Java

1) Features of Java ?

- Pointers → Secure (cannot hack it easily)
- Keywords (53) we are using 49.

2) Interface is the prop' of Java which includes all 4 oop's concepts. (Java prop's)

*] Encapsulation :- [object existing into programming language = why oop's is object oriented.]

⇓ Package TestNG

```
Public class pen
{
  String brand;
  int Price;
  String colour;
```

→ states

for Behaviours we need to create non-static method.

```
  Public Void write ()
{
  sopln ("Used to write");
}

  MM
{
  Pen p = new pen ();
    P. write();
s.o.pln ( P.brand); sopln (P. colour); Sopln (P.price);
```

o/p

1) Used to write
null
null
0

This value is assigned by Compiler.

null bcoz = not assigned any value.

Scanned with CamScanner

```
Public Pen (String brand, int price, String Colour)
{
  this.brand = brand;
  this.price = price;
  this.colour = colour;
```

Pen p = new pen (10 Blue);

Rules of Encapsulation
1) Public 2) Atleast 1 Const
3) Private WebElement 3) only Getters

If i want brand to be constant

final String brand = "parker";

Real time ex
1) Def^ D/S^
2) Pom class for every Page.

```
Public class pen
{
  Private String brand;
  Private int price
  Private String colour;
  M.M
  {
    Pen p = new pen() {
      p.write();
```

Adv
* Code optimiz^.
* M & M of elements easier.
* Limit access = getters & setters.
* Inorder to handle StaleElement excep^.

Public void write()
{
sysO("U t w");}

```
Public String toString() {
  return "brand" + " " + "colour"
```

[toString = override] the values.

- In Encapsulation Datamembers are Private. Private members can be accessed using getters & setters by making class as Public.

Scanned with CamScanner

• Pom class where we have achieved encapsulation.

→ Public class pen {
Private webelement brand ;
Price
colour;

Public webelement getBrand() {
return brand;
}

— " — " — getprice() {
return price;
}

— " — getColour() {
return Colour ;
}

1) encapsulation ? How to achieve ?
2) where did u use encap in ur FW ? How ?
3) Why did you use encapsulation in FW ?
Ans :-
4) what is Pom class ? which concept of Java is used in pom class ?

Scanned with CamScanner

5) Can i inherit pom class?

=) Yes

6) Can i override pom methods?

7) Can we make getters & setters private or not?

8) — " — business logic — " —

9) How to access private data members of pom class in another class? => using getters method.

10) How to access private fn members of class in another class?

11) Inheritance and types of I.h? Where we have used I.h in our f.w?

12) What type of inheritance we have used in our F.W?

13) In how many ways we can call prop' of one class to another class?
=> 2 ways: 1) classname 2) Object

14) Can i achieve multiple Ih in Java?

15) Achieve Ih by using extends keyword?

16) Diff b/w super() & super keyword?
FW = No

17) — " — this(); & this keyword?

Scanned with CamScanner

*] Inheritance :-

Inheritance is nothing but acquiring Prop' of superclass to subclass using extends and implements keyword.

Types of Inheritance :-

1) single level Inheritance
2) Multi —"— —
3) Hierarchical —"— —
4) Multiple —"— —
5) Hybrid —"— .

⇒ Public abstract class super1
{
abstract void m1();

```
Public class Runner extends super1
{
    @override
    void m1();
```

i)
```
class super {
  super() {
    }
  }
class sub extends super {
    sub() {
    super();{
    }
  }
}
```

ii)
```
abstract class super {
abstract public void m1(); (for abstraction we need to give implementation).
{ super() { } }

class sub extends super {
    public void m1() {
    }
    sub();{
    super();}
```

Scanned with CamScanner

*) Polymorphism :-

```
Public class RCB
{
String Player1= "Virat Kohli";
String Player 2 = "MD Siraj";
String Player 3 = "Glenn maxi";

Public void batsmen ()
{

}

Public void bowler ()
{

}

Public void allrounder ()
{

}
}
```

- In our FAW, (Generic utility), Select, Ain we have used Method overloading, bulinell logic Method.
- In login page, Un, Pwd. & click on login. (POM)

1) Method overloading :

```
Public class RCB
{
  Public void m1 (int a) { sopln("a");
}
  -1 ——— (float f) { s.o.pln("b");
  }
  MM
{
RCB r = new RCB();
  r.m1(18);
  r.m1(18.8 f);
```

```
Public void m1(String s)
{
sopln("———");
}
      m1(object o)
{
sysu("————");
r.m1("Java");
r.m1(new object());
```

2) Method overriding :-

=> organ^z^n => Lalt name = Method overload.

In Vtiger = MO is used

GMail => Search bar -> Search name -> MO.

```
Public void m1 (StringBuffer o) {
  sysu ("———————")
  r.m1 (new StringBuffer());
```

Scanned with CamScanner

Constructor overloading :-

In our F.W We have not used CO.

In Selenium, alert popup is given by browser n/a to appl' itself it is n/a to CO.

17/05/22 :-

RunTimePolymorphism :- (Late binding) (Dynamic binding)

Overload method override in subclass.

subclass:

```
Public class super1 extends methods.
{
  Public void m1 (string s)
  {
    Sysout("
  }
  Public void m1 (StringBuffer o)
  {
    Sysout("  ")
  }
  MM
  {
    sub s = new sub();
      s.m1 ("Java");
    methods m = new methods();
      m.m1 ("Jp");
```

Scanned with CamScanner

methods mn = new sub(); (upcasting)

mn.m1("Java se"); → abstraction.

While compile Time point out to superclass

@ Run Time subclass.

Search for running object

• Visibility for users.

1) Disadvantage of Interface

```
Public interface class father
{
  Public void money();
  { sopln("Money belongs to father");
  }
  Public void vehicle(); {
    sopln("- ———— ");
  }

  Public void Property () {
    sopln(" —— ");
  }
}
interface
~~Public~~ class Mother
{
  ~~Public~~ void gold(); {
    sopln(" ——— ");
    money() { }
    vehicle() { }
```

Scanned with CamScanner

```
Public class child extends Implements Father, Mother {
    @override
    Public void money() {
        @override
            gold() {
            vehicle() {
            Property() {
```

Abstraction :-

```
class Samsung
{
    public void calling() { }
            texting() { }
            camera() { 4 Mp }
class Samsunggalaxy extends Samsung { }
Public void Pattern()
{
    }
}
class Samsung S20 Extends
{
Public void frontcamera ()
    }
Public void camera () {
    "8 Mp
}
```

Scanned with CamScanner

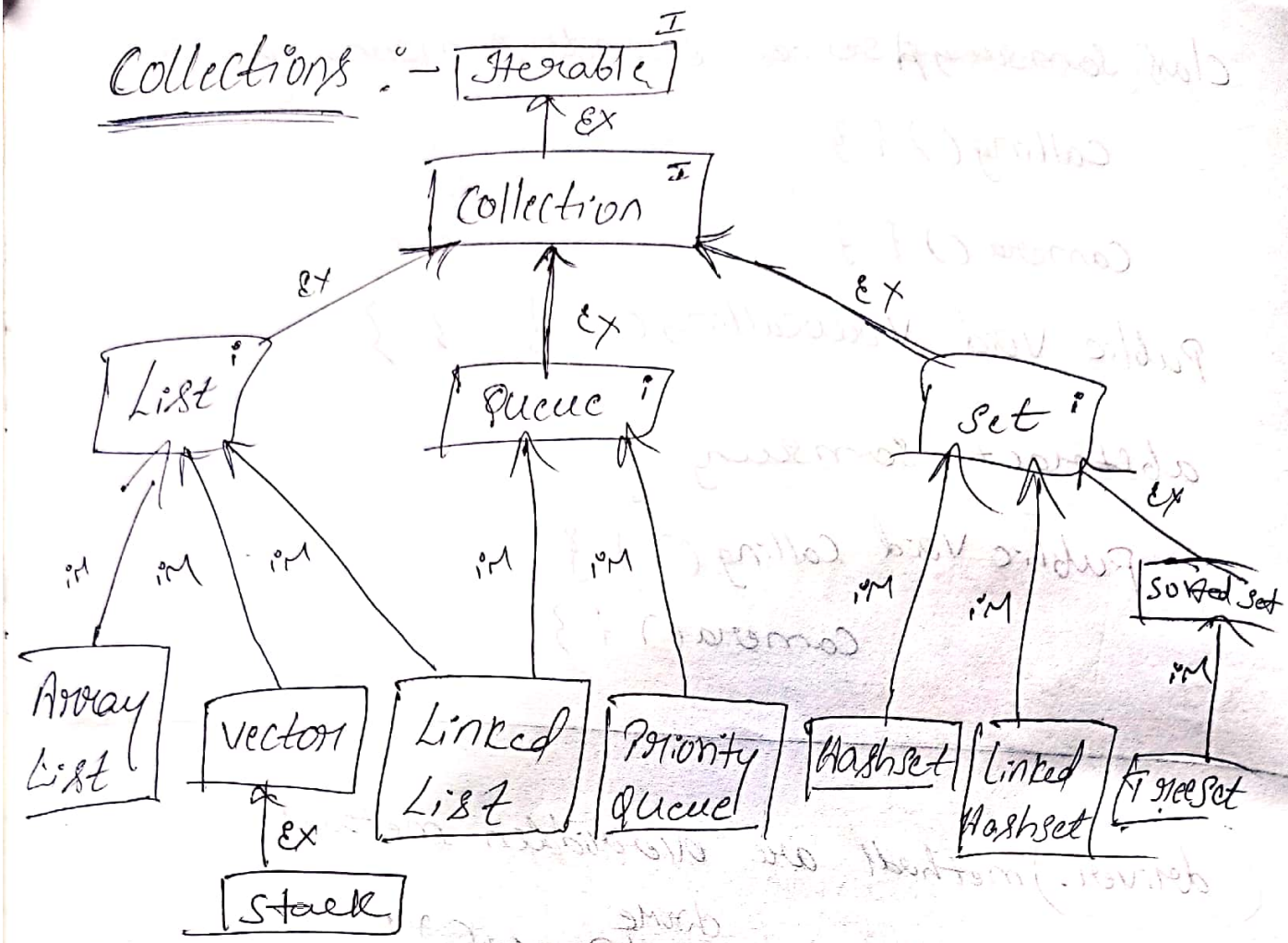
```
class SamsungA series extends samsung {
    calling() { }
    camera() { }
public void Videocalling() { } }

abstract samsung {
  public void calling() { }
          camera() { }
```

(driver.) methods are overridden methods.

```
WebDriver d = new ~~web~~chromeDriver();
```

Scanned with CamScanner

Collections :-



Disadvantages of Array :-

i) No inbuit fn's.

ii) Stores only homogeneous data

iii) Insertion & deletion is not possible

iv) Fixed length.

Adv of Collections :-

i) Collection are dynamic

ii) Stores heterogeneous.

iii) Contains inbuit fn's.

iv) Insertion & deletion is possible.

Scanned with CamScanner

Interview questions on Collection

1) Disadv of array?
2) Why do we go for collections?
3) Adv. of collections?
4) Diff b/w Array & Collection?
5) Parent of Coll^n interface.
6) Diff b/w ArrayList and LinkedList?
7) -||- — Hashset and Treeset?
8) -||- — List and Set?
9) -||- — Array and Arraylist?
10) -||- — Arraylist and Hashset?
11) -||- — Set and Map?
12) -||- — Hashmap and Treemap?
13) When do we go for Linked Hashset?
14) Diff b/w Hashset and Hashmap?
15) Where do we use Coll^ns in Your F.W?
16) Diff b/w Hashmap & Hashtable?

Scanned with CamScanner

List :-

* Insertion order is maintained.
* Duplicates are allowed.
* Allows null value.
* Stores heterogeneous type of data.
* Default capacity is 10.

Arraylist :-

* O.O.I is maintained.
* Duplicates are allowed.
* Allows null value.
* Stores heterogeneous type of data.
* Default capacity is 10.
* Data structure is dynamic.
* whenever we are searching the element in collection we go for arraylist.
* It has 3 constructors
  i) Arraylist ( )
  ii) Arraylist (collection ref)

Scanned with CamScanner

1) Public class Arraylist {

M.M {

```
Arraylist<object> list = new Arraylist<>();
    list.add('A');
           (10);
           ("hi");
           (10);
           (null);
    list.add(2, "Hello");
    for (object i1 : list)
    {
        sopln (i1);
    }
}
```

o/p

A
10
hello
hi
10
null

```
Arraylist<object> list1 = new Arraylist<>();
    list1.add('c')
           ('d')
    list.addAll(list1);
    sopln(list);
    sopln(list1);
}
```

o/p

[A, 10, Hello, hi, 10, null, c, d]

[c, d].

Scanned with CamScanner

```
if (list.contains('A'))
{
  System.out.println("Yes");
  }
else
  System.out.println("no");
  }
;
```

used to validate.

2) A.L < ~~object~~ Integer > list = new A.L < > ();

```
list.add (5);
   .add (20);
        (10);
        (1);

Collections.sort (list);
for (Integer obj : list)
{
    System.out.println (obj);
    }
}
```

Scanned with CamScanner

Vector :-

Vector is like a dynamic array which can grow (or) shrink its size ~~its size~~.

- We can store 'n' no of elements unlike array.
- There is no size limit.

Array List	Vector
* It is not synchronized	* It is synchronized.
* It increases its size by 50%.	* 100%.
* It is ~~Fast~~ Fast bcoz it is not synchronized.	* Slow bcoz it is synchronized.

3) ~~Arraylist~~

```
Vector<Object> v = new Vector<>();
v.
```

Scanned with CamScanner

Stack :-

- It is a linear Data structure. i.e., Used to store the collection of objects.
- It is based on last in first out.
- Stack is a class which extends vector class which belongs to Java.util Package.
- Push and pop method.

Push = It is used to push (or) add the data into the stack.

Pop = Fetch data one by one.

```
stack<object> s = new stack<>();
s.push(10);
s.push(20);
sopln(s.pop());
sopln(s.pop());
```

o/p

20

10

Scanned with CamScanner

Linked List :-

* O.O.I is maintained.
* Duplicates are allowed.
* Allows null value.
* Stores heterogeneous type of data.
* Default capacity is 10.
* ~~Default~~ Data Structure is doubly linked list.
* usage deletion and insertion of data.
* There are 2 constructors.

```
⇒ LinkedList <object> list = new LinkedList <>();
    list.add (10);
            ("hi");
            ('B');
    addfirst("Hello");
    addLast ("Bye");
    sopln (list);
          (list.getfirst());
          (list.getlast());
          (    .get (2));
```

Hello, 10, hi, B, Bye

Hello
Bye
hi

Scanned with CamScanner

*) Queue :-

- It is an interface which extends collection interface it has two subclass.
  i) Linked List
  ii) Priority queue.

→ PriorityQueue :-

It is a class which implements queue interface which is the Pure queue type.
To fetch the elements we use poll and Peek method.

Poll :- It will fetch the topmost element of the queue and remove the element of the queue.

Peek :- It will topmost element of the queue and it will not remove the element of queue.

Scanned with CamScanner

Set :-

- Set is a Interface
- O.O.I is not maintained.
- Duplicate values are not allowed.
- Default capacity is 16.
- Set is an interface which has 3 subclass.
  - i) Hashset.
  - ii) Linked Hashset.
  - iii) Tree Set.
- ~~Doesnot allow null value~~.

Hashset :- (follows O.O.I, remove Duplicate)

- Datastructure is hashtable
- It has 4 Constructors.
- We can add only one null has a value stores heterogeneous collection.
- Elements are displayed randomly and Duplicates are not allowed.
- h.add, h.addall, h.contains, h.remove

Scanned with CamScanner

· Iterator :-

Iteratorce interface of Java Coll^n F.W. allows us to access elements of a Collection.

· It has a subinterface i.e., List iterator.

· All the Java Collection include an iterator method. These method returns an instance of iterator used to iterate over elements of Coll^n.

· The iterator interface Provides 4 methods.

i) Hasnext () :-

Returns true if there exist an element in the Collection.

ii) next () :-

Returns the next element of the Coll^n.

iii) Remove () :-

. It removes the last element return by the next.

iv) for each remaining () :-

Perform the specified a/n for each remaining element of the Coll^n.

Scanned with CamScanner

```
Iterator <object> i1 = hs
    while (i1.hasNext() {
        {
        sopln(i1.next());
        }
    }
```

```
=> List list = new ArrayList<> (h);
    s.o.pln (list.get (2));
    }
}
```

- **LinkedHashSet :-**
  - It will follow O.O.I.
  - Duplicates removed
  - It has 4 Constructors.
  - We can add only one null ~~that~~ a value.
  - Stores heterogeneous collection.
  - Data structure is hashtable.

```
=> M.M
{
LinkedHashSet <object> lhs = new
    lhs.add ('A'); ("hi"); (null);
        (10); (null); (10);
    s.o.pln (s); }
```

Scanned with CamScanner

- **Sorted set** :- (It is an interface)

It is present in Java.util package it extends set interface. This interface contains methods inherited from the set interface and adds a feature that stores all the elements in this interface to be stored in a sorted manner.

```
SortedSet<object> set = new TreeSet<>();
set.add("Benz");
      ("Audi");
      ("Suzuki");
```

O/P
Audi
Benz
Suzuki

* Duplicates are not allowed
* Does not store heterogeneous value.

- **TreeSet** :- (It is autosorted) [Ascending order]
  - Datastructure is balanced Tree.
  - It has 4 constructors.
  - We can't add null as a value.
  - We cannot store heterogenous value, we can only store homogenous con^. If we try to store heterogeneous elements we get CCE.

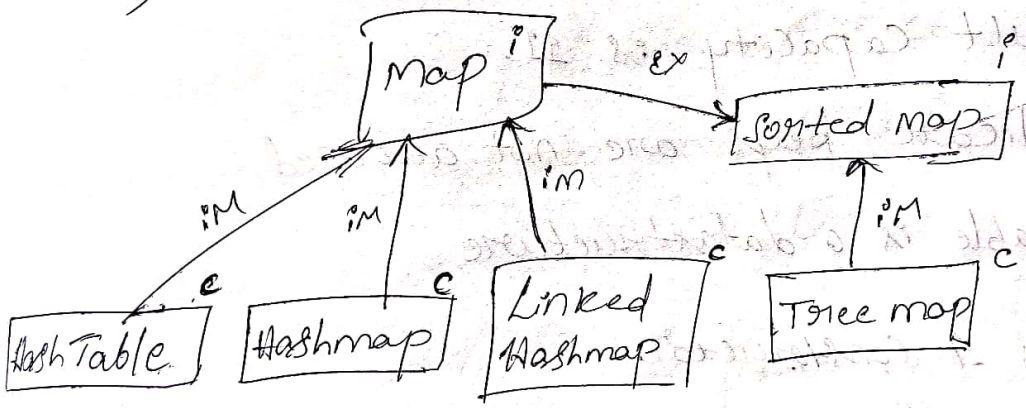
Scanned with CamScanner

```
M.M
{
TreeSet<object> t = new Treeset<>();
    t.add(20);
          (5);
          (10);
          (30);
          (8);
// t.add(null);
// t.add('A');
SYSO(t);
}
}
```

o/p
nullpointerexception.

o/p
classcastexception.

o/p
5, 8, 10, 20, 30



- Map contains values on the basis of key & value Pair each key & value Pair is known as "entry".
- Map contains unique keys.
- If you want to add any element we make use of Put method. elements are added in key & value Pair.

* O.O.I is not maintained.
* Duplicates are not allowed.

- Hashtable :- (Methods :- Put, PutAll, replace)
- If we want to add any element we use put method.
- Elements are added in key & value Pairs.
- O.O.I is not maintained.
- Doesnot allow null key or value.
- Hashtable class is synchronized.
- Default capacity is 11.
- Duplicate keys are not allowed.
- Hashtable is a datastructure.
- It has 4 constructor.

```
=> M.M
{
Hashtable<Integer, String> map = new HT<>();
map.put(10, "10");
        (20, "A");
        (30, "Hello");
        (null, "Hi");
sysO(map);
map.remove(30);
sysO(map);
}
```

O/P

{10=10, 20=A, 30=Hello, 40=Hi, 50=good}

{10=10, 20=A, 40=Hi, 50=good}

- Hashmap :-
  - If we want to use any element we use put method.
  - Key - value pairs.
  - O.O.I is not Maintained.
  - It has one null has a key & Multiple null values.
  - It is not synchronized.
  - Default capacity for hashmap is 16.
  - Duplicates keys are not allowed.
  - Data Structure is Hashtable.
  - It has 4 constructors.

O/P

{null=20, 20=null, 40=bye, 25=null, 10=A, 30=Hi}

Ex :- M.M

```
{
HM<Integer, String> map = new HM<>();
map.put(10, "A");
(30, "Hi");
(null, "Hello");
(null, "20");
(20, null);
(25, null); (40, null);
(40, "bye");
sopln(map);
```

```
for (Entry<Integer, String> m:map.entrySet())
{
Syso(m.getKey() + " " +"+m.getValue());
}
}
}
```

— o/p

20 null
40 bye
25 null
10 A
30 Hi

- **Linked Hashmap :-**

- If we want to add any element we use put method.
- Key Value Pair.
- O.O.I is maintained.
- It can have one null key, multiple null values.
- It is not synchronized.
- Default capacity is 16.
- Duplicate keys are not allowed.

Scanned with CamScanner

```
LHM <Integer, String> m = -
  m.put(10, "A");
    put(30, "Hi");
      (null, "Hello");
      (null, "20");
      (20, null); (25, null); (40, null); (40, "bye")

for (entry<Integer, String> m: map.entrySet())
{
  SYSO(m.getkey() + " " + m.getvalue());
```

		10	A	①
o/p :- null	20 ③	30	Hi	②
20	null ④	25	null	⑤
		40	bye	⑥

TreeMap :- (Ascending order)

- put Method
- Key & Value Pair
- O.O.I is Maintained.
- TM cannot have null key but can have multiple null values.
- TM is not synchronized.
- Duplicates keys are not allowed.
- Data structure is redblack Tree.
- It has 4 constructors.

Scanned with CamScanner

TM<I, S> m = _____

m.put (10, "A");

(30, "Hi");

(20, null); //map.put (null, "Hello");

(25, null); //map.put (null, "20");

(40, null);

(40, Bye);

for (ENTRY <

{

_____

}

<u>o/p</u> :-

10 A

20 null

25 null

30 Hi

40 Bye.

Scanned with CamScanner

Sorted Map :-

- It is a Interface.
- It Extends map interface & Provides a total ~~overriding~~ ordering of its elements parenthelis taken elements can be traversed in sorted order of keys.
- The class that implements this interface is a treemap.

⇒ M.M

```
SortedMap < I, S > m = new TM<>();
m.put (10, "A");
      (30, "Hi");
      (40, "Bye");
      (20, "Bye");
```

O/P

10	A
20	Bye
30	Hi
40	Bye

Scanned with CamScanner

Java Programs

1) Bubble Sort (Ascending, Descending)

2) 1st Min.no in an array.

3) 2nd Min.no in an array.

4) 1st Max no. in an array.

5) 2nd — " — ——

int[] a = {1, 2, 3, 4, 5, 6}

6) Last Min no in an array. (syso(a[a.length - 1]));

7) Last Max.no in array. (Descending order)

syso(a[a.length - 1]);

8) Both ascending & decending order.

3rd loop

```
for (int i=0; i<a.length; i++)
{
  syso (a[i])
}

for (int i=a.length-1; i>=0; i--)
{
  syso (a[i]);
}
```

Scanned with CamScanner

9) Sum of 3 no's in an array.

```
int sum = 0;
for ( int i = 0; i < 3; i++)
{
    sum = sum + a[i];
}
SYSO (sum);
}
}
```

10) Remove Duplicates in an array ?

5 - 1
9 - 1
5 - 1
4 - 1

* length — Arrays (can't use for string)
length () — String (— " — arrays)

:) String s = "Hello"
SYSO(a.length());

o/p
5.

:) For Each loop = ++ -- is not possible. (Disadv)
= Comparing the Values. (adv)

*) Print 9 to 10

```
M.M
{
for (int i = 9; i <= 10; i++)
{
    SYSO (i);
}
}
```

Scanned with CamScanner

iii) Print 1 to 10 without loop. (If i want to call method again & again make use of static)

⇒ ~~M.M~~ Public void num ()

{

```
Public static void num (int n)
{   if(n<=10) // 1<=10 (T); 2<10T
    Sop/n (n); //1, 2
    num (n+1); //1+1=2, 3
}
M.M
{
Printno pn = new printno();
  pn.num(1);
```

(or).

```
Public static void num (int a)
{
  if(a<=10)
  {
   S.o.p/n (a);
    a++;
  Print(a);
}}
M.M { Print(1);
```

to call
again
static)

```
Public static void num (int n)
{   if (n <= 20)
    sysO(n);
    num(n+1);
    }
}
M.M
{
obj. creation
m. num(10);
```

(10 to 20)

---

2) Print no's in reverse order

3) <u>Reverse a string</u> :-

```
String s = "Raghu";
{
for (int i=s.length()-1; i>=0; i--)
{
    sysO(s.charAt(i));
    }
}
```

Scanned with CamScanner

⇒ Reverse a string using rev :-

```
⇒ String s = "Raghu";
  String rev = "";
  for(int i = s.length-1(); i >=0 ; i--)
  {
    rev = rev + s.charAt(i);
  }
  Syso( rev);
  }
}
```

⇒ Rev. a string without using length method

```
String s = "Raghu";
char [] ch = s. tocharArray();
  int count = 0;
for( char c : ch)
{
  count ++;
}
Syso (count); // 7
for (int i = count-1 ; i >=0; i--)
{
  Syso (s.charAt(i));
  }
}
```

Scanned with CamScanner

```
String s = "gadag";
String rev = "";
for (int i = s.length()-1 ; i >= 0; i--)
{
    rev = rev + s.charAt(i);
}
SYSO (rev);
if (s.equals(rev))
{
  Sopln (rev + " is palindrome");
  else
{
  Sopln (rev + " is not a Palindrome");
}
}
```

A-E

$\frac{0}{10} = \infty$

Palindrome

```
int num = 454;
int sum = 0;
int rem = 0;
int temp = num;
while (num > 0)
{
    rem = num % 10;
    sum = (sum * 10) + rem;
    num = num / 10;
}
if (sum == temp)
{
SYSO( "Pal" );
}
else
{
SYSO( not );
}
```

Scanned with CamScanner

Method overloading :

Assert.equals = method overloaded

Reporter.log : — " — .

Custom overloaded :-

Strings

IMP Question :- imp

Group 1 :-

1) Find no. of occurance of each character in the given String.

2) Print only Duplicate characters in the String.

3) Remove duplicate — " — — " — .

4) Print only unique characters in the given String.

Group 2 :-

1) Find no. of occurance of each words in the given String.

2) Print only Duplicate words in the given String statements.

3) Remove Duplicate words — " — — " — .

4) Print only unique words in the given String statements ?

Scanned with CamScanner

4) occurance of each character

String s = "india";

s = inda (each & every char) for each

only loop for each (char ch : set) → (4 elements) (inda)

set → ref variable.

ref by ch

*) for each will iterate 4 times.

i - stored in ch (1st itt)

n - ch (2nd

d - ch (3rd

a - ch (4th)

int count = 0;

for (int i = 0; i < s.length(); i++) → Reading the char from string.

{

if (ch == s.charAt(i))

{

count ++;

}

Step 1) :- Read the data; for loop.

Step 2) :- Compare each char of set with all the char of given string. (first read)

for each loop

~~for~~

Scanned with CamScanner

```
for (character ch : set) //inda
{
    int count = 0;
    // reading element from con.
    for (int i = 0 ; i < Strength ; i++)
        // reading element from string.
    {
        if (ch == S.charAt(i)) // If Matching then increment then count
        {
            // If it is Matching q^e the count; so declare Count.
            Count ++;
        }
    }
```

[4 itteration can happen]

[i = 1st {Maintain O.O.I}]

Step 4) :- Print both ch & Count

```
SYSO(ch + " : " count +);
}
```

Group 2) : (imp) String[] str = S.split(" ");

P 1) :- Step 1) : Remove Duplicate word.

Step 2) :- Compare each word of set with all words.

```
for ( String string : set) // welcome to india.
{
    int count = 0;
    for (int i = 0; i < str.length ; i++)
    {
        if (string.equals(str [i]))
        {
            Coun ++; }}
```

(== compare integer, .equall = String)

SYSO(String + " " + Count])

Scanned with CamScanner

* if (count == 1) ==> unique char

* if (count > 1) ==> Print the duplicates.

~~Ans~~

Group 3 :-

1) Segregate character, sp char, no's in given string.

for ex :- String s = "Welcome to Tyss Bengaluru";

String str[] = s.split(" "); [return type = String array]

for (int i = str.length-1; i>=0; i--)

{

System.out.print(str[i]+ " "

2) Swap the 1st and last word in a given string ?

ASCII

A = 65 till 91 = Z 0 to 9 is = 48 to 57.

a = 97 till 122 = z

0 = 48, 9 = 57

3) a$#12@3

O/P = 6,

Scanned with CamScanner

3] Input = "aabbcab" ;
o/p = a2b2c1a1b1 ;

Prgm :- Assignment

1) WAJP to check whether given no is Prime or not.

2) Print Fibonacci no's till 25. (for(int i=0; i< { i<=25)

3) WAP sum of each digit of given no. (1234)

4) WAP to check whether the no is Palindrome or not.

5) String s = "india" ; I/P ; String S = "Hello" ;
o/p = i a o/p ; e o

Character	Word	unique = if(count==1) Print Duplicate = if(count>1)
1] for(int i=0;i<s.length;i++) { set.add(s.charAt(i)); } for(char ch : set){ int count=0; for(int i=0; i<s.length;i++){ if(ch == ~~strength~~ s.charAt(i)) { count ++; }} if	1] split ; String s = "welcome back back" String[]str=s.split(" "); for(int i=0; i<str.length; i++) { set.add(str[i]); } 2] inside for each while comparing ~~if(ch~~ for(String s : set) { int count=0; for(i:[i] { if(s.equals(str.~~length~~)){ count++; }	

Scanned with CamScanner

List = Links (All the links).
= Price of the Products.

custom = Ex:,
i) Arraylist
ii) Linked List

② Duplicate Contacts are present or not.
Assume 100 Contacts.

List <WE> list = driver.fe's (BY.xpath("..");
out of 100 Contacts. 10 duplicate (make use of Hashset).
Hashset S = new Hashset (list);
S.Size() → Method present in ~~Size~~ Set.

Ex:- List <We> list = dr.fe's (BY. xp
list.Size(); → 100

HS set = new HS (list);
Set.Size(); → 90

Win in F.W:-
way 1) :- Inbuilt f/n's SwitchWindow.

Prim

```
int n=11;
int p = 2;
while (n >=p) {
  if (n%p==0); {
      break; }
  else{
    p++ } }
```

```
if (n==p)
{ sysO("prime");
else
{
 sysO("not");
}
```

1	2	3	4	5	a[a.length]
0	1	2	3	4	

3] int a[] = {1, 2, 3, 4, 5}

o/p : 2 3 4 5 1 (Key 1)

int temp = a[0]

for (int i = 1; i < a.length; i++)

{

1 < 5 2 < 5 3 < 5 4 < 5 5 < 5

a[i-1] = a[i] a[0] = 2

a[1] = 3

a[a.length - 1] = temp; a[2] = 4

a[0] = 2

a[1] =

3] String s = "Welcome to india";

o/p :- Welcome To India (carelessness)

imp

4) String s = a11b22c33

o/p = 66.

Imp

5) String s = "india"

o/p = in id ii ia nd ni na di da

4) Swap the given no's without using 3rd variable

=> int a = 10;
int b = 20;

a = a + b; //30
b = a - b; //10
a = a - b //30 - 10 = 20 (a = a - b)

5) String S1 = "Java" S1+S2 = S1

String S2 = "selenium".

6) String S = "how are you";

o/p : You how are. (key)

(a=a-b)

5)

6)

1) Explain oop's along with your F.W ?

ADV

2) Why we follow Encapsulation rule to design pom class.

3) Diff b/w method overloading and method overriding.

4) Diff b/w Abstract class & interface

5) Diff b/w Array and ArrayList ?

6) Diff b/w HashSet and TreeSet ?

7) When do we go for Linked HashSet ?

8) Diff b/w Set and Map ?

Scanned with CamScanner

Programs :-

1) check whether given string is palindrome or not ?

2) find the no. of occurance of each character in the given string ?

3) I/p :- string s = "Welcome to india welcome to mandya".

o/p :- India mandya (unique char).

4) I/p :- string s = abbcccca

o/p :- a1b2c3a1

5) Find the 2nd largest no in the given array without using sorting ~~array~~ technique.

6) I/p :- int a[] = {2, 3, 4, 5, 6, 7, 8, 9, 10, 11}

o/p :- 2 3 5 7 11

Scanned with CamScanner