OOPs - 1

```
// store 5 roll no ——
int[] roll_no = new int[5];
// store 5 name ——
String[] name = new String[5];
// store 5 marks —
Float[] marks = new Float[5];
```

Someone said

I need to store these things

what if we need to store all these ——

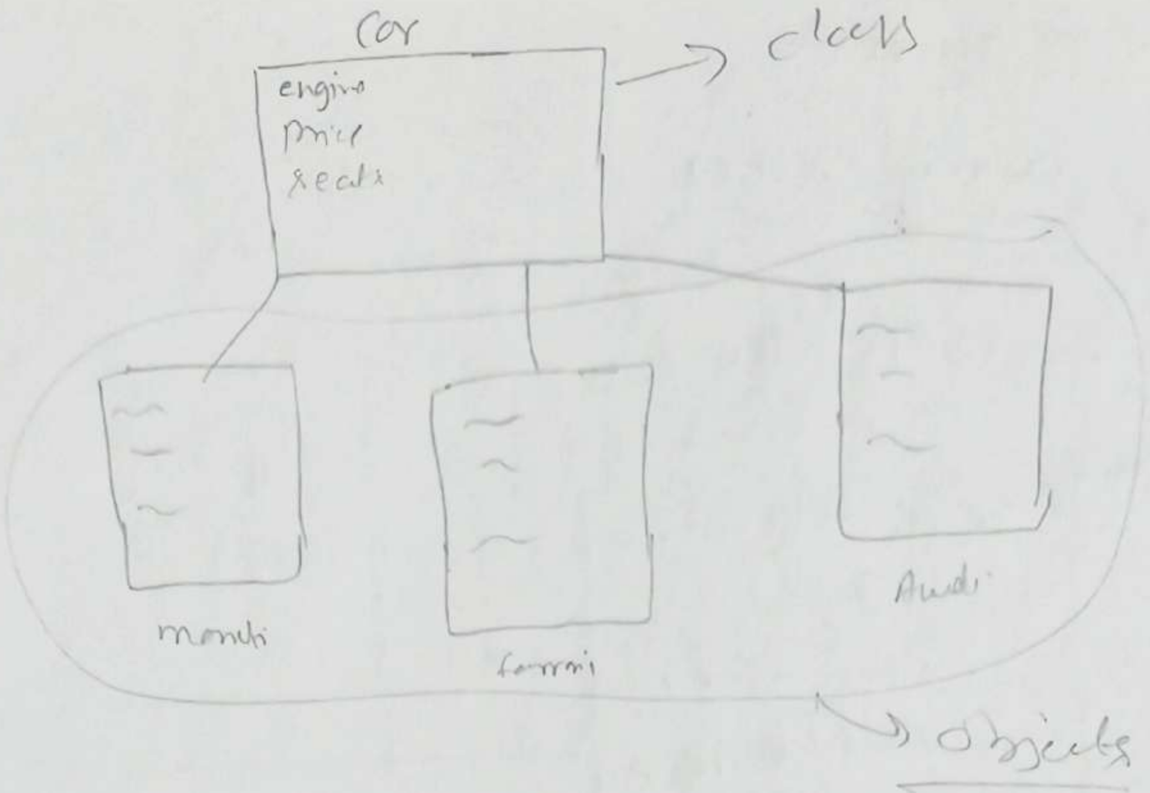
· we have classe ——

A class is a template for an object, and an object is an instance of a class.

```
class Student {
    int[] roll_no = new int[5];
    String[] name = new String[5];
    float[] marks = new float[5];
}
```

This is a specific data type we create ——

```
Student [illegible] = new Student [5];
```



class → logical construct
object → physical reality | → // occupy space in memory

→ State of the obj
→ Identity of object
→ behavior of object

Search →

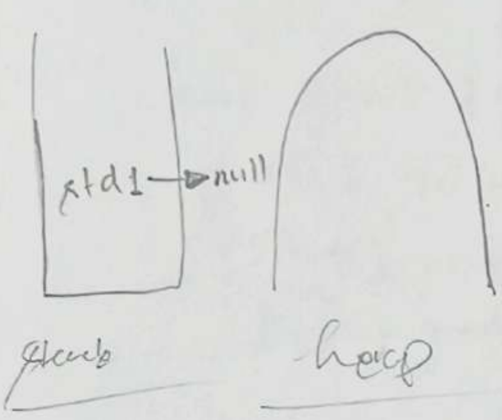
how to access —

using dot (.)

Ex- (Student1 . roll_no) → dot as a separator

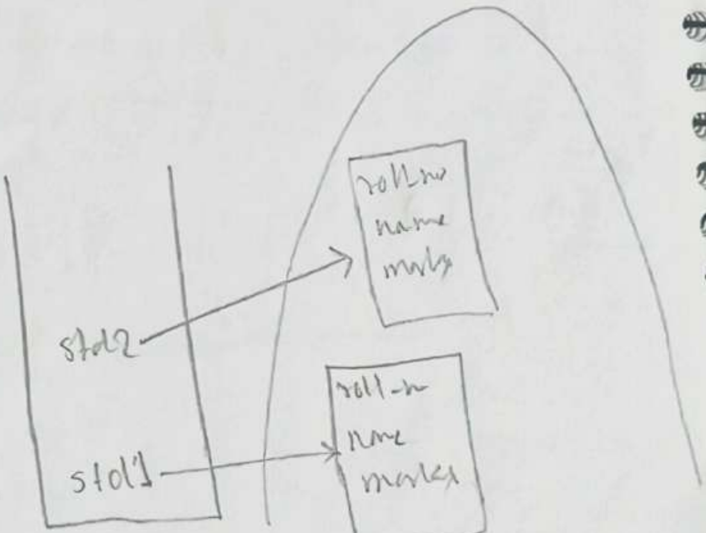
```
Student {
    int roll_no;
    String name;
    float marks;
}
```

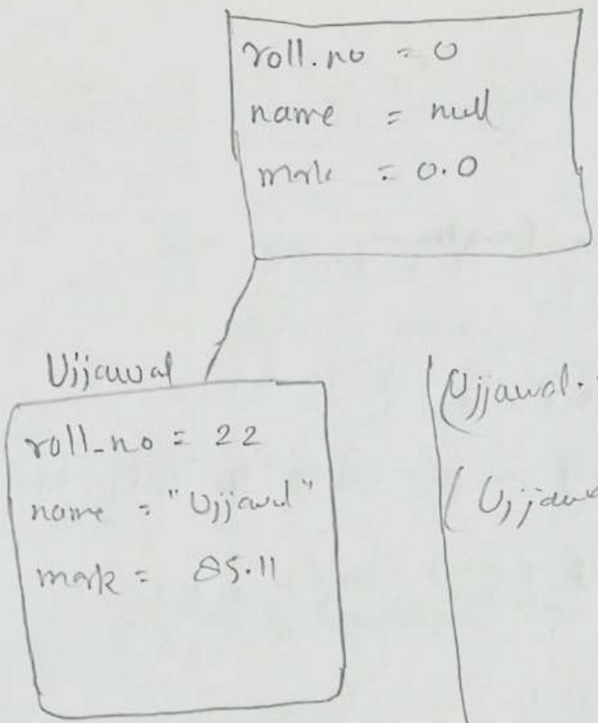
Student std1; → // declaring
it will in stack memory — pointing to null



std1 = new Student();

new → dynamically allocate memory & return a ref variable to it





$(\text{Ujjawal.roll\_no})_{\text{print}} \rightarrow 22$

$(\text{Ujjawal\_name})_{\text{print}} \rightarrow$ Ujjawal.

Constructor ? what happens when your object will be created.

Student std1 = new $\boxed{\text{Student();}}$

↓

constructor → By default Constructor.

As we did not created any function in class Student as Student().

" Constructor is a special function, that runs when you create an object & it allocate some variables. "

But, what if we need to do this —

Student std1 = new Student (22, "Ujjawal", 85.11);

↓

we need to create a function now

this.___

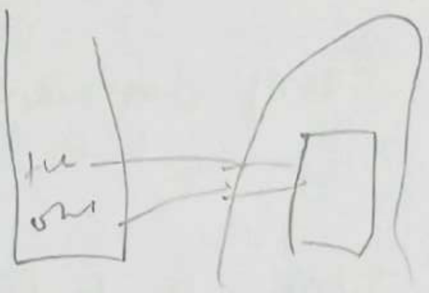
Calling Constructor from another Constru

```
Student {        public class ___ {
                   void main (String[] args) {
~~int name~~;          Student std1 = new Student(11, "ABC", 33.33f);
                       Student std2 = new Student();
                   }

                   class student {
                       int roll_no;
                       String name;
                       float mark;
                       Student () {
                          this(13, "default", 00.00f);
                       }
                       Student (int rno, String naam, float mark ){
                           this. roll_no = rno;
                           this. name = naam;
                           this. mark = mark;
                       }
                   }
```

Memory Allocation of 'new'?

```
Std one = new Student()
Std two = one
```



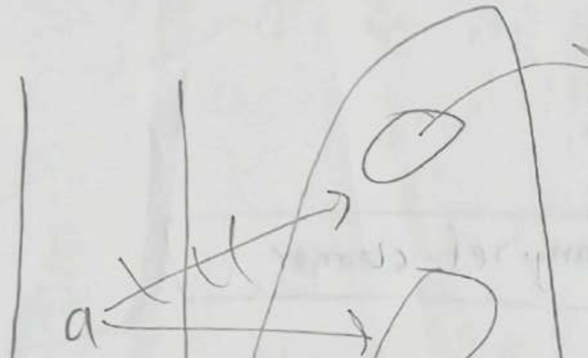
Wrapper Classe –

Integer, Boolean

"final"

```
final int min_age = 18;
```

Garbadge Collection –



Basics —

Autometic memory managment, in Java we don't mannually free() memory like c/c++.

The JVM's Garbage Collector (GC) reclaims memory from objects that are no longer reachable

An object is eligible for GC if no live thread can access it through any chain of reference from GC Roots —

- → local var in stack
- → Active threads
- → Static var
- → JNI reference

how finalize() works

↓

Every class inherits — protected void finalize() from object.

JVM call finalize() once before GC destroys the object.

Ex.

```
@override
protect void finalize() {
    System.out.println("Object is being destroyed")
```

But big issue —

i) unpredictable

ii) performance hit

iii) Dangerous — if finalize() resurrects the object, cleanup gets delayed.

iv) → Removed in Java 9

modern Version → java.lang.ref.Cleaner

? will learn after Interface ?