Class : Blueprint

Object : Instance of the blueprint.

Class → attributes
→ functionalities.

```
Class Car {
    Color ;
    size ;
    brand ;
    break() ;
    speed() ;
    music() ;
}
```

Marooof : Car
Color : red
size : 4
brand : Honda

Hitesh : Car
Color : red
size : 5
brand : Kia

break() ; ⟷ break() ;
speed() ; ⟷ speed() ;
music() ; ⟷ music() ;

```
# Class Student {
    int rollno ;
    String name ;
    int m1, m2, m3 ;
    int maxMarks() {
        return max(m1, m2, m3) ;
    }
    int totalMarks() {
        return m1+m2+m3 ;
    }
    Void printName() {
        S.O.P(name) ;
    }
}
```

Student S = new Student();

Dot operator

S.rollno = 123
S.name = "Hitesh"
S.m1 = 90
S.m2 = 85
S.m3 = 95

rollno = 123
name = "Hitesh"
m1 = 90
m2 = 85
m3 = 95

S.prithName() ⇒ Hitesh.

Object vs **Reference**

1) ✓ Student S = new Student()

S — reference of Student class.

new Student() — Creates the object of Student class.

S → #123468 → rollno, name, m1, m2, m3

#123468

2) Student S1 = new Student()

S1 → #43158
S3 → #43158

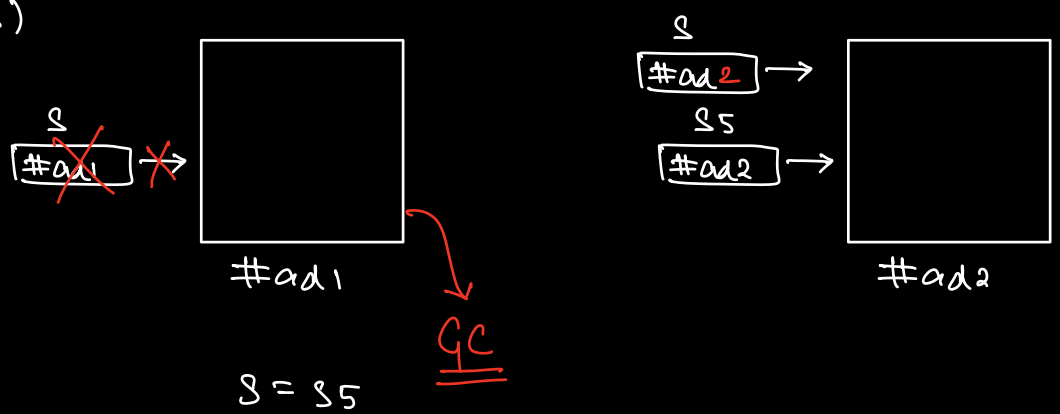
#43158 → rollno, name, m1, m2, m3

3) Student s2; ✕ In Java ✕

4) Student s3 = s1;

5) Student s4 = Null;
nothing.

6)



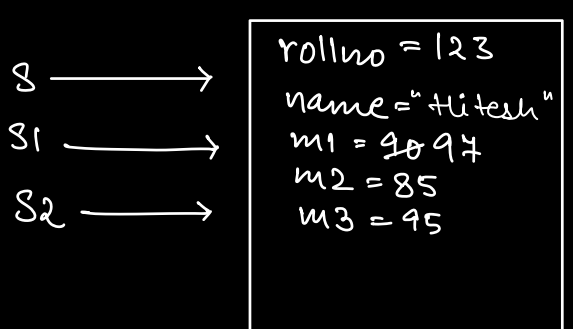
s = s5

Multiple Object reference

Student s = new Student();

s.rollno = 123
s.name = "Hitesh"
s.m1 = 90
s.m2 = 85
s.m3 = 95

Student s1 = s;
Student s2 = s;



S1.m1 = 97

S.totalMarks() ⇒ 95|97

S1.name = "Jeevan";

S.printName() ⇒ Jeevan

#

```
Class Pair {
    int x;
    int y;
    Pair(int x, int y){
        this.x = x;
        this.y = y;
    }
}
```

this

```
Pair p1 = new Pair();
// p1.x = default value
// p1.y = default value
```

p1.x = 10
p1.y = 20

```
Pair p1 = new Pair(10,20)
```

p1.x ⇒ 10
p1.y ⇒ 20

⇒ Constructor : It is used to initialise attributes of class at the time of object creation itself.

1) Name should be same as class name.

2) No return type

———— * ————