used is Balanced-Binary Tree.
- TreeSet is mainly used for sorting purpose. TreeSet follows default natural sorting order(Ascending order).
- Treeset must be homogeneous. If TreeSet is Heterogeneous we will get classcastexception.
- TreeSet does not accepts Single Null value. If Null is added we will get nullpointerexception.
- Duplication is not allowed.
- Data will be sorted in Ascending order.
- TreeSet will follow Inorder.
- TreeSet either uses Comparable or Comparator interface
- Implements NavigableSet, Clonable , Serializable Interfaces.
- NavigableSet interface extends sortedSet and sortedSet extends Set.
- It has a one Constructor that will takes Comparator Object. It is used for sorting purpose.

▼ Comparable or Comparator interface means:

- Comparable is used for default natural sorting.
- Comparator is used for Custom sorting

▼ Example Program

```
import java.util.*;
public class Treeset1 {
  public static void main(String[] args)
  {
    TreeSet f1=new TreeSet();

    f1.add(10);
    f1.add(7);
    f1.add(53);
    f1.add(100);
    f1.add(32);
    f1.add(2);
    System.out.println(f1);
  }
}
OUTPUT:
[2, 7, 10, 32, 53, 100]
```

22. What are Hash-Based Collection ?

▼ Ans

1. HashSet
2. LinkedHashSet
3. HashMap
4. LinkedHashMap
5. HashTable

23. What is Map ?

▼ Ans

- Map is a Collection or Set of entries. Map is a separate Interface which doesn't inherit Collection interface and present in java.util package since JDK1.2 . Map Interface is implemented by HashMap class, LinkedHashMap class, TreeMap class.
- Where we use Map: example: RollNumer-Student, Eid-Employee
- Where we use Map: example: Facebook—>Key is Username, values is Password

- Map Interface has nested Interface i,e Entry Interface. Inside Entry Interface 2 method are there 1)getKey(); 2)getValue();
- 1) getKey() means it will get particular Key.
- 2) getvalue means It will get particular Values.

```
Interface Map
{
    Interafce Entry
     {
         getKey();
         getValue();
     }
}
```

24. What is Entry ?

▼ Ans

- Entry is basically a data which is available in the form of “key” and “value” pair. where key must be unique value can be duplicated.
- “Key” and “Value” can be null also

25. What are methods present in Map?

▼ Ans

abstract method present in Map Interface

▼ put() :

- method is used to insert an entry to the Map.

▼ get() :

- method is used to fetch the corresponding value based on the key.
- For key corresponding value not there, we get Null.

▼ containsKey() :

- method used to check, if a particular key present in the Map or not.

▼ containsValue() :

- method is check, if a particular value is present in the Map or not.

▼ size() :

- method is used to define the current number of entries present in the Map.

▼ isEmpty() :

- method is used to check, if a map is empty or not.

▼ remove() :

- method is used to delete a particular entry from Map from based on key.

▼ clear() :

- method is used to delete all the entries from the Map.

▼ entrySet() :

- method is used to fetch all the key and value pair from the Map.
- return type or datatype is Map.entry

▼ keySet() :

- method is used to fetch all the keys from Map.

▼ values() :

- method is used to fetch all values from the Map.

▼ putAll() :

- method is used to **fetche**s all the entries from one Map and put it into another Map.

▼ Example Program

```
import java.util.*;
public class Map1
{
  public static void main(String[] args)
  {
    HashMap <Integer,String> hm=new HashMap <Integer,String>();

    hm.put(7,"MSD");
    hm.put(18,"Virat");
    hm.put(45,"Rohith");
    hm.put(null,null);
    hm.put(45,"Raina");
    hm.put(1,"Virat");

    System.out.println(hm.get(1));
    System.out.println(hm);
    System.out.println(hm.containsKey(1));
    System.out.println(hm.containsValue("Virat"));
    System.out.println(hm.size());
  }
}
OUTPUT:
Virat
{null=null, 1=Virat, 18=Virat, 7=MSD, 45=Raina}
true
true
5
```

26. What is Iterator ? uses

▼ Ans

- Iterator is an Interface present in java.util package since JDK1.2 . It is used to iterate any collection like Set and List. Iterator is not index based. Iterator can iterate only in forward direction.
- Iterator is Cursors.
- Iterators are used in the Collections to retrieve elements one by one.

▼ hasNext() :

- method is used to check if the next element is present or not.

▼ next() :

- method is used to move the cursor one position forward.

27. What is ListIterator ? uses

▼ Ans

- ListIterator is a child Interface of Iterator. ListIterator can be used to iterate only List but not Set. It can iterate both in forward as well as backward direction.
- Why it is not possible to iterate in set using listiterator.

▼ hasPrevious() :

- method is used to check, if the previous element is present or not.

▼ previous() :

- method is used to move the cursor one position back.

28. What is HashMap ?

▼ Ans

- HashMap is a map based collection which is used to store entries , and it is a child class of Map interface. and present in java.util.HashMap package since JDK1.2.
- It is also a hash based collection , hence data structure used is Hashtable.
- Default capacity is 16.
- Fill ratio or load factor is 75%.
- There are 4 overloaded constructors
  HashMap(int initialCapacity)
  HashMap ()
  HashMap (Collection c)
  HashMap (int initialCapacity, float fillratio)
- Only One Null Insertion is Allowed.
- HashMap can store heterogeneous data
- Implements Map, Cloneable, Serializable Interfaces.

▼ Example Program

29. What is LinkedHashMap ?

▼ Ans

- LinkedHashMap is a map based collection which is used to store entries , and it is a child class of HashMap. and present in java.util.LinkedHashMap package since JDK1.4.
- It is also a hash based collection , hence data structure used is Hashtable.
- Default capacity is 16.
- Fill ratio or load factor is 75%.
- There are 4 overloaded constructors
  LinkedHashMap (int initialCapacity)

LinkedHashMap ()
LinkedHashMap (Collection c)
LinkedHashMap (int initialCapacity, float fillratio)

- Only One Null Insertion is Allowed.
- LinkedHashMap can store heterogeneous data.
- Extends HashMap Implements Map Interfaces.

▼ Example Program

30. What is Hashtable ?

▼ Ans

- Hashtable is a map based collection which is used to store entries , and it is a child class of Map Interface. and present in java.util.HashSet package since JDK1.0.(Legacy)
- It is also a hash based collection , hence data structure used is Hashtable.
- It is a single threaded
- Default capacity is 1.
- Fill ratio or load factor is 75%.
- There are 4 overloaded constructors
  Hashtable (int initialCapacity)
  Hashtable ()
  Hashtable (Collection c)
  Hashtable (int initialCapacity, float fillratio)
- Not Even a single Null Insertion is not Allowed.
- Hashtable is slower than HashMap. Because it is Single threaded.
- Extends Dictionary class Implements Map, Cloneable, Serializable Interfaces.

▼ Example Program

31. What is TreeMap?

▼ Ans

➢ It is one of the implementation class of Map interface.
➢ Present since JDK 1.2.
➢ TreeMap is used mainly for sorting data based on key.
➢ TreeMap implements default natural sorting order on the key using Comparable interface.
➢ For custom sorting we use Comparator.
➢ It cannot store even a single null key.
➢ TreeMap is homogeneous.
➢ NullPointerException , when we try to add null.

- TreeMap will Sort based on Key only.
- It has a one Constructor that will takes Comparator Object. It is used for sorting purpose.
- Implements NavigableMap, Cloneable, Serializable Interfaces.
- NavigableMap Interface Extends SortedMap Interface.
- SortedMap Extends Map Interface.

32. Advantages of Collection?

▼ Ans

- Collection is dynamic in size.
- Collection is heterogeneous as well as hemogeneous.
- Collection has many Inbuilt utility methods.

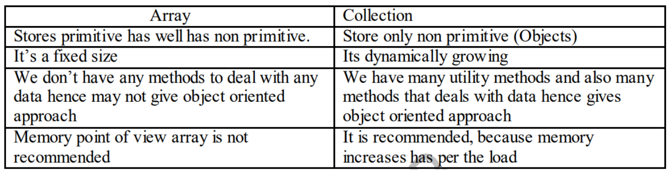
33. Dis-Advantages of Collection?

▼ Ans

1. It must cast to correct type.
2. It can't be done compile-time type checking.

34. Difference between Array and Collection ?

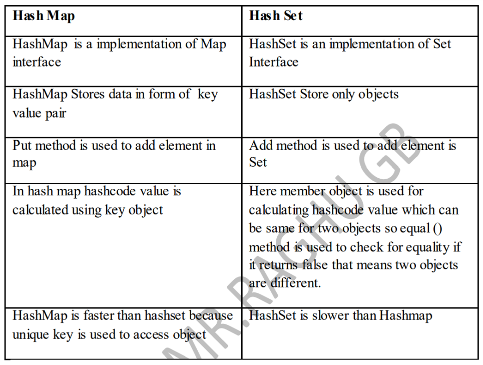
▼ Ans



Array	Collection
Stores primitive has well has non primitive.	Store only non primitive (Objects)
It's a fixed size	Its dynamically growing
We don't have any methods to deal with any data hence may not give object oriented approach	We have many utility methods and also many methods that deals with data hence gives object oriented approach
Memory point of view array is not recommended	It is recommended, because memory increases has per the load

35. Difference between HashMap and HashSet?

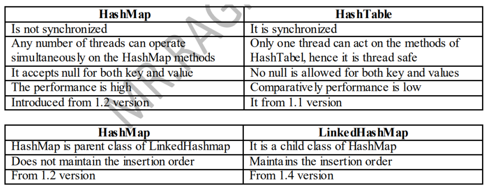
▼ Ans



Hash Map	**Hash Set**
HashMap is a implementation of Map interface	HashSet is an implementation of Set Interface
HashMap Stores data in form of key value pair	HashSet Store only objects
Put method is used to add element in map	Add method is used to add element is Set
In hash map hashcode value is calculated using key object	Here member object is used for calculating hashcode value which can be same for two objects so equal () method is used to check for equality if it returns false that means two objects are different.
HashMap is faster than hashset because unique key is used to access object	HashSet is slower than Hashmap

36. Difference between HashMap with HashTable and HashMap with LinkedHashMap ?

▼ Ans

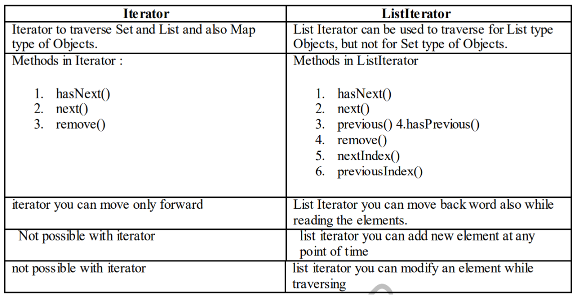


HashMap	**HashTable**
Is not synchronized	It is synchronized
Any number of threads can operate simultaneously on the HashMap methods	Only one thread can act on the methods of HashTabel, hence it is thread safe
It accepts null for both key and value	No null is allowed for both key and values
The performance is high	Comparatively performance is low
Introduced from 1.2 version	It from 1.1 version

HashMap	**LinkedHashMap**
HashMap is parent class of LinkedHashmap	It is a child class of HashMap
Does not maintain the insertion order	Maintains the insertion order
From 1.2 version	From 1.4 version

37. Difference between Iterator and ListIterator?

▼ Ans



Iterator	**ListIterator**
Iterator to traverse Set and List and also Map type of Objects.	List Iterator can be used to traverse for List type Objects, but not for Set type of Objects.
Methods in Iterator : 1. hasNext() 2. next() 3. remove()	Methods in ListIterator 1. hasNext() 2. next() 3. previous() 4.hasPrevious() 4. remove() 5. nextIndex() 6. previousIndex()
iterator you can move only forward	List Iterator you can move back word also while reading the elements.
Not possible with iterator	list iterator you can add new element at any point of time
not possible with iterator	list iterator you can modify an element while traversing

38. What is synchronizetion ?

▼ Ans

- Synchronization is the capability to control the access of multiple threads to any shared resource.

39. How can we Sort a Collection ?

▼ Ans

- List can be sorted using Comparable and Comparator.
- Set can be sorted using TreeSet.
- Map can be sorted using TreeMap.

40. Why we use Queue ?

▼ Ans

If we want do priority task then we will queue and it is used in O.S (real time example)

41. What is meaning growable or resizeble ?

▼ Ans

- means size of collection will change according to user.

42. Tree-Set follows which order ?

▼ Ans

- Inorder

43. What is comparable and comparator?

▼ Ans

1. We can campare collection objects in 2ways and

▼ Comparable Interface

1. CompareTo() method is used for sorting purpose. here in compareTo() mrthod we are passing only one parameterized object so comparison takes place directly on current object property with previous object property.
2. In this way comparable will sort the element by overriding compareTo() method. but that class should implements Comparable interface.

3. If we have access to the class, we go with comparable.
4. We can define default sorting order using Comaparable.

▼ Program

```
//comparable
public class Employee implements Comparable
{
  int id;
  String name;
  double sal;
  public Employee(int id,String name,double sal)
  {
    this.id=id;
    this.name=name;
    this.sal=sal;
  }
  @Override
  public String toString()
  {
    return this.id+" "+this.name+" "+this.sal;
  }

  public static void main(String[] args)
  {
    Scanner sc=new Scanner(System.in);
    ArrayList al=new ArrayList();
    while(true)
    {
      int id=sc.nextInt();
      String name=sc.next();
      double sal=sc.nextDouble();
      Employee emp=new Employee(id,name,sal);
      al.add(emp);
      System.out.println("More Elements /Data?");
      String s=sc.next();
      if(s.equalsIgnoreCase("no"))
        break;
    }
///Original form
    ListIterator itr=al.listIterator(al.size());
    while(itr.hasPrevious())
    {
      System.out.println(itr.previous());
    }
    System.out.println("-----------");
////sorted form
    Collections.sort(al);
    ListIterator tr=al.listIterator();
    while(tr.hasNext())
    {
      System.out.println(tr.next());
    }
    sc.close();
  }
  @Override
  public int compareTo(Object o)
  {
    Employee emp=(Employee)o;
    return -((Double)this.sal).compareTo(emp.sal);
  }
}
```

▼ Comparator interface

1. compare() method is used for sorting purpose. here in compare() method we are passing two object so comparison takes place based on object address not hashCode value. so we need to change comparison implementation so that we are using equals() and by overriding equals method we can change the implementation.
2. In this way Comparator will sort the elements by overriding compare() and equals() methods.
3. If we don't have access to the class, we go with the comparator.
4. We can define Customized sorting order by giving our own sorting.

▼ Program

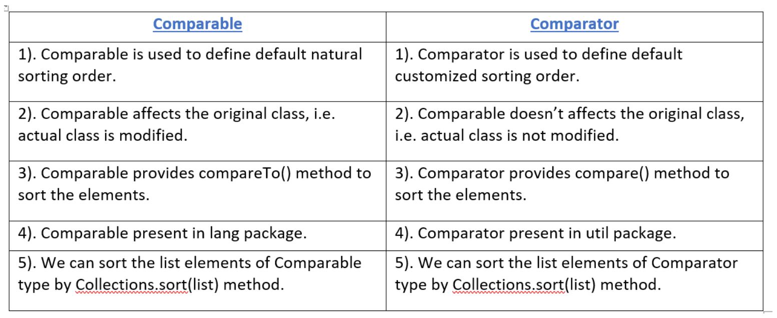
```
//comparator
public class Employee2
{
  public static void main(String[] args) {
    Scanner sc=new Scanner(System.in);
    Comparator com=new Comparator()
    {
      @Override
      public int compare(Object o1,Object o2)
      {
        Employee e1=(Employee) o1;
        Employee e2=(Employee) o2;
        return ((Double)e1.sal).compareTo(e2.sal);
      }
    };
    ArrayList al=new ArrayList();
    while(true)
    {
      int id=sc.nextInt();
```

```
        String name=sc.next();
        double sal=sc.nextDouble();
        Employee emp=new Employee(id,name,sal);
        al.add(emp);
        System.out.println("More object are There?");
        String s=sc.next();
        if(s.equalsIgnoreCase("no"))
          break;
      }
      Collections.sort(al,com);
      ListIterator itr=al.listIterator();
      while(itr.hasNext())
      {
        System.out.println(itr.next());
      }
    }
  }
```

2. By using Comparable and Comparator, we can sort the collection elements.
3. Comparable interface is present in lang package and it contains only 1 abstract method i.e. int compareTo(); It return 0,1,-1.
4. Comparable interface is present in util package and it contains two abstract method i.e. int compare(,); and boolean equals();
5. We can sort elements using comparable and comparator in only Array-List and Vector. We connect use Comparable and Comparator in hash-based because data structure is in node form. this is the main advantages of comparable and comparator.
6. Collections.sort(list) —> List.sort(list)—>Comaprable —> comparaTo().
7. Collection.sort(list,Comparator object)—>List.sort(list,Com)—>Comparator—>compare()—>compareTo().

44. Difference between Comparable and Comparator ?

▼ Ans



Comparable	Comparator
1). Comparable is used to define default natural sorting order.	1). Comparator is used to define default customized sorting order.
2). Comparable affects the original class, i.e. actual class is modified.	2). Comparable doesn't affects the original class, i.e. actual class is not modified.
3). Comparable provides compareTo() method to sort the elements.	3). Comparator provides compare() method to sort the elements.
4). Comparable present in lang package.	4). Comparator present in util package.
5). We can sort the list elements of Comparable type by Collections.sort(list) method.	5). We can sort the list elements of Comparator type by Collections.sort(list) method.

45. What is get() method ?

- ▼ Ans
  - get() method is in List Interface. Used to get the elements present in this list at a given specific index.
  - We cannot use get() method in Set because get() is not present in Set Interface.

46. What are legacy in java?

- ▼ Ans
  - Dictionary is a legacy class present util package.
  - Vector is a legacy collection present util package.
  - Enumeration is legacy cursor to iterate.
  - Hashtable is legacy class present util package.

47. What are advantage and disadvantages of array ?

- ▼ Ans
  - ▼ Advantages
    - In an array, accessing an element is very easy by using the index number
    - The search process can be applied to an array easily
    - 2D Array is used to represent matrices
    - For any reason a user wishes to store multiple values of similar type then the Array can be used and utilized efficiently
  - ▼ Dis-Advantages

- **Array size is fixed:**
  The array is static, which means its size is always fixed. The memory which is allocated to it cannot be increased or decreased.

48. What happens if you don’t give a Size to an array ?

▼ Ans

Complie Time Error.

49. Dis-advantages OF LinkedList?

▼ Ans

- Searching an element is costly and requires O(n) time complexity.

50. How do you sort a List?

▼ Ans

- By using Comparable or Comparator.

51. What is the purpose of HASHBASED Collection ?

▼ Ans

- It maintains Uniqueness and if we want get Unique data we should go with Hash-based Collection.

52. Write the dis-advantage of ArrayList ?

▼ Ans

- Addition or deletion of data from the middle is time consuming as data needs to be shifted to update the list.
- Resizing of an arraylist when it reaches it's capacity with it initial capacity which is 10 is a costlier process as the elements will be copied from old to new space with 50% more capacity.

53. Can we possible to write null insertion inside TreeSet and HashSet ?

▼ Ans

- TreeSet does not accepts Single Null value. If Null is added we will get nullpointerexception.
- Only One Null Insertion is Allowed in HashSet

54. What is the Data Structure of ArrayList, LinkedList, Vector and TreeSet ?

▼ Ans

- Arraylist—→ Growable or resizeble
- Linkedlist—→ Doubly Linkedlist
- Vector—→ Growable or resizeble
- TreeSet—→ Balanced Binary Tree

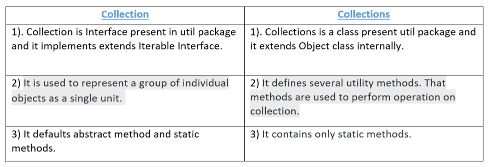
55. TreeSet in the nature of heterogeneous or homogeneous ?

▼ Ans

- Treeset must be homogeneous. If TreeSet is Heterogeneous we will get classcastexception.

56. Difference between collection and collections ?

▼ Ans



Collection	Collections
1). Collection is Interface present in util package and it implements extends Iterable Interface.	1). Collections is a class present util package and it extends Object class internally.
2) It is used to represent a group of individual objects as a single unit.	2) It defines several utility methods. That methods are used to perform operation on collection.
3) It defaults abstract method and static methods.	3) It contains only static methods.

57. Define pop(), peep(), push(), search() ?

▼ Ans

- **push():** is a method in Java is **used** to insert an element into the stack
- **pop():** is a method in Java is **used** to remove an element from the stack
- **top():** is a method in Java is **used** Returns the top element of the stack.
- **isEmpty():** is a method in Java is **used** returns true is stack is empty else false
- **size():** is a method in Java is **used** returns the size of stack
- search(): is a method in Java is **used to search for an element in the stack and get its distance from the top.**

58. Define Stack ?

▼ Ans

- The stack is a linear data structure that is used to store the collection of objects. It is based on Last-In-First-Out (LIFO).

59. Can we possible list collection to convert heterogeneous to homogeneous ?

▼ Ans

- If we use generics. It is possible

60. Explain the data-structure called growable or resizable array ?

▼ Ans

61. What are homogeneous collection in java ?

▼ Ans

62. Explain hashing algorithms ?

▼ Ans

63. Where do you find implementation for iterator method ?

▼ Ans

64. Difference between Array and ArrayList ?

▼ Ans

65. Where we prefer array over ArrayList ?

▼ Ans

65. Why we are not considering Map as a collection ?

▼ Ans