Q:-diff b/w hashmap and hashtable other than synchronized.?

==========

Ans:- hashtable Hashmap

========== ==========

synchronized method. none synchornize method

thread safe not thread safe

performance low performance high

null not insert in both null Insertion is allowed for Key (Only Once) and allowed for Values (Any Number ofTimes)

size of bucket is 11 size of bucket is 16

version 1.0v version 1.2

legacy class. non legacy class

==========

Q:-how to work hashtable internally.?

Ans:- In hashtable it used array of bucket to store elements and array of bucket size is 11.

steps of work:-

---

1. when we store element in hashtable using put() method First put api caluclate hashcode based on key.

int hashcode=hash(key value);

2. we can't store hsahcode directly in hastable. bcz array of bucket size is 11.

3. then it will convert hashcode into index of bucket.It will divide hashcode by 11 and give reminder as a index value.

4. Now, on based this value it will store element in hashtbale.

5. when we printall value it will follow top to bottom and right to left(for multiple value in single bucket) rule.

5. If same hashcode came for two keys value. then Hash conclusion will create. now it save this same value in same bucket.

6. null Insertion is allowed for Key (Only Once) and allowed for Values (Any Number ofTimes) .

Q:- diff b/w stack and queue.?

Ans:- stack

---

1. It is the Child Class of Vector.
2. It is a Specially Designed Class for Last In First Out (LIFO) Order.
3. version 1.0

Queue=

---

1. Queue is a Child Interface of Collection.
2. If we want to Represent a Group of Individual Objects Prior to processing then we should go for Queue.

3. From 1.5 Version onwards LinkedList also implements Queue Interface.
4. Usually Queue follows FIFO Order. But Based on Our Requirement we can Implement Our Own Priorities Also (PriorityQueue)
5. LinkedList based Implementation of Queue always follows FIFO Order.
6. 1.5 version

Q:- Diff b/w linkedlist and arrayList.? Is RandomAccess acess a marker interface.?
Ans:- LinkedList:-

==========

1. The Underlying Data Structure is Double LinkedList.
2. Implements Serializable and Cloneable Interfaces but Not RandomAccess Interface.
3. Best Choice if Our Frequent Operation is Insertion OR Deletion in the Middle.
4. Worst Choice if Our Frequent Operation is Retrieval.
5. we can decalre two construtor linkedlist(),linkedlist(collection c).

ArrayList:-

==========

1. The Underlying Data Structure for ArrayList is Resizable Array OR Growable Array.
2. Implements Serializable and Cloneable Interfaces, RandomAccess Interface.
3. Best Suitable if Our Frequent Operation is Retrieval Operation.
4. Worst Choice if Our Frequent Operation is Insertion OR Deletion in the Middle. Because it required Several Shift Operations Internally.
5. we can decalre three construtor arraylist(),arraylist(intial capacity),arraylist(collection c).

Q:- how to sort employee class.?
Ans :-

======DNS of employee class example 1 uding comparable interface==========

```
        import java.util.TreeSet
    public class EmployeeDNS {
        public static void main(String[] args) {
                employeeSort e1 = new employeeSort("nag", 100);
                employeeSort e2 = new employeeSort("balaiah", 200);
                employeeSort e3 = new employeeSort("chiru", 50);
                employeeSort e4 = new employeeSort("venki", 150);
                employeeSort e5 = new employeeSort("nag", 100);
                TreeSet t = new TreeSet();
                t.add(e1);
                t.add(e2);
                t.add(e3);
                t.add(e4);
```

```
                t.add(e5);
                System.out.println(t);
        }
}
class employeeSort implements Comparable {
        String name;
        int eid;
        employeeSort(String name, int eid) {
                this.name = name;
                this.eid = eid;
        }
        public String toString() {
                return name + "---" + eid;
        }
        public int compareTo(Object obj) {
                int eid1 = this.eid;
                employeeSort e = (employeeSort) obj;
                int eid2 = e.eid;
                if (eid1 < eid2) {
                        return -1;
                } else if (eid1 > eid2) {
                        return 1;
                } else {
                        return 0;
                }
        }
}

============2nd Example custome sort employee class  using comparator=========
import java.util.*;
public class ComparableDemo4 {
        public static void main(String[] args) {
                employee1 e1=new employee1("nag",100);
                employee1 e2=new employee1("balaiah",200);
                employee1 e3=new employee1("chiru",50);
                employee1 e4=new employee1("venki",150);
                employee1 e5=new employee1("nag",100);
                TreeSet t1 =new TreeSet(new MyComparator5());
                t1.add(e1);
                t1.add(e2);
                t1.add(e3);
                t1.add(e4);
                t1.add(e5);
```

```
                System.out.println(t1);
        }
}
class employee1
{
        String name ;
        int eid;
        employee1(String name, int eid)
        {
                this.name=name;
                this.eid=eid;
        }
        public  String toString ()
        {
        return name+"---"+eid;
        }
}
class MyComparator5 implements  Comparator
{
        public int compare(Object obj1,Object obj2)
        {
                employee1 e1=(employee1)obj1;
                employee1 e2=(employee1)obj2;
                String s1=e1.name;
                String s2=e2.name;
                return s1.compareTo(s2);  // this will perform ASC for DEC to change this with
s2.compareTo(s1)

                ----------2nd way to override to compare method from comparator for integer
value ----------
                public int compare(Object obj1, Object obj2) {
       Integer i1 = (Integer)obj1;
       Integer i2 = (Integer)obj2;
       if(i1 < i2)
        return +1;      //here + using for dec order
       else if(i1 > i2)
       return -1;
       else
       return 0;
}

        }
}
```

NOTE:-All Wrapper Classes, String Class Already Implements Comparable Interface. But StringBuffer Class doesn't Implement
Comparable Interface. SO all wrapper class will perform DNS sorting. But If we want to Cutomized sorting (DEC order) of these
classes then we should go for comparator interface.
But for Our Own classes like employee ,student if we want to perform DNS (ASC sorting) and DEC sorting then we can use comaprable
and comparator interface any one can chosse.

==========sorting employee class with ID and Name Example 3==========

```
import java.util.*;

public class ComparableDemo4 {
	public static void main(String[] args) {
		employee1 e1 = new employee1("nag", 100);
		employee1 e2 = new employee1("balaiah", 200);
		employee1 e3 = new employee1("chiru", 50);
		employee1 e4 = new employee1("venki", 150);
		employee1 e5 = new employee1("nag", 100);
		ArrayList t1 = new ArrayList<>();
		t1.add(e1);
		t1.add(e2);
		t1.add(e3);
		t1.add(e4);
		t1.add(e5);
		System.out.println("Without Sorting Operation" + t1);
		Collections.sort(t1, new MyComparator5());
		System.out.println("After sorting list of Employees based on name" + t1);
		Collections.sort(t1, new MyComparatorID());
		System.out.println("After sorting list of Employees based on id" + t1);
	}
}

class employee1 {
	String name;
	int eid;
	employee1(String name, int eid) {
		this.name = name;
		this.eid = eid;
	}
	public String toString() {
```

```
                return name + "---" + eid;
        }
}
class MyComparator5 implements Comparator {
        public int compare(Object obj1, Object obj2) {
                employee1 e1 = (employee1) obj1;
                employee1 e2 = (employee1) obj2;
                String s1 = e1.name;
                String s2 = e2.name;
                return s1.compareTo(s2);
        }
}
class MyComparatorID implements Comparator {
        public int compare(Object obj1, Object obj2) {
                employee1 e1 = (employee1) obj1;
                employee1 e2 = (employee1) obj2;

                int eid1 = e1.eid;
                int eid2 = e2.eid;
                if (eid1 < eid2) {
                        return -1;
                } else if (eid1 > eid2) {       // this logic perfoming ASC order of we want to perform
DEC then change + with - and - with +.
                        return 1;
                } else {
                        return 0;
                }
        }
}
```

----------4th example ----------

Write a Program to Insert Employee Objects into the TreeSet where DNSO is Based on Ascending Order of EmployeeId and Customized Sorting Order is Based on Alphabetical Order of Names:

```
import java.util.*;
class Employee implements Comparable {
String name;
int eid;
Employee(String name, int eid) {
this.name = name;
this.eid = eid;
}
public String toString() { return name+"-----"+eid; }
```

```
public int compareTo(Object obj) {
int eid1 = this.eid;
Employee e = (Employee)obj;
int eid2 = e.eid;
if(eid1 < eid2) return -1;
else if(eid1 > eid2) return 1;
else return 0;
}
}
class CompComp {
public static void main(String[] args) {
Employee e1 = new Employee("Nag", 100);
Employee e2 = new Employee("Bala", 200);
Employee e3 = new Employee("Chiru", 50);
Employee e4 = new Employee("Venki", 150);
Employee e5 = new Employee("Nag", 100);
TreeSet t = new TreeSet();
t.add(e1);
t.add(e2);
t.add(e3);
t.add(e4);
t.add(e5);
System.out.println(t);
TreeSet t1 = new TreeSet(new MyComparator());
t1.add(e1);
t1.add(e2);
t1.add(e3);
t1.add(e4);
t1.add(e5);
System.out.println(t1);
}
}
class MyComparator implements Comparator {
public int compare(Object obj1, Object obj2) {
Employee e1 = (Employee) obj1;
Employee e2 = (Employee) obj2;
String s1 = e1.name;
String s2 = e2.name;
return s1.compareTo(s2);
}
}
```

Q:- reverse string using recursion in java.?

```
public class StringReverse {
        void reverse(String str)
          {
             if ((str==null)||(str.length() <= 1))
               System.out.println(str);
             else
             {
                System.out.print(str.charAt(str.length()-1));
                reverse(str.substring(0,str.length()-1));
             }
          }

          /* Driver program to test above function */
          public static void main(String[] args)
          {
                System.out.println("Enter string value");
              Scanner sc=new Scanner(System.in);
             StringReverse obj = new StringReverse();
             obj.reverse(sc.next());
          }
}
```

==========

Q:- Comparison of Comparable and Comparator:

ans:- Comparable:---

==========

Present in java.lang Package
It is Meant for Default Natural Sorting
Order.
Defines Only One Method compareTo().
All Wrapper Classes and String Class
implements Comparable Interface.

Comparator:--

==========

Present in java.util Package
It is Meant for Customized Sorting Order.
Defines 2 Methods compare() and equals().
The Only implemented Classes of
Comparator are Collator and
RuleBaseCollator.

Q:-how to synchronized to hashmap in singleton class.?
Ans:-

```
 package singleTon;
import java.util.Collections;
import java.util.HashMap;
import java.util.Map;

public class EagerSingleTon {

	private EagerSingleTon() {}    //1 at declare private constructor
	public static final  Map Instance =Collections.synchronizedMap(new HashMap());

	public static  Map getInstance() {
		// TODO Auto-generated method stub
		return Instance;
	}
}
```

```
 package singleTon;
import java.util.Map;

public class client {

	public static void main(String[] args) {
		Map  register1= EagerSingleTon.getInstance();
		Map register2= EagerSingleTon.getInstance();
		System.out.println(register1==register2);
	}

}
```

Q:-how to synchronized hashmap.?
Ans:-

Exmaple synchornized hashMap

==========

```
  class  synchronizedHashMap
  {
  public static void main (String args[])
  {
  Map<Integer,String> map=new HashMap<Integer,String>();
  map.put(101,"siva");
  map.put(102,"reddy");
```

```
  Collections.synchornizedMap(map);
  Set<Map,Entry<Integer,String>> entries= map.entrySet();
  for(Map.Entry<Integer,String> value: entries)
  {
    System.out.println("Key :"+value.getkey());
          System.out.println("value :"+value.getValue());
  }
  }
  ============================================================
```

Example synchornize Hashmap with Threading

```
==========================================
import java.util.concurrent.ConcurrentHashMap;
class ThreadClass extends Thread
{
          static Map< Integer,String> hashMap = new HashMap<>();
   static Map<Integer,String> hashmp = Collections.synchronizedMap(hashMap);
public void run()
{
 try
 {
           Thread.sleep(200);
 }
 catch(InterruptedException e)
 { }
 System.out.println("Child thread updating list ");
 hashmp.put(103,"C");
}
public static void main(String[] args) throws InterruptedException {
           hashmp.put(101,"A");
           hashmp.put(102,"B");
ThreadClass mc=new ThreadClass();
mc.start();
Set s1=hashmp.keySet();
Iterator itr = s1.iterator();
while (itr.hasNext()) {
Integer I1 = (Integer) itr.next();
System.out.println("Main Thread Iterating list adn current Entery is "+I1+"..."+hashmp.get(I1));
MainClass.sleep(300);
}
System.out.println(hashmp);
}
}
```

output:-

Main Thread Iterating list adn current Entery is 101...A
Child thread updating list
Exception in thread "main" java.util.ConcurrentModificationException
---

Another Big Problem with Traditional Collections is while One Thread iterating Collection, the Other Threads are Not allowed to Modify Collection Object simultaneously if we are trying to Modify then we will get ConcurrentModificationException.
Σ Hence these Traditional Collection Objects are Not Suitable for Scalable Multi Threaded Applications.
Σ To Overcome these Problems SUN People introduced Concurrent Collections in 1.5 Version.

==========ConcurrentHashMap example==========

```
import java.util.concurrent.ConcurrentHashMap;
class ThreadClass extends Thread
{
        static ConcurrentHashMap hashmp=new ConcurrentHashMap();
public void run()
{
  try
  {
          Thread.sleep(200);
  }
  catch(InterruptedException e)
  { }
  System.out.println("Child thread updating list ");
  hashmp.put(103,"C");
}
public static void main(String[] args) throws InterruptedException {
        hashmp.put(101,"A");
        hashmp.put(102,"B");
ThreadClass mc=new ThreadClass();
mc.start();
Set s1=hashmp.keySet();
Iterator itr = s1.iterator();
while (itr.hasNext()) {
Integer I1 = (Integer) itr.next();
System.out.println("Main Thread Iterating list adn current Entery is "+I1+"..."+hashmp.get(I1));
```

```
ThreadClass.sleep(300);
}
System.out.println(hashmp);
}
}
```

output:-

```
Main Thread Iterating list adn current Entery is 101...A
Child thread updating list
Main Thread Iterating list adn current Entery is 102...B
Main Thread Iterating list adn current Entery is 103...C
{101=A, 102=B, 103=C}
```

==========

Q:- diff b/w synchronized hashmap and cuncurrent hashmap.?

Ans:-

HashMap VS ConcurrentHashMap

==========

1. It is Not Thread Safe.
2. It is Thread Safe.
3. Relatively Performance is High because Threads are Not required to wait to Operate on HashMap.
3. Relatively Performance is Low because Some Times Threads are required to wait to Operate on ConcurrentHashMap.
4. While One Thread iterating HashMap the Other Threads are Not allowed to Modify Map Objects Otherwise we will get Runtime Exception Saying ConcurrentModificationException.
4. While One Thread iterating ConcurrentHashMap the Other Threads are allowed to Modify Map Objects in Safe Manner and it won't throw ConcurrentModificationException.
5. Iterator of HashMap is Fail-Fast and it throws ConcurrentModificationException.
6. Iterator of ConcurrentHashMap is Fail-Safe and it won't throws ConcurrentModificationException.
7. null is allowed for Both Keys and Values.
7. null is Not allowed for Both Keys and Values. Otherwise we will get NullPointerException.
8. Introduced in 1.2 Version.
8. Introduced in 1.5 Version.

==========
Q:-how to sort list of objects.?
Ans:- To sort the array of objects we will use Arrays.sort() method.
If we need to sort collection of object we will use Collections.sort().

```
private void sortNumbersInArrayList() {
        List<Integer> integers = new ArrayList<>();
        integers.add(5);
        integers.add(10);
        integers.add(0);
        integers.add(-1);
        System.out.println("Original list: " +integers);
        Collections.sort(integers);
        System.out.println("Sorted list: "+integers);
        Collections.sort(integers, Collections.reverseOrder());
        System.out.println("Reversed List: " +integers);
}
```

==========
Q:- what is hash collision.?
ans:-A collision occurs when a hash function returns same bucket location for two different keys.
==========
Q:- how to put custom class object in map .?
Ans:- using put() method.
==========
Q:- ArrayList vs Vector.?
Ans:- ArrayList and Vector

1.Every Method Present Inside ArrayList is Non – Synchronized.
1. Every Method Present in Vector is Synchronized.
2. At a Time Multiple Threads are allow to Operate on ArrayList Simultaneously and Hence ArrayList Object is Not Thread Safe.
2. At a Time Only One Thread is allow to Operate on Vector Object and Hence Vector Object is Always Thread Safe.
3. Relatively Performance is High because Threads are Not required to Wait.
3. Relatively Performance is Low because

Threads are required to Wait.

4. Introduced in 1.2 Version and it is Non – Legacy.
5. Introduced in 1.0 Version and it is Legacy.
6. In array list we can use three constructor.
6. In vector we can use four constructor.
7. In arraylist size is 10 but formula of array incease is cc*3/2+1.
7. In vector size is also same but foumla is cc*2.

==========

Q:- sorting of an array using library function ASC order.

Ans:

```
import java.util.Arrays;

public class SortExample
{
   public static void main(String[] args)
   {
      // Our arr contains 8 elements
      int[] arr = {13, 7, 6, 45, 21, 9, 2, 100};

      // Sort subarray from index 1 to 4, i.e.,
      // only sort subarray {7, 6, 45, 21} and
      // keep other elements as it is.
      Arrays.sort(arr, 1, 5);

      System.out.printf("Modified arr[] : %s",
                  Arrays.toString(arr));
   }
}
```

----------DEC order example 2----------

```
import java.util.Arrays;
import java.util.Collections;

public class SortExample
{
   public static void main(String[] args)
   {
      // Note that we have Integer here instead of
      // int[] as Collections.reverseOrder doesn't
      // work for primitive types.
      Integer[] arr = {13, 7, 6, 45, 21, 9, 2, 100};
```

```
        // Sorts arr[] in descending order
        Arrays.sort(arr, Collections.reverseOrder());

        System.out.printf("Modified arr[] : %s",
                          Arrays.toString(arr));
    }
}
```

==========

Q:- HashSet vs Hashmap.?

Ans:- HashMap:-

---

1. Duplicate Keys are Not Allowed. But Values can be Duplicated.
2. Heterogeneous Objects are allowed for Both Keys and Values.
3. null Insertion is allowed for Key (Only Once) and allowed for Values (Any Number of Times)

HAshSet:=

---

1. Duplicate Objects are Not Allowed. If we are trying to Insert Duplicate Objects then we won't get any Compile Time OR Runtime Error. add() Simply Returns false.
2. null Insertion is Possible.
3. Heterogeneous objects are allowed.

Q:- arraylist iterate using for loop.?

Ans :-

```
public static void main(String[] args) {
ArrayList namesList = new ArrayList();
   namesList.add("aaa");
   namesList.add("bbb");
   namesList.add("ccc");
   namesList.add("ddd");

for(int i = 0; i < namesList.size(); i++)
{
   System.out.println(namesList.get(i));
}
}
}
```

==========2nd way ==========

Iterate arraylist with foreach loop

```
ArrayList namesList = new ArrayList();
                namesList.add("aaa");
                namesList.add("bbb");
                namesList.add("ccc");
                namesList.add("ddd");

for(String name : namesList)
{
   System.out.println(name);
}
```

Q:- how to remove duplicay from arrayList.?
Ans:--
```
public class RemoveDuplicateArrayList {
   public static void main(String[] args) {
      List<String> l = new ArrayList<String>();
      l.add("Mango");
      l.add("Banana");
      l.add("Mango");
      l.add("Apple");
      System.out.println(l.toString());
      Set<String> s = new LinkedHashSet<String>(l);
      System.out.println(s);
   }
}
```

If you don't want duplicates in a Collection, you should consider why you're using a Collection that allows duplicates.
The easiest way to remove repeated elements is to add the contents to a Set (which will not allow duplicates) and
then add the Set back to the ArrayList:

```
Set<String> set = new HashSet<>(yourList);
yourList.clear();
yourList.addAll(set);
```
==========
Q:- 1.Write a Java program to remove duplicate elements from an arraylist without using collections (without using set)
Ans:-
```
import java.util.ArrayList;

public class RemoveDuplicates {
public static void main(String[] args){
```

```
    ArrayList<Object> al = new ArrayList<Object>();

    al.add("java");
    al.add('a');
    al.add('b');
    al.add('a');
    al.add("java");
    al.add(10.3);
    al.add('c');
    al.add(14);
    al.add("java");
    al.add(12);

System.out.println("Before Remove Duplicate elements:"+al);

for(int i=0;i<al.size();i++){

 for(int j=i+1;j<al.size();j++){
            if(al.get(i).equals(al.get(j))){
                al.remove(j);
                j--;
            }
    }

 }

    System.out.println("After Removing duplicate elements:"+al);

}

}
```

==========

Q:- If we have equals() then why we use comparable and comparator.?
ans:- equals() method only used for content comparision . if will not perform sorting.
But cmparable and comparator used for sorting to element with there method. based on comaprison it wil sort to element.

==========

Q:- Diff b/w sorted and order collection.?

Ans:- An ordered collection means that the elements of the collection have a specific order. The order is independent of the value. A List is an example.
A sorted collection means that not only does the collection have order, but the order depends on the value of the element.
A SortedSet is an example.

Q:- how get method of hashmap works.?
Ans:- Using get method()

Now lets try some get method to get a value. get(K key) method is used to get a value by its key.
If you don't know the key then it is not possible to fetch a value.

Fetch the data for key sachin:
map.get(new Key("sachin"));
Steps:
=====
1. Calculate hash code of Key {"sachin"}. It will be generated as 115.
2. Calculate index by using index method it will be 3.
3. Go to index 3 of array and compare first element's key with given key. If both are equals then return the value, otherwise check for next element if it exists.
4. In our case it is found as first element and returned value is 30.

Q;- how to find middle elementof linkedlist with single go.?
ans:-

```
//Node class which has two attributes
public class Node {
    int data;
    Node next;
    public Node(int data) {
        this.data = data;
        this.next = null;
    }
}
        * Main class
 */
public class MiddleElement {
    private Node head;
    //In constructor, Initialize head attribute to null
    public MiddleElement() {
        this.head = null;
    }
```

```
public Node insert(int data) {
   if (head == null) {
      head = new Node(data);
    } else {
       // Create a new node
       Node temp = new Node(data);
       // New node points to head
       temp.next = head;
       // Head points to a new node
       head = temp;
    }
    return head;
}
//Logic to print middle element of a linked list
public void printMiddleElement() {

     Node slow = head;
     Node fast = head;
     while (fast != null && fast.next != null) {
           slow = slow.next;
           fast = fast.next.next;
     }

     System.out.println(" Middle Element of a Linked List is " + slow.data);
 }
 public void print() {

    Node temp = head;

    /*
     * Traverse a list and check if it not points to null. If it points to
     * null, It means there is no node present after that and we need to end
     * the loop.
     */
     while (temp != null) {
        System.out.println(temp.data);
        temp = temp.next;
     }
  }

 public static void main(String[] args) {

     MiddleElement ll = new MiddleElement();
```

```
        ll.insert(6);
        ll.insert(5);
        ll.insert(8);
        ll.insert(9);
        ll.insert(15);
        ll.print();
        ll.printMiddleElement();
    }
}
```

==========

Q;- hashset internal implementation how it store elements.?
ans:- same as HAshmap and hashtable.

Q;-what if when a hashmap doesnot have space to add new object in it.?
Ans:-- Hashmap has a capacity and load factor parameter.So ,
if the number of items in this hashmap is more than capacity* load factor, a new hashmap will be reconstructed

==========

Q:- write a code to implement your own map .? don't use map implementation of collection framework.
Ans:=

==========

Q:= diff b/w arrayList and treeSet
Ans:- from copy

==========

Q:- how treeset maintain ordering.?
Ans:- The ordering of the elements is maintained by a set using their natural ordering whether or not an explicit comparator is provided.
 This must be consistent with equals if it is to correctly implement the Set interface.
It can also be ordered by a Comparator provided at set creation time, depending on which constructor is used.

==========

Q;- what is comparator.?
Ans:-Present in java.util Package
It is Meant for Customized Sorting Order.
Defines 2 Methods compare() and equals().
The Only implemented Classes of
Comparator are Collator and

RuleBaseCollator.

==========

Q:- how treeset know which comparator to use.?

Ans :- TreeSet t = new TreeSet(Comparator c);

Creates an Empty TreeSet Object where all Elements will be Inserted According to Customized Sorting Order which is described by Comparator Object.

==========

Q:- what all collection classes introduced in java 1.5.?

ans:- from copy

==========

Q:-how to traverse backword in set and map.?

Ans:- In Map :=

```
static void methodOne() {
        Map<String, String> map = new LinkedHashMap<String, String>();
        map.put("Key1", "Value1");
        map.put("Key2", "Value2");
        map.put("Key3", "Value3");
        map.put("Key4", "Value4");
        map.put("Key5", "Value5");
        map.put("Key6", "Value6");
        //create an arraylist initialized with keys of map
        ArrayList keyList = new ArrayList(map.keySet());
        for (int i = keyList.size() - 1; i >= 0; i--) {
                //get key
                String key = keyList.get(i);
                System.out.println("Key :: " + key);
                //get value corresponding to key
                String value = map.get(key);
                System.out.println("Value :: " + value);
                System.out.println("--------------------------------");

                ==================In SET==========================

Set<MyType> mySet = new LinkedHashSet();
...
MyType[] asArray = mySet.toArray();

for (int i = asArray.length - 1; i>=0; i--){
```

..

}

Q;-how to use custom class as a key in hashmap.?
Ans:-
==========
Q:- what is cursor.? temporay manupulation on set of data.?
Ans:- If we want to get Objects One by One from the Collection then we should go for Cursors.
Σ There are 3 Types of Cursors Available in Java.
1) Enumeration
2) Iterator
3) ListIterator

==========
Q;- backword traverse in linkedlist without listIterator.? or How to reverse a Singly Linked List in Java
Ans:- // Java program for reversing the linked list

```
class LinkedList {

	static Node head;

	static class Node {

		int data;
		Node next;

		Node(int d) {
			data = d;
			next = null;
		}
	}

	/* Function to reverse the linked list */
	Node reverse(Node node) {
		Node prev = null;
		Node current = node;
		Node next = null;
		while (current != null) {
			next = current.next;
			current.next = prev;
```

```
                prev = current;
                current = next;
            }
            node = prev;
            return node;
        }

        // prints content of double linked list
        void printList(Node node) {
            while (node != null) {
                System.out.print(node.data + " ");
                node = node.next;
            }
        }

        public static void main(String[] args) {
            LinkedList list = new LinkedList();
            list.head = new Node(85);
            list.head.next = new Node(15);
            list.head.next.next = new Node(4);
            list.head.next.next.next = new Node(20);

            System.out.println("Given Linked list");
            list.printList(head);
            head = list.reverse(head);
            System.out.println("");
            System.out.println("Reversed linked list ");
            list.printList(head);
        }
}


// This code has been contributed by Mayank Jaiswal
```

Q:- how to iterate in hashmap.?

Ans:-

First way using For each Loop

```
 import java.util.HashMap;
import java.util.Map;
public class IterateHashMap {
        public static void main(String[] args) {
```

```
                Map<String, String> map = new HashMap<String, String>();
                map.put("key1", "value1");
                map.put("key2", "value2");
                for (Map.Entry<String, String> entry : map.entrySet()) {
                   System.out.println(entry.getKey() + " = " + entry.getValue());
                }
        }
}
```

----------2nd way using iterator----------

```
Iterator<Entry<String, String>> it = map.entrySet().iterator();
                while (it.hasNext()) {
                        Map.Entry<String, String> pair = (Map.Entry<String, String>) it.next();
                        System.out.println(pair.getKey() + " = " + pair.getValue());
                }
```

==========

Q;- how to sort arrayList consiting of string objects.? and how to sort array list consiting of user defined object.?

Ans use collection.sort(list),collection.Sort(list,comparator), for user defined use compareTo() method.

==========

Q;- what is hashing how hashing works.?

Ans:- above answer

==========

video--question===

==========

Q:-What is the difference between Collection API and Streams API? What is the value added by streams actually?

Ans:- Main differences between Collection and Stream API in Java 8 are:

I. Version: Collection API is in use since Java 1.2. Stream API is recent addition to Java in version 8.

II. Usage: Collection API is used for storing data in different kinds of data structures. Stream API is used for computation
of data on a large set of Objects.

III. Finite: With Collection API we can store a finite number of elements in a data structure. With Stream API, we can handle

streams of data that can contain infinite number of elements.

IV. Eager vs. Lazy: Collection API constructs objects in an eager manner. Stream API creates objects in a lazy manner.

V. Multiple consumption: Most of the Collection APIs support iteration and consumption of elements multiple times. With Stream
API we can consume or iterate elements only once.

==========

Q:- What is Concurrent Hashmap? Compare it with Hashmap.

Ans:-

HashMap V/S ConcurrentHashMap

==========

1. It is Not Thread Safe.
1. It is Thread Safe.
2. Relatively Performance is High because Threads are Not required to wait to Operate on HashMap.
2. Relatively Performance is Low because Some Times Threads are required to wait to Operate on ConcurrentHashMap.
3. While One Thread iterating HashMap the Other Threads are Not allowed to Modify Map Objects Otherwise we will get Runtime Exception Saying
3. ConcurrentModificationException.
   While One Thread iterating ConcurrentHashMap the Other Threads are allowed to Modify Map Objects in Safe Manner and it won't throw ConcurrentModificationException.
4. Iterator of HashMap is Fail-Fast and it throws ConcurrentModificationException.
4. Iterator of ConcurrentHashMap is Fail-Safe and it won't throws ConcurrentModificationException.
5. null is allowed for Both Keys and Values. null is Not allowed for Both Keys and Values.
5. Otherwise we will get NullPointerException.
6. Introduced in 1.2 Version.
6. Introduced in 1.5 Version.

==========

I have a set of elements which are not duplicates. Which collection you would use to store these objects?

==========
==========
Q:- Can you explain the hashing technique to make Java store all specific objects in one bucket?
Ans:- above explain

==========
==========
Q;- How can you able to sort out the data available in a list of employees on basis of employeeName and employeeAge?
Ans:- above answer.

==========
==========
How can you sort out the above data on basis of empDept and if same dept, then sort out on basis of empName.
How can you do so?
==========
what is the difference between HashSet and LinkedHashSet?
Ans;- HashSet V/SLinkedHashSet
---
1. The Underlying Data Structure is
Hashtable.
1. The Underlying Data Structure is a Combination of LinkedList and Hashtable.
2. Insertion Order is Not Preserved.
2. Insertion Order will be Preserved.
3. Introduced in 1.2 Version.
3. Introduced in 1.4 Version.
==========
Q:- what is the default sorting order in TreeMap?
Ans :- DNS
==========
Q:- what are collection you have used for sort out in application.?
Ans;- ArrayList,TreeSet etc
==========
Q:- have you idea about current hashmap.?
Ans:- yes explain above
==========
Q:- diff b/w linked list and array list.?
Ans:- above explin
==========
7. what is list .?
Ans:- above

==========
8. which list is most useful for searching.?
Ans:- arrayList
==========
9. which list is most useful for insertion and deletion.?
Ans:- linkedlist
==========
10. which list is most useful for travesring.?
Ans:- arrayLsit
==========
11. what is diff b/w vector and array list.?
Ans abovve
==========
12. how memory manage in arraylist and vector.?
Asn:- above
==========
13. how to synchornize two hashmap in singleton class.?(once again singleton topic).
Ans:- above
==========
14.how to sort list of objects.? (Once again list sort question)
Ans;- above
==========
Twice null added in set what will happen.?
==========
diff b/w arraylist and linked list .? Is random access a marker interface?
Ans:- above
==========
Q. What is Difference between Iterator and ListIterator?
Ans:- from pdf notes

==========
Q. What is Comparator?
Ans:- If we are Not satisfied with Default Natural Sorting Order OR if Default Natural Sorting Order is Not Already Available then we can Define Our Own Sorting by using Comparator Object.
This Interface Present in java.util Package.
Methods: It contains 2 Methods compare() and equals().
==========
1.Why Map interface doesn't extend Collection interface?
Set is unordered collection and does not allows duplicate elements.
List is ordered collection allows duplicate elements.
Where as Map is key-value pair.
It is viewed as set of keys and collection of values.

Map is a collection of key value pairs so by design they separated from collection interface.

==========
=
2.What is difference between HashMap and Hashtable?
Synchronization or Thread Safe
Null keys and null values
Iterating the values
Default Capacity
Hashmap%2Bvs%2Bhashtable
Click here for the
Differences between HashMap and Hash-table

==========
3.Differences between comparable and comparator?
Comparable Interface is actually from java.lang package.
It will have a method compareTo(Object obj)to sort objects
Comparator Interface is actually from java.util package.
It will have a method compare(Object obj1, Object obj2)to sort objects
Read more : comparable vs comparator
==========
=======
4.How can we sort a list of Objects?
To sort the array of objects we will use Arrays.sort() method.
If we need to sort collection of object we will use Collections.sort().
==========
==========
5.What is difference between fail-fast and fail-safe?
Fail fast is nothing but immediately report any failure. whenever a problem occurs fail fast system fails.
in java Fail fast iterator while iterating through collection of objects sometimes concurrent modification exception will come there are two reasons for this.
If one thread is iterating a collection and another thread trying to modify the collection.
And after remove() method call if we try to modify collection object

==========
6. What is difference between Iterator ,ListIterator and Enumeration?
Enumeration interface implemented in java 1.2 version.So Enumeration is legacy interface.
Enumeration uses elements() method.
Iterator is implemented on all Java collection classes.
Iterator uses iterator() method.
Iterator can traverse in forward direction only.

ListIterator is implemented only for List type classes
ListIterator uses listIterator() method.
Read more :
What is difference between Iterator ,ListIterator and Enumeration?
==========
==
7.What is difference between Set and List in Java?
A set is a collection that allows unique elements.
Set does not allow duplicate elements
Set allows only one null value.
Set having classes like :
HashSet
LinkedHashSet
TreeSet
List having index. and ordered collection
List allows n number of null values.
List will display Insertion order with index.
List having classes like :
Vector
ArrayList
LinkedList
==========
8.Differences between arraylist and vector?
Vector was introduced in first version of java . that's the reason only vector is legacy class.
ArrayList was introduced in java version1.2, as part of java collections framework.
Vector is synchronized.
ArrayList is not synchronized.
Read more: Differences between arraylist and vector
==========

9.What are the classes implementing List interface?
ArrayList
LinkedList
Vector
==========
10. Which all classes implement Set interface ?
HashSet
LinkedHashSet
TreeSet
==========
11.How to make a collection thread safe?
Vector, Hashtable, Properties and Stack are synchronized classes, so they are thread-safe and can be used in multi-threaded environment.

By using Collections.synchronizedList(list)) we can make list classes thread safe.
By using
java.util.Collections.synchronizedSet() we can make set classes thread safe.
==========
12.Can a null element added to a TreeSet or HashSet?
One null element can be added to hashset.
TreeSet does not allow null values
==========
13. Explain Collection's interface hierarchy?

InterfacesHierarchy
==========
14.Which design pattern Iterator follows?
Iterator design pattern
==========
15.Which data structure HashSet implements
Hashset implements hashmap internally.
-==========
16.Why doesn't Collection extend Cloneable and Serializable?
List and Set and queue extends Collection interface.
SortedMap extends Map interface.
==========
=
17.What is the importance of hashCode() and equals() methods? How they are used in Java?
equals() and hashcode() methods defined in "object" class.
If equals() method return true on comparing two objects then hashcode() of those two objects must be same.
==========
18.What is difference between array & arraylist?
Array is collection of similar type of objects and fixed in size.
Arraylist is collection of homogeneous and heterogeneous elements.
==========
19.What is the Properties class?
Properties is a subclass of Hashtable. It is used to maintain lists of values in which the key and the value is String.
==========
20.How to convert a string array to arraylist?
ArrayList al=new ArrayList(Arrays.asList(new String[]{"java", "collection"}));
 arrayList.toArray(); from list to array
==========
Twice null added in set what will be happen.?