1. What are the differences of LinkedList and ArrayList.

ArrayList:

1.ArrayList internally uses Dynamic array to store the elements.
2.ArrayList implements List only.
3.ArrayList is better for storing and accessing data.
4.Manipulation is slow because it internally uses array.If any element is removed from the array, all the bits are shifted in memory.

LinkedList:

1.LinkedList internally uses doubly linked ist to store the elements.
2.LinkedList implements List and Deque interfaces.
3.LinkedList is better for manipulating data.
4.Manipulation is faster because it uses doubly linkedlist, so no bit shifting is required in memory.

2. How to sort ArrayList of strings alphabetically in java?

```
Collections.sort(list, new Comparator<String>() { @Override public int compare(String s1, String s2) { return s1.compareToIgnoreCase(s2); } });
```

```
        Collections.sort(list,new Comparator<String>{
                @override
                Public int compare(String s1,String s2){
                        returns1.compareToIgnoreCase(s2);
                        }
        }
)
```

Or if using JAVA8

```
list.sort(String::compareToIgnoreCase);
```

```
List.sort(String::compareToIgnoreCase);
```

3. How to convert a string array to ArrayList?

```
Public class StringArrayTest {
        Public static void main(String args[]){
  String [] words = {“ace” , “boom”, “dog”, “Monkey”,”donkey”};

List<String> wordslist = Arrays.asList(words);
```

```
for(String elist : wordslist){
system.out.println(elist);
}
}
}
```

The key here is : List<String> wordslist = Arrays.asList(words);

4. Which collection classes are synchronized or thread-safe ?
The collection classes that are thread-safe in Java are
1.vector. 2.stack. 3.properties. 4.hashtable
WHY? Here's the reason: Synchronization can be very expensive! You know, Vector and Hashtable are the two collections exist early in Java history, and they are designed for thread-safe from the start (if you have chance to look at their source code, you will see their methods are all synchronized!). However, they quickly expose poor performance in multi-threaded programs. As you may know, synchronization requires locks which always take time to monitor, and that reduces the performance. That's why the new collections (List, Set, Map, etc) provide no concurrency control at all to provide maximum performance in single-threaded applications.

5. What is the difference between peek(),poll() and remove() method of the Queue interface ?
peek(): This method returns the object at the top of the current queue without removing it. If the queue is empty, this method returns null .

poll(): The poll() method of queue interfaces returns the object at the top of the current queue and removes it. If the queue is empty, this method returns null.

Remove(): The remove() method of the queue interface returns the object at the top of the current and removes it. If the queue is empty, then it throws an "NoSuchElementFoundException" error.

6. How HashSet works internally in java?
 we create a HashSet, it internally creates a HashMap and if we insert an element into this HashSet using *add()* method, it actually call *put()* method on internally created HashMap object with element we have specified as it's key and

constant Object called **“PRESENT”** as it’s value. So we can say that **a Set achieves uniqueness internally through HashMap**.

7. What are the differences between iterator and enumeration.

Iterator:

1.In Iterator, we can read and remove element while traversing element in the collections.

2.It can be used with any class of the collection framework.

3.Only forward direction iterating is possible.

4.Any changes in the collection, such as removing element from the collection during a thread is iterating collection then it throw concurrent modification exception.

5.It has following methods −

*hasNext()

*next()

*remove().

Enumeration:

1.Using Enumeration, we can only read element during traversing element in the collections.

2.It can be used only with legacy class of the collection framework such as a Vector and HashTable.

3.Remove operations can not be performed using Enumeration.

4.Enumeration is Fail safe in nature. It doesn’t throw concurrent modification exception.

5. It has following methods −

*hasMoreElements()

*nextElement().

8. What is difference between fail-fast and fail-safe?

Fail-Fast:

1.Any changes in the collection, such as adding, removing and updating collection during a thread are iterating collection then Fail fast throw concurrent modification exception.

2.ArrayList and hashmap collection are the examples of fail-fast iterator.

3.It's work is on actual collection instead. So, this iterator doesn't require extra memory and time.

4.Iterators don't allow modifications of a collection while iterating over it.

Fail_Safe:

_____1.The fail-safe collection doesn't throw exception.

2.CopyOnWrite and concurrent modification are the examples of a fail-safe iterator.

3.It's working on a clone of the collection instead of actual collection. It is overhead in terms of time and memory.

4.Fail-Safe iterators allow modifications of a collection while iterating over it.

9. What is the importance of hashCode() and equals() methods? How they are used in Java?

Equals():

- The java equals() is a method of *lang.Object* class, and it is used to compare two objects.
- To compare two objects whether they are the same, it compares the values of both the object's attributes.
- By default, two objects will be the same only if stored in the same memory location.

Syntax:
Public boolean equals(Object obj)

Parameter:
obj: It takes the reference object as the parameter, with which we need to make the comparison.

Returns:
It returns the true if both the objects are the same, else returns false.

HashCode():

- A **hashcode** is an integer value associated with every object in Java, facilitating the hashing in hash tables.

- To get this hashcode value for an object, we can use the hashcode() method in Java. It is the means ***hashcode() method that returns the integer hashcode value of the given object***.
- Since this method is defined in the Object class, hence it is inherited by user-defined classes also.
- The hashcode() method returns the same hash value when called on two objects, which are equal according to the equals() method. And if the objects are unequal, it usually returns different hash values.

Syntax:

Public int hashcode()

Returns:

It returns the hash code value for the given objects.

10. What are the difference between List, Set and Map interfaces in java?

Set:

1.The **set** follows the unordered way and it found in java.util package.
2.Duplicate item will be ignored in Set.
3.it will not print in the final output.

List:

1.List contains the index-based methods to insert, update, delete, and search the elements.
2.It can have duplicate elements also. We can also store the null elements in the list.
3.List preserves the insertion order, it allows positional access and insertion of elements.

Map:

1.A map in java, not extends the Collection interface. It represents a group of special elements or objects.
2. Every map element or object contains key and value pair.

3. A map can’t contain duplicate keys and one key can refer to at most one value.

11. Difference between enumeration and iterator? give example.

Iterator Example:

```
package IteratorExample;

// Import the ArrayList class and the Iterator class
import java.util.ArrayList;
import java.util.Iterator;
public class IteratorEx {


  public static void main(String[] args) {

    // Make a collection
    ArrayList<String> cars = new ArrayList<String>();
    cars.add("Volvo");
    cars.add("BMW");
    cars.add("Ford");
    cars.add("Mazda");

    // Get the iterator
    Iterator<String> it = cars.iterator();

    // Print the first item
    System.out.println(it.next());
  }
}
```

Enumeration Example:

```
package EnumExample;

import java.util.Vector;
import java.util.Enumeration;
```

```
public class EnumEx {

  public static void main(String args[]) {
    Enumeration<String> days;
    Vector<String> dayNames = new Vector<String>();

    dayNames.add("Sunday");
    dayNames.add("Monday");
    dayNames.add("Tuesday");
    dayNames.add("Wednesday");
    dayNames.add("Thursday");
    dayNames.add("Friday");
    dayNames.add("Saturday");
    days = dayNames.elements();

    while (days.hasMoreElements()) {
      System.out.println(days.nextElement());
    }
  }
}
```

12. How TreeSet avoids duplicates and ensures sorted order.

TreeSet class in Java is part of Java's collections framework which implements the **NavigableSet Interface**, which provides functionalities to navigate through the SortedSet. The NavigableSet further extends the **SortedSet Interface**, which provides functionalities to keep the elements sorted.

As the TreeSet class implements a NavigableSet interface, it has features of both – the NavigableSet and the SortedSet.

The TreeSet sorts or orders the elements according to the natural ordering. All the Wrapper classes and the String class already implements Comparable interface. But in the case of custom objects, one has to implement Comparable or Comparator interfaces in the corresponding class, so that TreeSet can sort the objects as we desire.

13. What are the characteristics of TreeMap in java?

- Java TreeMap contains values based on the key. It implements the NavigableMap interface and extends AbstractMap class.
- Java TreeMap contains only unique elements.
- Java TreeMap cannot have a null key but can have multiple null values.
- Java TreeMap is non synchronized.
- Java TreeMap maintains ascending order.

14. What is stack ? what are the operations we can perform on stack?
Stack: Stack is collection of some elements arranged in a fashion such that every new element will be placed on top of the previous element.
we need to ensure the following operations.

1. Every new element will be placed on the top of the previous element.
2. Elements will be accessed one by one and in the reverse order of adding the elements means last added element will be pulled out first.
3. It will have the operations as
    1. Push : means adding new element to the array.
    2. Pop : means pulling out the last element of the array.
4. A pointer or indexer which keeps the index Location of the last element called as Top.
5. If stack is empty means no element on the stack then Top = -1
6. If stack have only one element then Top = 0 I.e. First index of the array.
7. If stack is full means every index of the array have element and array size is Not then Top = N-1
8. Before doing any pop operation ensure that Top != -1.
9. Before doing any push operation ensure that Top < N-1.
10. For every pop operation Top should be decrease.
11. For every push operation Top should be increase.

15. What is the time complexity of the following:- i) Fixed sized stack pop(),push(),peek() operations ii) Dynamic sized stack pop(),push(),peek() opeartions iii) Regular queue enqueue(), dequeue() , peek() operations iv) Priority queue enqueue(), dequeue() , peek() operations

i) Fixed sized stack pop(),push(),peek() operations - O(1)
ii) Dynamic sized stack pop(),push(),peek() opeartions- O(1).
iii) Regular queue enqueue(), dequeue() , peek() operations - O(1), peek()-O(n).
iv) Priority queue enqueue(), dequeue() , peek() operations - O(1), peek()-O(n).

16. What is queue? mention different types of queue? what are the operations we can perform on each queue type.
A queue is a data structure in which whatever comes first will go out first. It follows the FIFO (First-In-First-Out) policy. In Queue, the insertion is done from one end known as the rear end or the tail of the queue, whereas the deletion is done from another end known as the front end or the head of the queue. In other words, it can be defined as a list or a collection with a constraint that the insertion can be performed at one end called as the rear end or tail of the queue and deletion is performed on another end called as the front end or the head of the queue.

1.Linear(Simple) Queue:A simple queue is the most basic queue. In this queue, the enqueue operation takes place at the rear, while the dequeue operation takes place at the front.
Its applications are process scheduling, disk scheduling, memory management, IO buffer, pipes, call center phone systems, and interrupt handling.

2.CIrcular Queue: In this queue, the last node points to the first node and creates a circular connection. Thus, it allows us to insert an item at the first node of the queue when the last node is full and the first node is free.
It's also called a ring buffer.
It's used to switch on and off the lights of the traffic signal systems. Apart from that, it can be also used in place of a simple queue in all the applications mentioned above.

3.Priority Queue: A priority queue is a special kind of queue in which each item has a predefined priority of service. In this queue, the enqueue operation takes place at the rear in the order of arrival of the items, while the dequeue operation takes place at the front based on the priority of the items.

That is to say that an item with a high priority will be dequeued before an item with a low priority.

In the case, when two or more items have the same priority, then they'll be dequeued in the order of their arrival. Hence, it may or may not strictly follow the FIFO rule:

4.Double-Ended Queue(Deque):A deque is also a special type of queue. In this queue, the enqueue and dequeue operations take place at both front and rear. That means, we can insert an item at both the ends and can remove an item from both the ends. Thus, it may or may not adhere to the FIFO order:

It's used to save browsing history, perform undo operations, implement A-Steal job scheduling algorithm, or implement a stack or implement a simple queue.
Further, it has two special cases: input-restricted deque and output-restricted deque.

17. How do you implement Thread in java?
There are two ways to create a thread:

1. By extending Thread class
2. By implementing Runnable interface.

By Extending Thread class:

```
package CreateThread;

public class Multi extends Thread {

        public void run(){
        System.out.println("thread is running...");
        }
        public static void main(String args[]){
        Multi t1=new Multi();
        t1.start();
         }
                }
```

By implementing Runnable interface

```
package CreateThread;

public class Multi1 implements Runnable {
        public void run(){
        System.out.println("thread is running...");
        }

        public static void main(String args[]){
        Multi1 m1=new Multi1();
        Thread t1 =new Thread(m1);
        t1.start();
         }
        }
```

18. What is the difference between notify() and notifyAll() in java?

Notify: 1.In case of multiThreading notify() method sends the notification to only one thread among the multiple waiting threads which are waiting for lock.

2.As in case of notify the notification is sent to single thread among the multiple waiting threads so it is sure that which of those waiting thread is going to receive the lock.

3.In case of notify() method the risk of thread missing is high as notification is sent only single thread and if it misses that than no other thread would get notification and hence the lock.

4.Memory and CPU drain is less as compare to notifyAll as notification is sent to single one thread so performance is better as compare to notifyAll.

5.In case of the notify() method as only single one thread is in picture hence no concept of thread Interchangeable is possible.

NotifyAll():1. While notifyAll() methods in the same context sends the notification to all waiting threads instead of single one thread.

2.On other hand notifyAll sends notification to all waiting threads hence it is not clear which of the thread is going to receive the lock.

3.While in case of notifyAll as notification is to all the waiting threads and hence if any thread misses the notification, there are other threads to do the job.Hence risk is less.

4.On other hand as the cost of no notification is dropped and notification is sent to all waiting threads the memory and CPU drain is more as compare to notify and hence performance of notifyAll is lesser.

5.While we should go for notifyAll() if all your waiting threads are interchangeable (the order they wake up does not matter).

19. What is the difference between sleep() and wait() methods in java multithreading?
Wait:1Wait() method belongs to Object class .
2.Wait() releases the lock on an object .
3.Wait() can be called on object itself .
4.Wake up condition:until call notify(), notifyAll() from object.
5.Program can get spurious wakeups.

Sleep():1.Sleep() method belongs to Thread class.
2.It does not release lock on an object .
3.Sleep() can be called on thread .
4.Wake up condition:until at least time expire or call interrupt.
5.It will not get spurious wakeups.

20. Difference between livelock and deadlock in java.
Deadlock:Deadlock in Java is a condition where two or more threads are blocked forever, waiting for each other. This usually happens when multiple threads need the same locks but obtain them in different orders
Livelock:A thread often acts in response to the action of another thread. If the other thread's action is also a response to the action of another thread, then livelock may result. As with deadlock, livelocked threads are unable to make further progress.