Collections and Generics Interview Questions

1. What is Collection?
   It is a root interface which represents **group** of individual objects as a single entity (object).

2. What are collection properties?
   a) Duplicate
   b) Ordered
   c) Null values
   d) Sorted
   e) Synchronized

3. What are the differences between List and Set?

Collection property	Java.util.List	Java.util.Set
Duplicate	Yes	Never
Ordered	Yes	Depends on implementation class
Null values	Accepts duplicate null values.	0 or 1
Sorted	None	Some
Synchronized	Some	None

4. What is the difference between ArrayList and Vector?

ArrayList	Vector
It is non-legacy class hence by default it is non synchronized means non-thread safe.	It is legacy class hence by default it is synchronized means thread safe.

5. How to get synchronized version of ArrayList?
   By using public static List **synchroniedList**(List) method from Collections class.

6. What are legacy classes in java.util package?
   Vector, Statck, Hashtable, Properties, Dictionary.

7. When to use ArrayList and when to use LinkedList?
   In case of frequent search operations use ArrayList because it is a subclass of RandomAccess marker interface.
   In case of frequent insert/delete operations use LinkedList because it internally uses doubly linked list mechanism.

8. Which collection is preferred in case of frequent search operations?
   Both ArrayList and Vector classes are subclasses of RandomAccess, hence the time complexity is O(1).

9. Which collection is preferred in case of frequent insert operations?

The LinkedList internally uses doubly linked list mechanism hence the time complexity of inset/delete operations is O(1) irrespective of position.

10. What are differences between java.lang.Comparable and java.util.Comparator interfaces?

Comparable<<interface>>	**Comparator<<interface>>**
It belongs to java.lang package	It belongs to java.util package
Our class must be subclass of Comparable interface.	We can provide sorting techniques independently hence our class need not be inherited from Comparator interface.
We can provide only one sorting technique.	**We can provide multiple sorting techniques.**

11. What are differences among Enumeration, Iterator, and ListIterator interfaces?

Property	Enumeration	Iterator	ListIterator
Is legacy?	Yes	No	No
It is applicable for	Only legacy classes	All Collection implementation classes (both legacy & non-legacy)	Only for List implementation classes.
Navigation	Single direction (forward)	Single direction (forward)	Both directions (bi-directional).
How to get	Using **elements()** method	Using **iterator()** method	Using **listIterator()** method
Operations	Only read	Read and remove	Read, remove, modify and add
Methods	hasMoreElements(), nextElement()	hasNext() next() remove()	9 methods.

12. What will happen if we try to insert duplicate object into Set?
Neither generates compilation error nor throws exception rather returns **false**.

13. Difference between HashSet and TreeSet?

HashSet	**TreeSet**
It internally uses hashing mechanism hence elements are **unordered**.	It internally uses sorting mechanism hence elements are in sorting order.
Implementing equals() and hashCode() methods are required.	Implementing equals() and hashCode() methods are not required.

14. What is Map?
This is the root interface for all key and value based maps.
This is used to represent a group of **entries** as key and value pairs.

15. What are differences between Hashtable and HashMap?

HashMap	**Hashtable**

By default, it is not synchronized (non thread-safe)	By default, it is synchronized (thread-safe).
Performance is high	Performance is low
Introduced in 1.2 version	Legacy, introduced in 1.0 version.
Both key and values can be null.	Neither key nor value is null.

16. How to use user-defined class as a key in Hashtable or HashMap?

Ans) Implement both hashCode() and equals() methods so that we can use such class as a key.

Example:

```
import java.util.Hashtable;
class Book{
    private int id;
    private String name;
    public Book(int id, String name) {  … }
    @Override
    public int hashCode() { … }
    @Override
    public boolean equals(Object obj) { … }
}
public class HashtableDemo{
    public static void main(String[] args) {
            Hashtable ht = new Hashtable();

            Book b1 = new Book(1,"abc");
            Book b2 = new Book(2, "xyz");
            Book b3 = new Book (2, "xyz"); //duplicate

            ht.put(b1, new Integer(10));
            ht.put(b2, new Integer(20));
            ht.put(b3, new Integer(30));

            System.out.println(ht.size()); //2
    }
}
```

17. What is **ConcurrentHashMap?**

It allows concurrent modifications from **several threads** without blocking hence enhances performance. To better visualize the ConcurrentHashMap, let it consider as a group of HashMaps. In ConcurrentHashMap, the **difference lies in internal structure to store these key-value pairs**. ConcurrentHashMap has an addition concept of **segments**. It will be easier to understand it you think of one segment equal to one HashMap [conceptually]. A ConcurrentHashMap is divided into number of segments [default 16] on initialization. ConcurrentHashMap allows similar number (16) of threads to access these segments concurrently so that each thread work on a specific segment during high concurrency. This way, if when your key-value pair is stored in segment 10; code does

not need to block other 15 segments additionally. This structure provides a very high level of concurrency. In other words, ConcurrentHashMap uses a multitude of locks, each lock controls one segment of the map. When setting data in a particular segment, the lock for that segment is obtained.

18. What is TreeMap?
    The entries in TreeMap are in sorting order based on key.

19. What is WeakHashMap?
    This is exactly same as HashMap except the following differences.
    In case of HashMap, an object is not eligible for garbage collection even though it doesn't have any external references while it is associated with HashMap i.e. HashMap dominates Garbage Collection.
    In case of WeakHashMap, if an object doesn't have any external references then it is always eligible for Garbage Collection, even though it is associated with WeakHashMap i.e. Garbage Collection dominates WeakHashMap.

Example:

```
import java.util.*;
class Temp{
    public String toString(){
            return "temp";
    }
    Protected void finalize(){
            System.out.println("finalize method");
    }
}
class WeakHashMapDemo{
    public static void main(String[] args) throws Exception{
            //Map m=new HashMap();
            Map m=new WeakHashMap();
            Temp t=new Temp();
            m.put(t, "aspire");
            System.out.println(m);
            t=null;
            System.gc();
            Thread.sleep(5000);
            System.out.println(m);
    }
}
```

Output:

Map m = new HashMap()	Map m = new WeakHashMap()
{temp=aspire} {temp=aspire}	{temp=aspire} finalize method {}

20. How to compare two collections such as two stacks or two arraylists?

Ans) Using **containsAll(Collection)** method

Example:

```
Stack s1 = new Stack();
S1.add(1);  S1.add(2);  S1.add(3);

Stack s2 = new Stack();
S2.add(1);  S2.add(2);  S2.add(3);

sop(s1.containsAll(s2)); //true
```