classmate
Date
Page

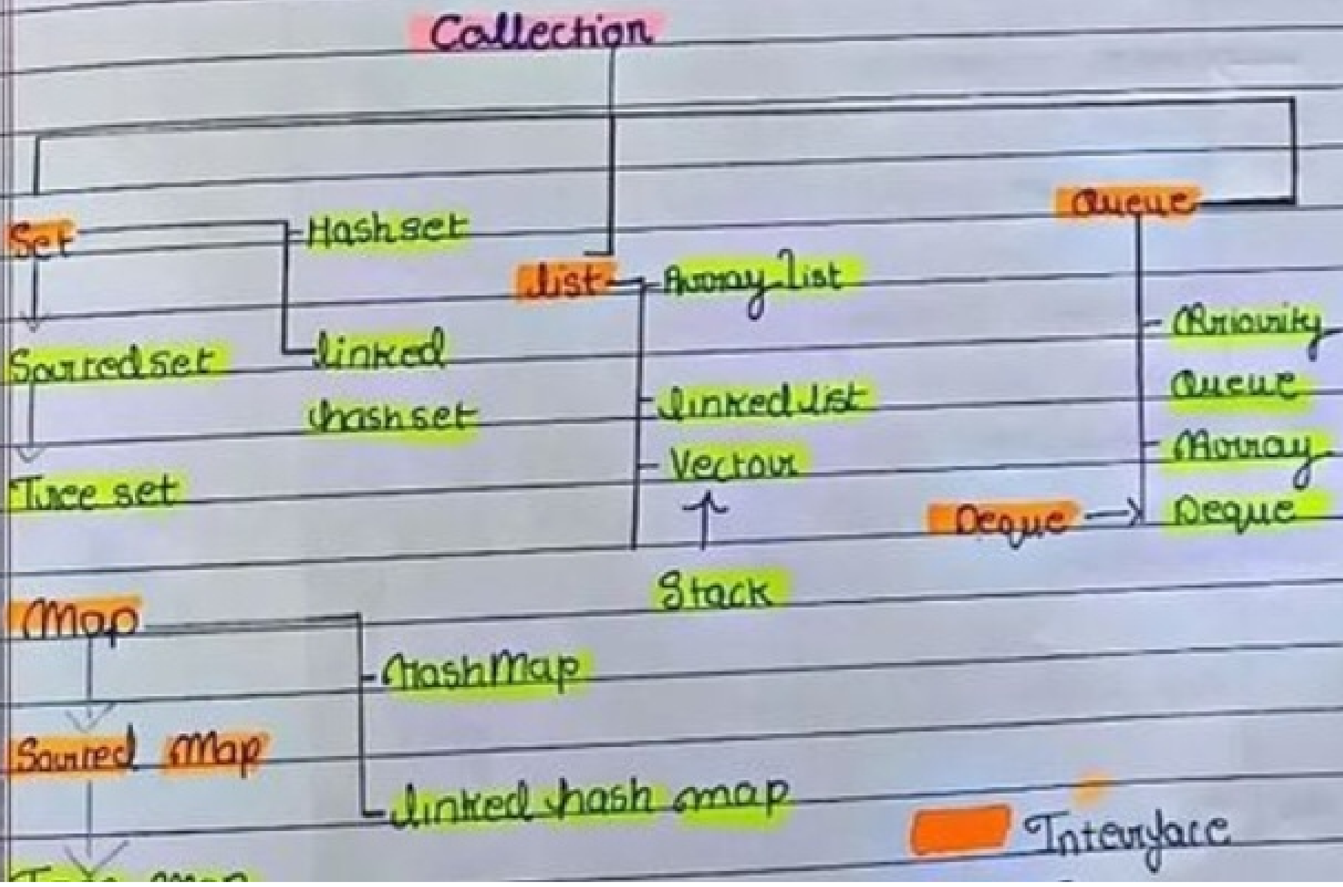
JAVA Collection's

(Short Notes)

★ What is java Collections?

→ The java Collections framework is a collection of interfaces and classes which make it easier for storing and processing collection of objects efficiently.

★ Hierarchy of collection framework



🟧 Interface

★ Advantages :-

(i) No third party API required.
(ii) No need to create API from handling collection objects.
(iii) Collection framework is tested and optimized.

★ Syntax of implementing collection

→ List < String> list = new ArrayList<>();

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

Set < String> set = new Hash set <>();

★ Types of method we can use in Collection :-

→ Add   Addall   Remove   Removeall   Removeif

Retain all   Size   Clear   Contains   Containsall

Iterator   toArray   Isempty   Equals   hashcode

★ Implement Iterator interface

→ Methods in Iterator

has next   next   remove

★ Implementation :-

```
-> public Class demo {
public Static void main (String [] args) {
List <String> list = new Array List<> ();
    list.add ("Akash");
    list.add ("Rahul");
    list.add ("Rohit");

    PrintList(list);
    remove (list);
   System.out.println ("After using remove")
    PrintList(list);

}

public Static void PrintList (List<String> list) {

 Iterator <String> iterator = list.iterator();

 while (iterator.hasNext()) {
   }

}

public Static void remove (List<String> list) {
Iterator <String> iterator = list. iterator ();
While (iterator.hasNext ()) {
     String item = (String) iterator .next () ;

     if (item .equals ("Akash")) {
         iterator .remove () ;
       }
   }
```

★ List Vs Set Vs Map

(i)	List allows to Store duplicate in java	Set does not allow to Store duplicate elements in java.	Map Stores data in form of key value pair it does not allow to Store dup-licate keys but allows duplicate values in java.
(ii)	List returns a ordered Collection data in java.	Set doesn't returns a ordered manner list in java.	Map also doesn't maintain ordered list in java.
(iii)	List allows To Store many null keys in java.	Set allows to add only one null key.	Map allows to add one null key and many null values.
(iv)	List are Resizeable array implementation of the java.util. List interface in java.	Set uses Map for their implementation Hence. Structure is map based and resizing depends on Map implem-entation. ex:- Hashset internally uses	Map uses hashing Technique for Storing key-value pairs.

* Comparatour Vs Comparable

(i) The comparatour provides multiple sourting sequences. In otheur words we can sourt the collection on the basis of multiple elements such as id, name and price etc.	Comparable provides a single sourting sequence. In otheur words, we can sourt the collection on the basis of a single element such as id name and price etc.
(ii) Comparatour doesn't affect the original class, ie, the actual class is not modified.	Comparable affects the ori... ginal ie the actual class is modified.
(iii) Comparatour provides compare() method to sourt elements.	Comparable provides CompareTo() method to sourt elements.
(iv) A comparatour is present in the java.util package.	Comparable is present in java.lang package.
(v) We can sourt list elements of comparatour type by Collections.sourt(list, comparatour) method.	We can sourt list elements of comparable type by Collections.sourt(list) method.

* Comparable and comparatour both are used four sourting list in a ourdeur eg:- Ascending and descending so we can whetheur use comparable our comparatour.. both will generate same output

☆ lets implement comparable and comparator interface on user defined class.

```
// This class implementing
// Comparable interface
/*
Class Student implements Comparable <Students> {
    int id;
    String name;
   public Student (int id, String name) {
    this.id = id;
    this.name= name;
 }
 @override
 public int compareTo (Student s) {
   if (this.id = s.id ) {
   return 0;
  } else if (this.id > s.id) {
     return 1;
  } else
  return -1;
}

@override
public String toString () {
   return id + " "+ name;
  }
} */
```

```
String name;

public Student (int id, String name){
    this.id = id;
    this.name = name;
}

@override
 public String toString (){
   return id+" "+ name;
 }

}
// This class implementing
// Comparator interface

class idComparator implements Comparator <Student>{
   @override
  public int compare (Student s, Student s1) {
     if (s.id = s1.id){
         return0;
     }else if(s.id>s1.id){
        return 1;
     }else
        return-1;
     }
}

public Class Demo {
   public Static void main (String[]args){
```

```
List <Student> list = new ArrayList<>();

list.add(new Student (2, "Akash"));
list.add(new Student(1, "Akash"));

four (Student s : list ){
    System.out.println(s);
}

System.out.println ("Afteur Sourting");

// This method is usable four comparable
// Collections . sourt (list);

// This method is usable four Comparatour
Collections . sourt (list, new idComparatour());

foun (Student s : list) {
    System.out.println(s);
}
}
}
```

Ouput:
```
2 Akash
1 Akash
Afteur Sourting
1 Akash
2 Akash
```

classmate
Date
Page

Follow
For
More