Data Structures Interview Questions

- **What is the internal data structure used by ArrayList ?**
  Object[]
- **How does add and remove method of array list works ?**
  Add: Data will be added to object array based index. When ever the index is same or greater than the length of object array then it creates new array and copies the data. This work can be done by Arrays.copyOf() method
  Remove(): Remove method deletes the element in the given index and performs the left shift operation.
  To do this we use System.ArrayCopy();
- **Write the custom array list ?**
  First Create the class
  Initialize the Object [] in the constructor
  Implement Add method ->if (index>=ar.length){Arrays.copyOf}else{
  Ar[index]=obj;
  }
  Look at the program
- **What is the internal data Structure used by Hash Map ?**
  Node[]
  Put method implementation: 1. calculate the hashcode for the given key. Find the index-> Create the Node object by passing hashcode, key , value and address of next node
  Get(Key) implementation: calculate the hashcode for the given key, find out the index. And then compare the hashcode of the given and hashcode from the node present at the index. If it is matching compare the key from the given argument and key from the node object. If both are matching then get the value from Node object
- **What is collision in hashMap ?**
  H.put("A",123); -> A hashcode is 65->index: 65/16 =>4 -> Node Object (65,"A",123, null)
  H.put("B",124);-> B hashcode is 66->index:66/16-=>4->but at 4 already 1 node is present this situation is called hash collision. Incase of hash collision hash map follows linked list. SO create 1 more node at the same index (4)->Node Object(66,"B",124,null) Now the object is created, previous node will contain the address of this B node.
- **What is the internal implementation of hashMap ?**
  Node[]
  Write a program to insert unique employee objects in HashMap ?
  HashMap doesn't allow duplicate keys. In order to identify the duplicate key hashmap looks at the implementation of hashcode and equals method. In order not to store the duplicate employees we needs to override the equals and hashcode method and drive the unique employee condition based on emp id. Look at the program for reference
- **Write a program to sort the hashMap by keys ?**
  ->Convert the hashmap into set by calling the m.entrySet();
  ->convert the set into list or linked list
  ->Collections.sort(l) =>The generics of the list should implement the comparator or comparable or else Collections.sort(l, new Comparator())
  ->above step will give sorted list-> convert this into linked hashmap and store the data
  -> TreeMap m= new TreeMap(hashMap object)
- **What is the internal data structure used by Hash Set ?**
  HashMap:
  Method to add element in hashset ? Add():->the argument passed to the add method will be considered as Key and value is Object of object class every time.
- **Why hash set doesn't have get() method ?**
  Value present in the hashset is same for all the keys which is nothing but object of object class hence no need to have get method
- **What is concurrent modification exception ? And how java knew this ?**
  While iterating and perform the structural modification and calling the next or previous method we get concurrent modification exception.
  Java knew the structural modification based on mod count. When ever we add the data or delete the data mod count will be incremented and at the time of next or previous method mod count and expected mod count will be checked if both are not matching then concurrent modification exception will be thrown
- **Difference between synchronized vs concurrent ?**
  Synchronized ->entire object get locked by a thread
  Concurrent : lock will acquire at segment level or index level
- **In concurrent hash map how many threads can work at a time when it has default capacity**
  16 threads
- **How to remove element from linked list ? (deleting middle, head and last)**
  Middle element: start the process with head-> traverse till element to be deleted and hold the previous node value in previous variable then previous.next=temp.next
  Temp.next=null

  Head: temp=head
  head=head.next
  Temp.next=null
  Temp==null

  Last: preious.next=null
  Temp=null
- **What is the difference between Array Blocking Queue and Linked Blocking Queue?**
  - Arrayblocking queue size is fixed, where as linked blocking queue is not. Without providing the capacity we cannot create object for array blocking queue. We cannot add beyond the capacity to array blocking queue using add method. But we can use put method to perform blocking operations
  - If we create the Linked blocking queue without providing the size then the capacity of linked blocking queue is max size of integer data type. We can also perform blocking add and delete operations using put and take

method
- Generally in a Array blocking queue, if the capacity is full and try to add new element we get exception saying that queue is full where as Blocking queue if the capacity is full and if we are trying to add element using put method then put method goes to wait mode until some free space is created in the queue. Once the free space is available then it performs the operation
- Similarly, if the queue is empty and try to perform poll operation then we get null, where as in blocking queue if we perform take operation and if the queue is empty then it doesn't return null, it waits until the element is added to the queue and then performs delete operation

- **What is the difference between pop, peek ?**
  Stack follows LIFO
  Pop and peek are present in Stack
  Parent for Stack is Vector
  Stack is synchronized because vector is synchronized
  Pop -> delete the top of the stack (ar.length-1) and return the element
  Remove(ar.length-1) is nothing but pop method.

  Peek:=> return the top of the stack but doesn't delete
- **What happens if we call peek and pop on empty stack ?**
  Empty stack exception
- **What is the difference between poll, peek ?**
  These are present in Queue
  Queue is interface and follows FIFO
  Poll -> deletes the first element or head of the queue or deletes the zeroth index and returns it
  Peek : returns the 0th index value but doesn't delete it
- **What happens if we call poll and peek on empty Queue ?**
  It will return null
- **What is Priority Queue ?**
  It is a class which implements Queue interface and follows the natural sorting order.
  This internally uses the comparator to store the data inside the queue
- **What is the difference between array d queue and priority queue ?**
  D Queue is -> Double ended queue: we can perform operation from front and rear
  Pollfirst -> deletes from first (0th index)-> this is same poll method of Queue
  pollLast-> deletes from last(ar.length-1)-> this is same as pop method of Queue
  Peek first -> peek method behavior like queue-> returns 0th element
  peekLast-> peek method behavior like Stack -> returns last index
- **What is Stack and how can I implement Stack ?**
  Stack using array: initialize object[] : Refer to the custom Stack program
- **Difference between linear Search or Binary search ?**
  Binary search: 1. array to be in sorted order
  find the middle and compare the element to be searched with middle if it is greater go towards right side :
  Lesser go towards left side. Continue until we get the element
- **What is the problem with m=l+r/2 in binary search ?**
  Overflow issue: look at the below example : max is integer max value

```
int midIndex=(10000+2147483647)/2;
        System.out.println(midIndex);
```

- **What is pre requisite for binary search ?**
  Array to be in Sorted order
- **What is Queue and how can I implement Queue ?**
  FIFO: look at the code
- **Write a program to implement Stack and Queue using linked list ?**
  Create Node class internally: look at the code
- **How to merge two sorted arrays ? (result should be sorted array) ?**
  Two way merging: refer to the program
- **Write the algorithm for merge sort ?**
  To perform the merge sort the prerequisite is : Two way merging
  But to apply two ways merging we need 2 sorted arrays. To get the sorted arrays we follow the divide and conquer algorithem and divide the arrays into individual array elements and then perform two way merging
- **Write the algorithm for Quick sort ?**
  Quick sort work on the principle of playing cards
  Take the last element as PIVOT element and compare the elements from the stating to the pivot element if we find any lesser element than pivot then replace it with ith position.
  After completing the comparisons, elements before PIVOT are less (may or may not be sorted) . Elements after the PIVOT are higher value(may or may not be sorted) but we can say that PIVOT is sorted so follow the divide conquer based on 0th index to pivot-1 and pivot +1 to end of the array
  Refer to the program
- **Write a program to sort hashMap by values ?**
  Refer to Notes
- **What is a binary tree ?**
  A tree which has 0 or 1 or 2 child nodes
- **What is the difference between full binary tree and complete binary tree ?**
  FBT: all the nodes should be present till the height
  CBT: if the height of tree is H-> till h-1 level it should FBT and at h level nodes should left justified.

What is BST ?
- A tree can called as BST if the element lesser than root should be towards left side and greater element should be towards right side. Entire tree should follow the same structure.

- **Write a program to get the minimum and maximum values from binary search tree ?**
  - Start with Root node and go to root.left and call recursively until left become null. This will give min value in BST
  - Start with Root node and go to root.right and call recursively until right become null. This will give max value in BST
- **What are the tree traversal techniques ?**

- Inorder: Left ->root-> right: Inorder on BST will provide Ascending order
- Pre order : root -> left and right
- Post order: left ->right and root

- **What is the difference between hash set and tree set ?**
  - Hash set is doesn't follow insertion order and internal implementation of hashset is HashMap
  - Tree set is ordered by natural sorting order. Internal implementation is tree Map
- **What are the difference between tree map and hash map ?**
  - Tree map follows natural sorting order. Tree Map uses comparator internally
- **What is copy on write Array set ?**
  - To overcome concurrement modification exception in list -> we use copy on write array list
  - To overcome concurrement modification exception in set -> we use copy on write array set
  - To overcome concurrement modification exception in Map -> we use concurrent hash map
- **How to check whether the given tree is BST ?**
  - Write the in order -> check if it is ascending order -> BST
  - Start from extreme left and check the element should be lesserthan root and right should be greaterthan root
- **What is level order ?**
  - Writing all the nodes at each height
  - Calculate the height of tree and start with 1st node and print the root element. And call the same method by reducing the level
- **How to calculate the height of given tree ?**
  - Recursively traverse towards left side and rightside and then which ever is the higher value that value +1 will be height of tree
- **How to construct BST ?**
  - First create the node object make the first element as root node
  - For second element onwards iterate and check the value that is getting added is lesserthan root and add towards left side if not right side
- **What is Heap ?**
  - Heap is nothing but CBT
- **What is Max Heap ?**
  - It should be CBT and root element should be greater than child element
- **What is min heap ?**
  - It should be CBT and root element should be lesser than child element
- **How to implement BFS and DFS ?**
  - BFS: Visit the node and explore the node -> Queue is used
  - DFS: visit the node and explore until there no other node-> Stack
- **How graph can be represented ?**
  - Adjacency List
- **What is the difference between hoare's partition and lomuto's partitions ?**
  - Lomutos: PIVOT element is selected as last element
  - Hoare's : PIVOT element is selected as first element
- **Write a program to implement Stack using Queue ?**
  - Look at the program under questions package