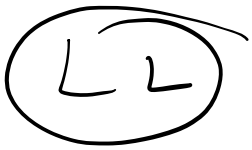
Class & Object ✓

OOP ✗

Syntax ✗



Class → It is a blueprint

Eg: floor plan of an Apartment!

Objects → Instance of a class!

Eg: Actual floor of an apartment!

One class can be used to create multiple objects.

Class → Attributes: to define the data.

→ Methods: to define the functionality.

```
class car {
    name
    color
    tyre
    |
    drive() {--}
    AC() {--}
    |
}
```

car: chinmay

Lambo
Pink
3
|

drive() {--}
AC() {--}

car: rajKumar

porsche
black
5
|

drive() {--}
AC() {--}

Same functionality across all objects!

```
class student {
    string name;
    int id;
    ⋮
    study() {--}
    bunk() {--}
    give exam() {--}
    ;
}
```

Attributes

Methods

```
student s1 = new student();
```

reference object, it contains the reference

[#1234]

name: "rahul"
id: 45

[#1234]

Memory allotted

```
s1.name = "rahul";
s1.id = 45
print(s1.name)
```

→ rahul

dot (.) operator is used to access attributes & methods!

```
s1.study();
```

```
student s2 = new student();
s2.name = "bitan";
s2.id = 21
```

name: ~~"bitan"~~ "sahil"
id: 21

[#4567]

```
student s3;
```

→ null

```
print(s3.name);
```

Null Pointer Exception!

```
student s4 = s2;
print(s2.name)
```

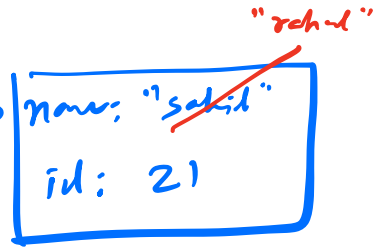
→ "bitan"

```
s4.name = "sahil";
print(s2.name)
```

→ "sahil"

// shallow copy!

```
Student s6 = new Student();
s6.name = s2.name;
s6.id = s2.id;
s6.name = "rahul";
print(s2.name); ⟶ "Sahil"
```



// DEEP COPY

Q Create a class Rectangle that supports:

1) find the area of the rectangle.
2) check if a rect. is a square!

```
class Rectangle {
    int l;
    int b;
    Rectangle(int l, int b) {
        this.l = l; this.b = b;
    }
    int area() {
        ret l x b;
    }
    bool isSquare() {
        ret (l == b);
    }
}
```

```
Rectangle r1 = new Rectangle();
r1.l = 10; r1.b = 20;
```

Constructor: method used to initialize attributes

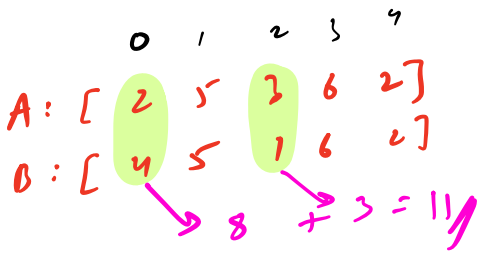
→ ret type: NONE

→ name: class Name

```
Rectangle r1 = new Rectangle(10,20);
Rectangle r2 = new Rectangle(5, 10);
```

Q. Given N rectangles with their L & B. in A[], B[]
(A[i], B[i]) → L & B of ith rectangle.
Find the sum of area of rectangles which are not square.

< Use the rectangle class >




```
Area = 0;   Rectangle r;
for( i: 0; i < N; i++) {
    Rectangle r = new Rectangle(A[i], B[i]);
    if( r.isSquare() == false) {
        Area += r.area();
    }
}
ret Area;
```

→ Add a method to check if area is:
1) greater than an int K.
2) greater than another rectangle.

class Rectangle {

f^n overloading

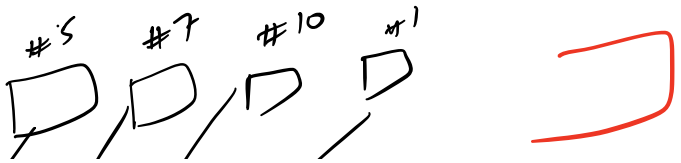
bool areaGreaterThan(int k) {
    ret (this.area() > K);
}

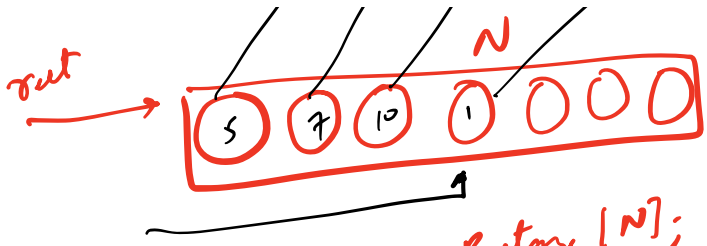
bool areaGreaterThan(Rectangle r1) {
    ret (this.area() > r1.area());
}

Q Given N rectangles with L & B in A[], B[].
(A[i], B[i]) → ith rect.

∀ index i, count the no. of squares on the left of i s.t. the area of the square is greater than the area of the ith rectangle!

	0	1	2	3	4
A:	2	5	3	6	2
B:	4	5	1	6	2
area:	8	25	3	36	4
ANS:	0	0	1	0	2

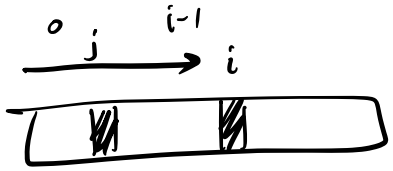




int A[] = new int[N];

Rectangle r[] = new Rectangle[N];

```
for( i=0; i<N; i++) {
    r[i] = new Rectangle( A[i], B[i]);
}
```

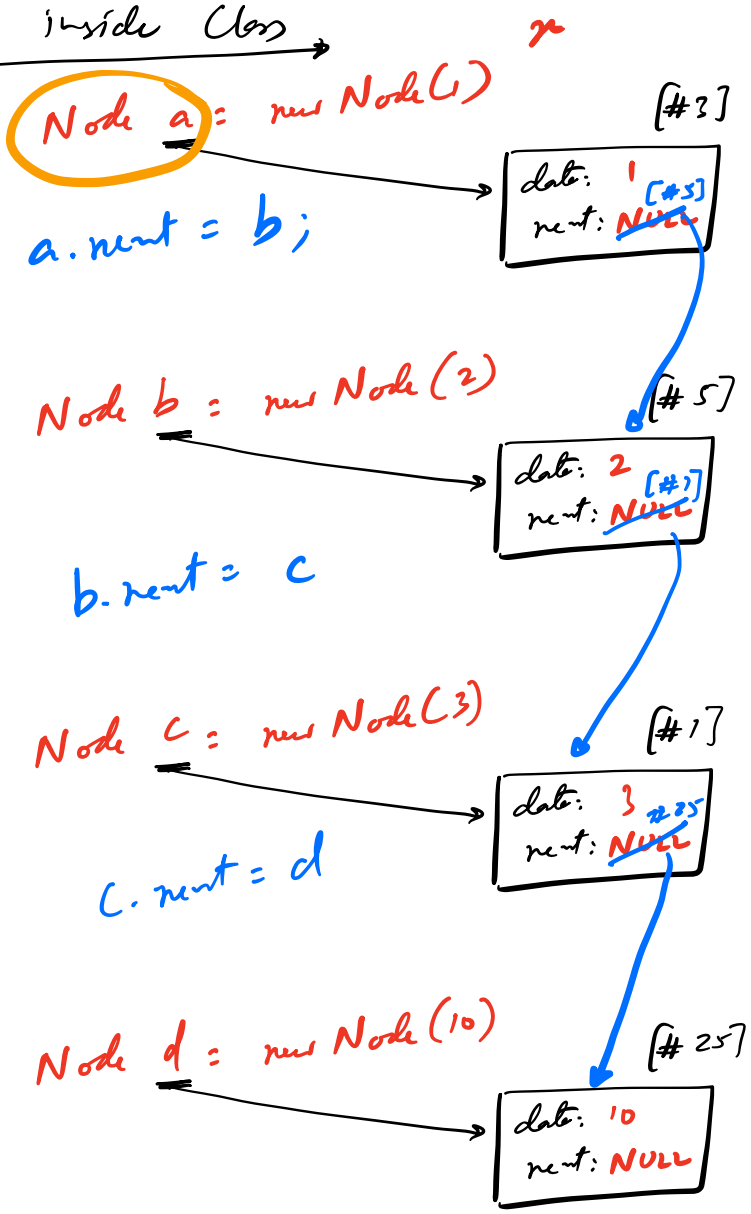



```
for( i=0; i<N; i++) {
    cnt=0
    for( j=0; j<i; j++) {
        if (r[j].isSquare()==true && r[j].areaGreaterThan(r[i])
                                                    ==true) {
            cnt++;
        }
    }
    print(cnt);
}
```

Object Reference inside Class

```
class Node {
    int data;
    Node next;

    Node(x) {
        data = x;
        next = NULL;
    }
}
```




```
Node x = a;
while( x != NULL) {
    print( x.data);
    x = x.next;
}
```