Exp. No. : .....

Date : .....

OOPS

OOPS - 1

①

Store 5 student no - ✓

Store 5 student name - ✓

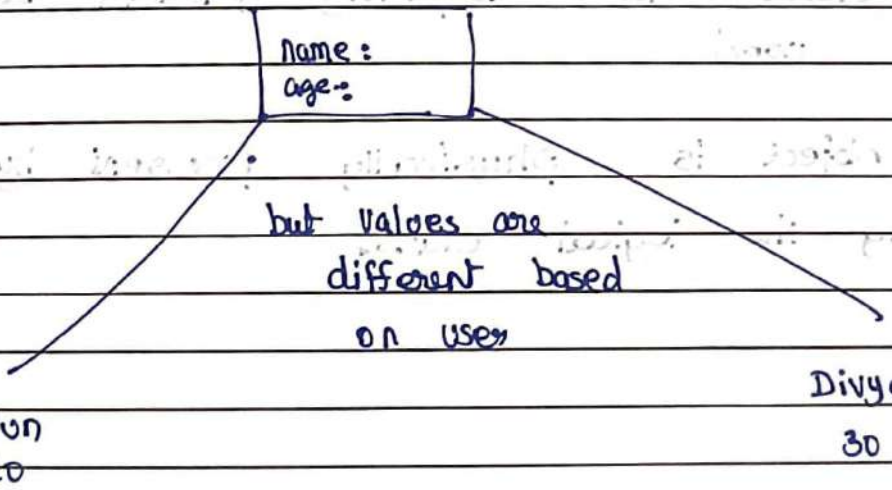
Now,

Create 5 data for students { name, roll, age } inside

↳ to Execute these kind of stuff Class is required.

Combining multiple entity inside one.

A template of same property



Basically class is not present
It does not have any reason to be in there unless
YOU HAVE AN OBJECT

Objects will initialize the class

More like

class as a blueprint for a new upcomming building

whereas

it has no value as a blue print

Unless it reaches an engineer who gives value to it

A class is logically present but not used

A object is physically present by using the logical class.

Exp. No. : .....

Date : .....

A object without a class : has no logic to be present.

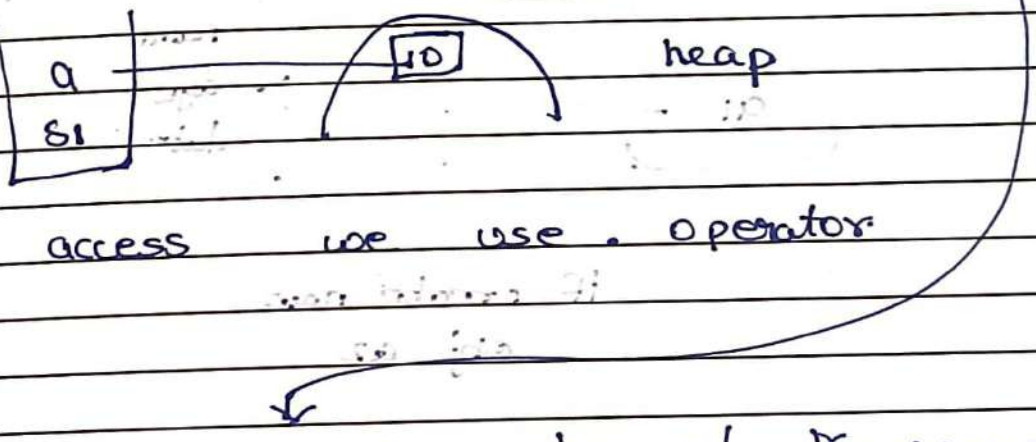
A class without an object , no value in a program, like no usage

x x

The variable created for the object is actually stored in stack memory like rest of the datatypes.

Ex :

```
int a = 10
Student s1 = new s1();
```



To access we use . operator

What exactly the 'new' Keyword

Page No.:

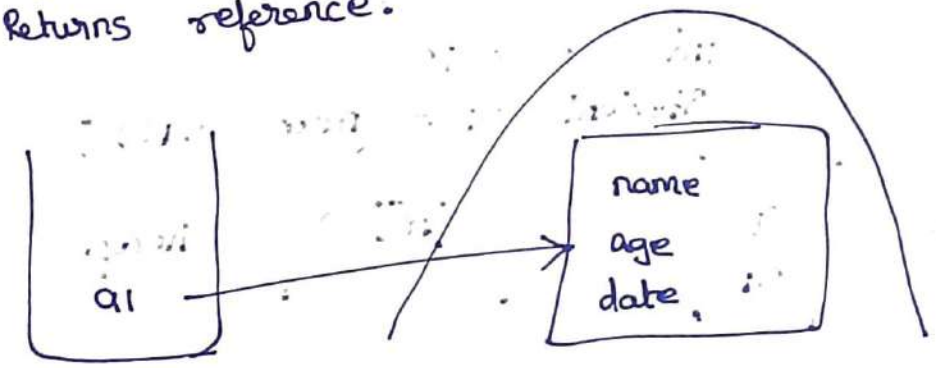
Ex: class a { }

a a1 ;
↳ reference variable

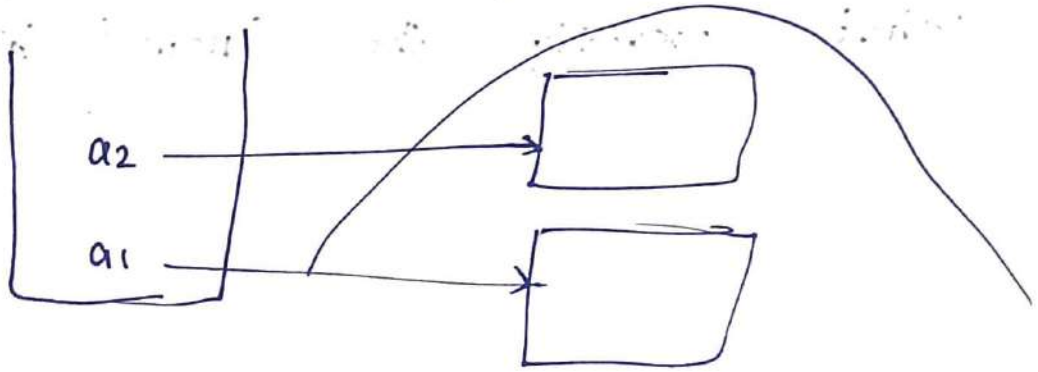
declaring a ref var for an object

a a1 = new a();

new - dynamically allocates memory & Returns reference.



if create new obj a2



Declaration # Initialization

here a a1 = new a();

work on compile time

work on Run time.

Works while writing code (or) Javac file.java

work on Run (or) java file.java

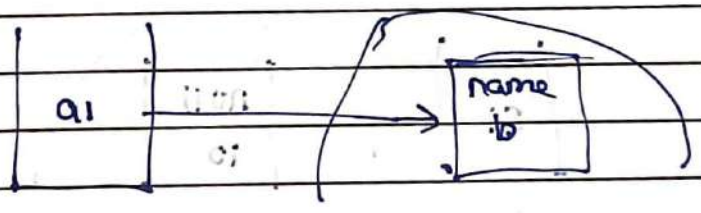
Ex:

a a1 = new a("Hello");

the hello will be only asigned when Run time. because of new keyword dynamic allocation.

```
class a {
    St name;
    Int b;
}
```

a a1 = new a();




```
Sout (a1.b);
// o/p = 0
a1.b = 10;
Sout // o/p = 10
```

Page No.:

Ex

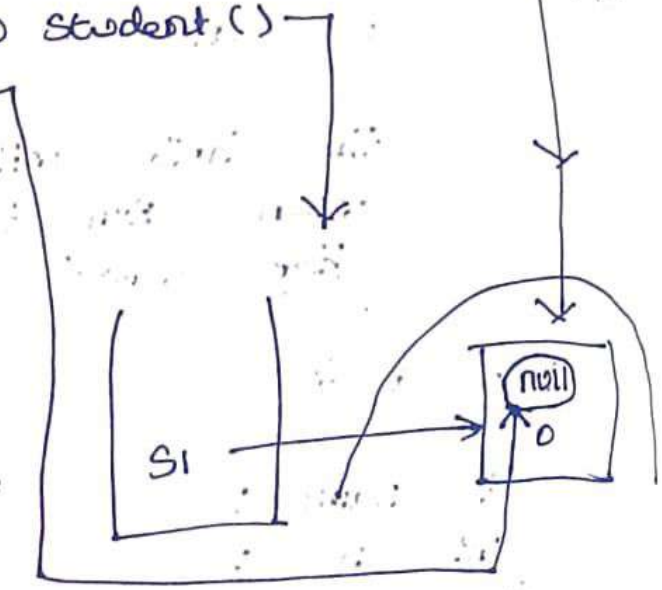
```
class Student {
    String name;
    int age;
}
```

Default

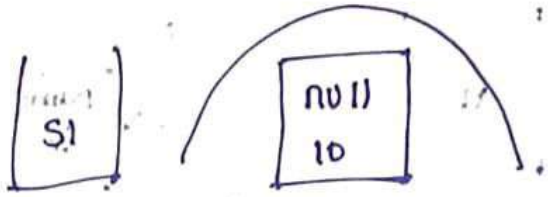
brings after initialization

After new initialization

(i) Ex Student S1 = new Student();
Sout (S1.name);



(ii) S1.age = 10;



Sout (S1.age);

Output : 10

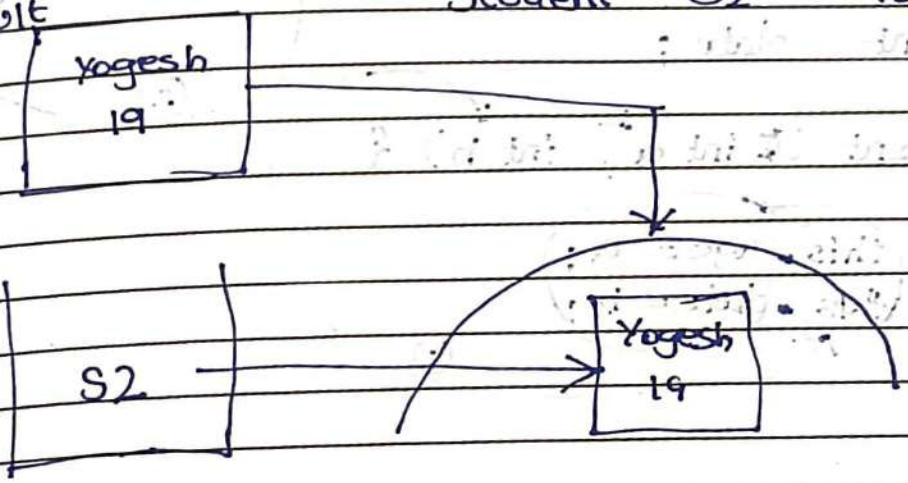
Exp. No. : .....

Date : .....

[Note : we can also manipulate the initialized value in class or blueprint]

Student S2 = new Student();

Default



S2.age = 20;

Sout (S2.age);

Output : 20

Now

Student S1 = new Student();

Calling class — R.Variable — Dynamic alloc — Constructor (default)

Responsible for initialization

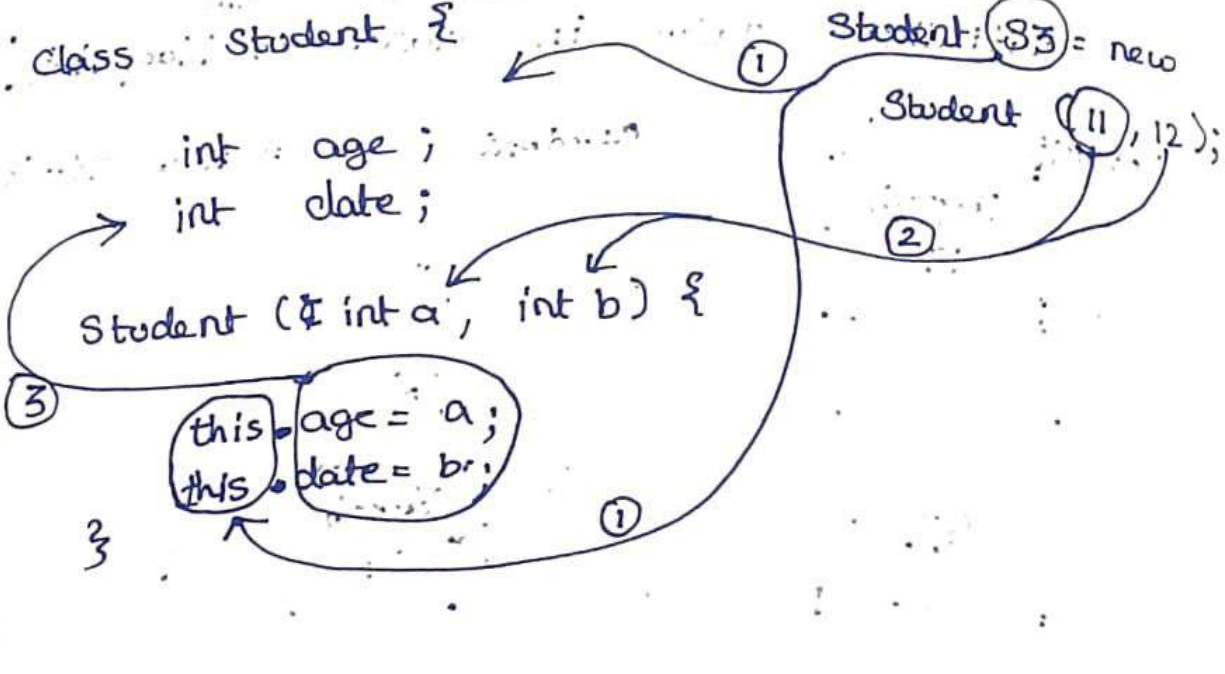
Page No.:

By Default

Student ()

But can change into

Ex ②



① : The S3 refers to the class but ② refers the methods value. So S3 is infered with 'this' key word so that the value of this very object will be assigned under S3 by a & b : to ③ age and date.

[REFERENCE VARIABLE IS REPLACED BY 'this']

Exp. No. : .....

Date : .....

without 'this' :

[this won't work but, 'this' means]

```
class Student () {
    int age ;
    int date ;

    Student (int a , int b) {        <-- (method)
        s3. age = a ;
        s3 . date = b ;
    }
}
```

```
Student s3 = new
Student (11, 12) ;
```

Internally `s3. age = 11` is replaced by `this . age = a` where `a = 11`

Page No.:

calling by object

Jake Ex - (2)

```
Class Student. {
    int age;
    Student ( Student l){
        this. age = l. age;
    }
    Student( int h.){
        this. age = h;
    }
}
```

```
Student h = new Student
            Student(
Student b = new Student
            (h);
```

l.age gets from h.age