1) What is Collection ?

Ans A Collection Represent a group of Objects.

2) What is Collection FrameWork ?

Ans Collection frameWork is a Unified Architecture for Representing and Manipulating Set of elements called Collection.

3) Four Basic interface of Collection ?

Ans four Basic interface of Collection is as follow :-

1) Collection interface.

2) List interface

3) Set interface

4) Queue interface

4) which Interface Contain what ?

Ans Set Can Contain Unique object whereas List Can Contain duplicate object & Map Contain

Key and Value Pair.

5) what is Iterator ?

Ans Iterator is Used to Read data Sequentially from a Collection. Collection interface Contain Iterator() that returns Itrator object.
-> hasNext() -> next() -> remove().

6) what is Enumeration ?

Ans Enumration is just like a iteration, It reads data Sequentially from Collection objects. It is Used with Historical classes.

7) What is Java.Util.Pakage ?

Ans It is the pakage that Contain's all Utility classes and Interfaces like Vector, ArrayList, HashMap, HashTable, Set, Map.

8) Difference blw ArrayList and Vector ?

Ans - ArrayList is Asynchronous & Not ThreadSafe where as Vector is Synchronous and Thread Safe

- ArrayList is Used for Single User System Whereas

Vector is Used for Multi User System.

- ArrayList Provide high Performance where-as Vector Provide low Performance.
- Grow by of its halfSize whereas Vistor is Grows by double of its Size.

9) What is List Interface. ?

Ans List Interface extends Collection Interface. List Interface declares methods for managing an ordered Collection of object.

→ Key Implementation Classes of List Interface

1) ArrayList
2) LinKedList
3) Vector
4) Stack

→ Method of List Interface

1) add()

2) get().

3) Remove().

4) Set().

5) SubList (fromIndex, toIndex).

6) IndexOF().

7) LastIndexOF().

10) What is Set Interface ?

Ans Set Interface extends Collection Interface. Set can Contains unique set of object or elements, It does not allow duplicate Element to be Stored.
A Set Can Contain at most One null element.

→ Key Implementation classes of Set Interface :-

1) HashSet.

2) TreeSet. → 4) SortedSet (Interface).

3) LinkedHashSet.

→ Methods of Set Interface ?

Ans No additional Method of Set Interface.

Q.11 What is SortedSet Interface ?

Ans The Sorted Interface is a Child of Set Interface. Its element are automatically Ordered.

→ Method of SortedSet Interface ?

Ans 1) first()

2) last()

3) headSet()

4) tailSet()

5) SubSet()

12) What is Queue Interface ?

Ans Queue Interface Contain Queue of elements It provide additional operation like insertion, extraction and inspection.

Queue does not accept null element.

13) Key Implementation Classes of Queue Interface

1) LinKed List.

2) PriorityQueue

14) Method of Queue Interface ?

1) add()

2) remove()

4) element()

5) offer()

6) Poll()

7) PeeK()

15) What is Deque Interface ?

Ans "Deque" stands for "double Ended Queue"

Deque interface is Subtype of Queue interface

. It Represent a queue where you can insert and Remove element from both ended queue

16) Key Concrete class of Deque Interface ?

Ans 1) ArrayDeque

2) LinkedList

17) Method of Deque Interface ?

Ans 1) addFirst()

2) offerfirst()

3) addlast()

4) offerLast()

5) removefirst()

6) Pollfirst()

7) RemoveLast()

8) PollLast()

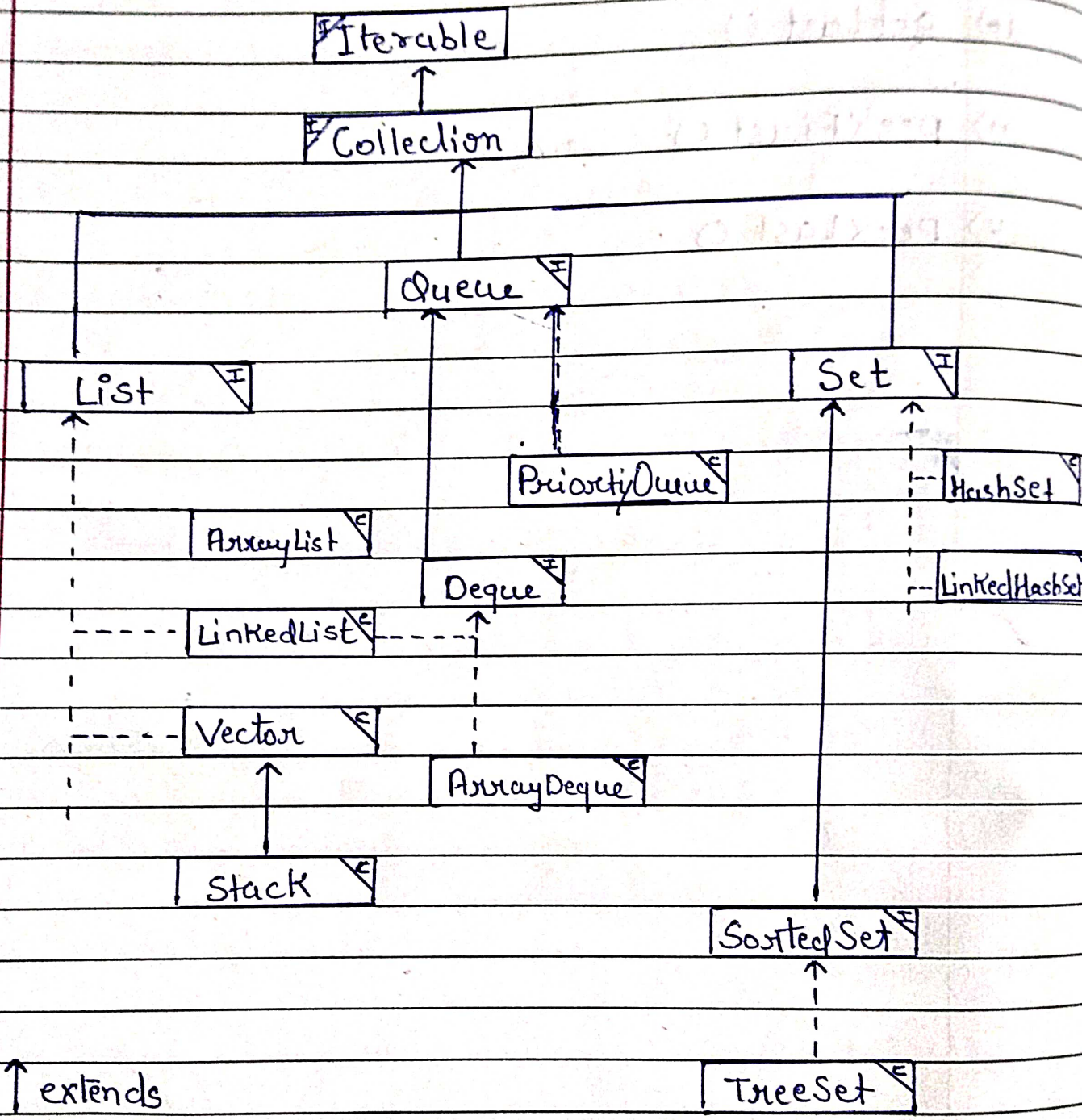
9) getFirst()

10) getLast()

11) peekFirst()

12) PeekLast().

18 Hierarchy of Collection Framework :-



Collection Interface Example :-

```
Public Class CollectionExm {
    Public Static Void Main(String[] args)
                {
    Collection a = new ArrayList();
    // adding elements Using add().
        a.add (101);
        a.add ("shubh");
        a.add ("saxena");
    SOP ("element of a" + " " + a);
    SOP ("Membership checking" + " " + a.Contain(101));
                // Output True (boolean)
    SOP ("List is Empty or Not" + a.isEmpty());
                // OutPut False
        a.clear();
    SOP ("clear" + a.clear); // Remove all
```

```
Collection b= new ArrayList();

    b.add("110");
    b.add("Ayush");
    b.add("Nigam");
  Object C= b.addAll(a); // Add Collection
x→SOP("Add a Collection "+ b.addAll(a))<-x

  SOP("Added Collection"+ C);

// For Iterate an object

   Iterator it = b.Iterator();

While (it.hasNext)
        {
          SOP(it.next);
        }
```

→ List Interface Example

```
Public Class ListExmp {

  Public Static Void Main (String[] args) {

      List a= new ArrayList();

      a.add (101);
      a.add ("ansh");
      a.add ("saxena");
      a.add ("Bhopal");

    SOP (a)
    a.add ((index) 1, (obj)"shubh");
    a.add (1, "shubh"); // add shubh at i-1

      SOP (a.get (0) // Value of index 0 (zero).
      SOP (a.set (1, "Pooja)) // replace object of
                                ansh to Pooja of
                                index - 1
      SOP (a.sublist (1, 4)); Print all object
                                b/w index 1 to
                                index 4.
      SOP (a.indexOF ("ansh")); // print the index
                                   of object ansh
      SOP (a.remove (2)); remove the obj of
                          index 2
      SOP (a.LastIndexOf ("saxena"));
                  // Search an element from end
                     and return Index
```

-> Set Interface Example

```
Public Class SetExam {

Public Static Void main (String[] args) {

    Set a = new HashSet ();
    Set b = new TreeSet ();

    a.add ("shubh");
    a.add ("a");
    a.add ("b");
    a.add ("b");

    SOP (a); // [a, b, shubh]

    SOP (b.isEmpty); // True

        }

    }
```

No additional Method in SetInterface it extends Collection Interface So we can use every method which is Present in Collection Interface.

-> SortedSet Example :-

```
Public Class SortedSetExam {

  Public Static Void main(String[] args) {

      SortedSet a = new TreeSet();

      a.add("a");
      a.add("c");
      a.add("b");
      a.add("d");
      a.add("e");

      SOP(a);          //[a,b,c,d,e.]
      SOP(a.first())   // [a]
      SOP(a.Last())    // [e]
      SOP(a.tailSet(c)) // [c,d,e] --->
      SOP(a.HeadSet(c)) // [a,b] <---
      SOP(a.Subset("a","e") // [a,b,c,d]

  }

}
```

Additional method of SortedSet Interface we use in example ie first(), Last(), tailSet(), HeadSet(), SubSet().

19) What is Map Interface ?

Ans It is an Object that Maps to Keys with Value. Map Contains Unique Keys. Each Key is Mapped with its Value. Two different Key May have Same Value.

20) Why Map is Not true Collection

Ans The Map Interface does not extends Collection Interface that's why it's not Called True Collection.

21) Key Implementation Classes of Map Interface

Ans 1) HashMap.

2) HashTable. (Historical Class).

3) LinkedHashMap.

4) TreeMap.

5) Properties. (Historical Class).

22. Method of Map Interface ?

Ans 1) Put() 3) ContainsKey 5) IsEmpty()

2) get() 4) ContainsValue 6) Size()

7) Remove

23) Historical Class of Collection frameWork ?

Ans 1) Vector

2) Stack

3) Hash Table

4) Properties

24) What is ArrayList ?

1) ArrayList is Resizable that implement List Interface.

2) ArrayList is internally creates an Array to Store elements.

3) It allows null Value to be inserted.

4) It gives you fast iteration and fast Random access.

5) It is Non-Synchronized and thread Safe.

24) What is Vector ?

Ans A Vector is same as an ArrayList, but Vector methods are Syncronized or thread Safe. Vector and arraylist both implement RandomAccess interface.

25) What is Fail-fast ?

if number of elements in a list are added or deleted after iterator is created without Using Iterator method then Iterator throw ConcurrentModificationException is called Fail-Fast.

26) Difference b/w ArrayList and Vector ?

Ans) ArrayList is NonSyncronized and not thread Safe whereas Vector is Synchronized and thread Safe

2) ArrayList is high Performance whereas Vector is low Performance.

3) ArrayList grow by of its half Size whereas Vector is grow by of its double Size.

4) ArrayList is Used for ~~Multi~~ single User System whereas Vector is Used in Multi User System

27) HashMap Vs Hashtable ?

Ans 1) HashMap is NonSynchronized and not thread Safe whereas Hashtable is Synchronise and thread Safe.

2) HashMap is Used in Single User System - whereas Hashtable is Used in MultiUser System.

3) HashMap is high Performance where-as Hash Table is low performance.

4) HashMap permits null Value where as Hash-table does not permits null Value.

28) ArrayList Vs LinkedList ?

	ArrayList	LinkedList
1)	ArrayList internally Create an Array to Store an element	LinkedList Consist of a chain of node to stored an element. both each element stored in node and pointer to the next node
2)	ArrayList Implements Random access Interface and Best for Random accers of element	LinkedList does not implement Random accers Interfas. Because it is Very Slow for random accers because it does Squential Search.
3)	By default new element is added at the end of array	By default new element is added as a Last node

4)

29) Set Vs List ?

	Set	List
1)	Set can Contain Unique set of element	List allow duplicate element.
2)	Set provide no Ordering of element	List provide ordered Collection of object.

30) HashSet Vs TreeSet Vs LinkedHashSet ?

Ans - HashSet Maintain any order of element.

- TreeSet Maintain element in Accending Order.

- LinkedHashSet maintains element in insertion Order

31) Iterator Vs Enumeration ?

Ans 1) Iterator has fail-fast whereas Enumration is not fail-fast.

2) Iterator has ability to remove an element from Collection whereas enumration Can not Remove element.

32) How Can you Synchronize a Map ?

Ans Using Collections.SynchronizedMap(c);

33) How Can you Synchronize a List

Ans Using Collection.SynchonizedList();

34) How Can you Synchronize Collection ?

Ans Using Collections.SynchronizedCollection();

40) Which type of Object are Compared by Comparable and Comparator object

Ans POJO, Vo, DTO

50) Method of iterator ?

1) hasNext()

2) next()

3) Remove()

51) Method of Enumration

1) hasMoreElement();

2) nextElement();

52) Method of Stack?

Ans) 1) Push();

2) Pop()

3) PeeK()

4) empty()

5) search()

53) Method of Collection

1) add() 2) addAll() 3) clear()

4) Contains() 5) ContainsAll() 6) isEmpty()

7) iterator() 8) remove() 9) removeAll()

10) Size()

54) Method of LinkedList

Ans 1) addfirst()

2) addLast()

3) getfirst()

4) getLast

5) Removefirst()

6) RemoveLast()

55) Which design Pattern is followed by iterator Interface.

Ans Iterator design Pattern

56) Difference blw Comparable and Comparator?

Ans 1) Comparable present in Java.lang Pakage whereas Comparator present in Java.Util pakage.

2) Comparable provides natural Sorting whereas Comparator provide Custom Sorting.

3) Comparable has CompareTo() method whereas Comparator has Compare() method

4) Comparable Compare the Current object with another object of same class whereas.

5) Comparator Compares the two different object of Single class.

57) How many Comparable and Comparator + Can be Created for class

Ans - One Comparable is for One class.

- number of Comparator for a class depends on number of ways we want to Sort our object.

58) What is generics ?

Ans generics is a Way to Communicate the type of a Collection to the Compiler.

59) Which Collection follows hashCode ?

Ans Hash Collection

60) Comparator & Comparable logics (-ve, 0, +ve)

Ans - (-)ve indicates the current object is smaller than second object

- (+)ve indicates the current object is greater than second object

- (0) indicates both are object is equal.

61) Why Map is Not true Collection ?

Ans Map Interface does not extends Collection Interface that why map is not true Collection.

62) What do you mean by Concurrent Collection

Ans Concurrent allows addition ~~or~~ and deletion of an element after creating iterator it does not throw Concurrent modification Exception.

63) Key interface of Collection framework ?

Ans 1) Collection 2) List 3) Set 4) SortedSet

5) navigableSet 6) Queue 7) Map 8) SortedMap

9) NavigableMap.

64) Internal Working of Hash Collection ?

Ans 1) Internally Hash Collection Using HashCode to Uniquelly identify an object while Saving or Searching an element.

2) Method Hashcode() is Used to Calculate and get the HashCode of an object. Process of Converting object into hashCode Called hashing.

3) Method equal() is Used to Compare two obj in hash Collection.

4) if you Override hashCode() method then Must Override equal() method for Consistant Comparision of object.

Q4) WAP of Comparable ?

→

```
Public class Student implements Comparable
    <Student> {

    int id;

    String name;

    String phoneNo;

  // Constructor :-

    Public Student (String name, int Id, String
                    PhoneNO) {

        this.Id = Id;

        this.name = name;

        this.phoneNO = PhoneNO;

    }

// override toString method

Public String toString () {

    return "Student[ id = "+id+", name = "name",
                PhoneNo = "PhoneNo"]";
}
```

```
// override CompareTo() method

Public int CompareTo(Student o) {

return this.name.compareTo(o.name);
                                    }
                                }
```

-> Public class StudentTestComparable {

```
Public Static Void main(String[] args) {

List<Student> list = Arrays.asList(new Student(1,
    "A", "222"), new Student(2, "D", "111"),
      new Student(3, "B", "444"),
      new Student(4, "F", "333"));

Collections.sort(list);

List.forEach(System.out::println);
                                    }
                                }
```

65) WAP of Comparator ?

→

```
Public Class Employee {

    Private int id ;

    Private String name ;

    Private int Salary ;

// generate getters for name and Salary

    Public String ~~name~~ getName () {

        return name

                    }

    Public int getSalary () {

            return salary ;

                    }

// override toString() method

Public String toString () {

return "Employee[id="+id+", name="+name+",
                    salary = "+salary"]" ;

                    }
```

→ Public class SortbyName implement Comparator<Employee>
{

// override Compare() method

Public int Compare (Employee O1, Employee O2) {

return O1.getName().CompareTo(O2.getName());

}

}

→ Public Class SortbySalary implement
Comparator <Employee> {

// override Compare() method

Public int Compare (Employee O1, Employee O2)
{

return if (O1.getSalary() == O2.getSalary())
{
return 0;
}
else if (O1.getSalary() > O2.getSalary())
{
return 1;
}
else {
return -1;
}
}

→ Public class SortEmpComparatorTest {

Public static Void main (String [] args)
{

List<Employee> list = Arrays.asList (
new Employee (1, "A", 200),
new Employee (2, "D", 300),
new Employee (4, "B", 500),
new Employee (3, "F", 400));

String S1 = "SortByName";

if (S1.equals ("SortByName"))

{
Collections.sort (list, new SortByName()
}
else
{
Collections.sort(list, new SortBySalary());
}

list.forEach (System.out::Println);

}

}

66) WAP for Concurrent Collection (Concurrent HashMap) implementation

Ans →

```
Public Class TestHashMap extends Thread {

Private Static ConcurrentHashMap<String, Integer>
        hm = new ConcurrentHashMap ();

Public Void Run () {
        hm.put ("four", 4);
}

Public Static Void main (String [] args) {

        hm.put ("One", 1);
        hm.put ("Two", 2);
        hm.put ("Three", 3);

TestHashMap t1 = new TestHashMap ();

        t1.start ();

for (object o : hm.entrySet ()) {
        System.out.println (o);
}
        System.out.println (hm);
}
```

4

67) WAP for Synchronized HashMap Implementation

→

```
Public Class TestHashMap extends Thread
{
Private Static Map<String, Integer> hm =
Collections.SynchronizedMap(new HashMap
<String, Integer>());
Public Void Run()
{
hm.put("four", 4);
}
Public static Void main(String[] args)
{
hm.put("One", 1);
hm.put("Two", 2);
hm.put("Three", 3);
TestHashMap t1 = new TestHashMap();
t1.start();
```

```
for (object o : hm.entrySet())
    {
    System.out.println (o);
    }
System.out.println (hm);
    }
    }
```

68) Key Interface of Concurrent Collection ?

Ans 1) BlockinDeque

2) BlockinQueue

3) ConcurrentMap

4) ConcurrentNavigableMap

5) TransferQueue

69) Key classes of Concurrent Collection ?

Ans 1) ArrayBlockingQueue.

2) SynchronousQueue.

3) LinkedBlockingQueue

4) LinkedBlockingDeque

5) LinkedTransferQueue

6) PriorityBlockingQueue

8) ConcurrentHashMap.

9) ConcurrentSkipListMap.

70) Synchronized Collection Vs Concurrent Collection

Ans 1) Synchronized Collection are Slow in Performance whereas Concurrent Collection are fast in performance

2) Only One thread can access the Collection at a time where as Multithreads can access the Collection at a time in Concurrent Collection.

3) Synchronized Collection locks the entire Collection. where as in Concurrent Collection, each element has a Separate lock.

4) Synchronized Collection permits null Value wehere as Concurrent Collection does not permit null Values.

-> In Synchronized Collection a thread lock the entire list, Only one thread access the entire list at a time.

-> In Concurrent Collection a thread lock one element, each element can be accesed by different thread Concurently.

7) WAP for hashCode() and equal() ?

→

```
Public Class Employee {
    int id;
    Public Employee (int id)
    {
        this.id = id;
    }

@Override
Public boolean equal (Object o) {
    if (this == o)
    {
        return True;
    }
    if (o == null)
    {
        return false
    }
    if (getclass() != o.getclass())
    {
        return false;
    Hashing or = (Hashing) o;
```

```
if(id != ot.id)
    return false
    return true;
}
}
```

```
@Override
Public int hashCode() {
  final int prime = 31;
  int result = 1;
result = prime x result +id;
  return result;
}
```

```
Public class EmployeeTest {

Public static Void main (String[] args) {

    Employee emp1 = new Employee (1);
    Employee emp2 = new Employee (1);

Map<String, Integer> map = new HashMap();

    map.put (emp1, "ansh");
    map.put (emp2, "ansh");

System.out.Println (map.size());

System.out.Println (map.entrySet());

    }

}
```

72) Exercise Question No. 8 Program of Stack

→

```
Public class Stack {

Public static Void main (String [] args)
                {

    Stack s = new Stack ();

      S.push ("Z");
      S.push ("A");
      S.push ("C");
      S.push ("D");

    System.out.Println (S);

String a = S.pop ();

String b = S.pop ()

String c = S.pop ()

String d = S.pop ();

      Stack S1 = new Stack ();

        S1.push (a);
        S1.push (b);
        S1.push (c);
```

```
        S1.push(d);
        System.out.println(S1);
    }
}
```

73) What is Stream in Java ?

Ans Stream is used to Create Sequential Streams for the given element. stream does not store data. It does only intermediate operations

74) Difference b/w Stream and Collection ?

Ans 1)- Stream does Not Store data whereas
- Collection Stores data.

2)- Stream is Read Only whereas
- Collection is Read and Write Both.

3)- If Stream Can Only be Read Once whereas
- Collection Can be Read Multiple times.

4)→ In Stream, element can not be directly accessed where as

- In Collection, element can be directly accessed.

74) 1) Create a Stream and Print all elements

```
List<String> iteam = Arrays.asList("One",
                    "Two", "Three", "Three", "four");
```

→ `iteam.stream().forEach(i->{SOP(i)});`

2) Sort the element and Print ?

→ `iteam.stream().sorted.forEach(i->{SOP(i)});`

3) Convert each element into Upper Case then print?

→ `iteam.stream().map(i-> i.toUpperCase()).forEach(so e->{SOP(e)});`

4) filter element starting with T character and Print

→ `iteam.stream().filter(e->e.startWith("T").map(ë-> i.toUpperCase()).forEach(System.out::println));`

5) Remove Duplicate elements.

→ iteam.stream().distinct().forEach(System.out::println);

List<Integer> l1 = Arrays.asList(2,4,3,10,50,67,69);

6) Get Collection of Stream.

→ List<Integer> list =

l1.stream().filter(i->(i%2==0)).collect(collectors.toList());

System.out.println(list);

o This is program of finding even number.

7) Intermediate Operations

An intermediate operation Transforms one Stream into another stream.

1) map().
2) filter()
3) distinct()
4) sorted
5) limit()
6) skip()

76) Terminal Stream operations.

Ans Stream are lazy, that means Sources and intermediate operation do not do any work Until a terminal is executed.

Terminal operations are :-

1) collect()
2) toArray()
3) Count()
4) Reduce()
5) min()
6) max()
7) anyMatch()
8) allMatch()
9) noneMatch()
10) findAny()
11) findfirst()