classes & objects concept ✓
OOPS ✗
Syntax ✗

class → It is a blueprint.
Eg → floorplan of an apartment/house.
Object → Real instance of a class.
Eg → Physical apartment/house.
(one class can have multiple objects.)

class → Attributes to define data.
class → Methods to define functionalities.

class Car {	Car : Uday	Car : Sairam
name	name → Jeep Compass	name → Ertiga
color	color → Galaxy Blue	color → Blue
milage	milage → 10 km/L	milage → 17 km/L
⋮		
drive () {...}	drive () {...}	drive () {...}
AC () {...}	AC () {...}	AC () {...}
⋮		
}		

object

same functionalities in all objects.

```
class Student {
    String name
    int    id
           ⋮

    study () {...}
    bunk () {...}
    exam () {...}
    ⋮
}
```

attribute: String name, int id

methods: study () {...}, bunk () {...}, exam () {...}

Student s1 = new Student();

[#2368]
object reference of Student class.

name = "Gaurav"
id = 32

[#2368]
Memory address

s1.name = "Gaurav"
s1.id = 32
↳ dot to access attributes & methods.
s1.study()

Student s2 = new Student();
s2.name = "Dhawal"
s2.id = 21

name = "~~Dhawal~~" "Amal"
id = 21
[#1234]

Student s3;
null
object reference of Student class.
print(s3.name); →
Null Pointer Exception Error!

Student s4 = s2; // ~~Shallow Copy~~
[#1234]
print(s2.name) → Dhawal
s4.name = "Amal"
print(s2.name) → Amal

Student s5 = s4;
s2.name = "Sagar"
print(s5.name) → Sagar

Student s6 = new Student();
s6.name = "Sairam";
s6.id = 87;

name = "~~Sairam~~" "Shayam"
id = 87
[#6741]

Student s7 = new Student();
s7.name = s6.name;
s7.id = s6.id;

name = "Sairam"
id = 87
[#7832]

print(s7.name) → "Sairam" // ~~Deep Copy~~
s6.name = "Shayam";
print(s7.name) → "Sairam"

Q → Create a class Rectangle that supports following functionalities → 1) find the area of a rectangle.
2) check if the given rectangle is a square.

```
class Rectangle {
    int l, b;
```

```
Rectangle r = new Rectangle();
    r.l = 4;
    r.b = 6;
```

l = 4
b = 6

```
Rectangle (int x, int y){
        l = x;    b = y;
}
int area (){
        return l*b;
}
boolean isSquare(){
        return (l == b);
}
}
```

Constructor → Method used to initialize the attributes of the class at time of object creation.

1) name is same as class name.
2) no return type.

Rectangle r = new Rectangle (4,6);

Q → Given N rectangles with length & breadth in A[] & B[].
(A[i], B[i]) ⟶ ith rectangle.
Find the sum of area of rectangles which are not square using Rectangle class.

```
ans = 0
for i → 0 to (N-1)
    Rectangle r = new Rectangle(A[i], B[i]);
    if ( ! r.isSquare())
        ans += r.area();
return ans
```

A = [2 5 3 6 2]
B = [4 5 1 6 2]
Area = 8 + 3 = 11 ✓

readable ✓
reusable ✓

```
Rectangle r;
ans = 0
for i → 0 to (N-1)
    r = new Rectangle(A[i], B[i]);
    if ( ! r.isSquare())
        ans += r.area();
return ans
```

```
class Rectangle {
    int l, b;
    Rectangle (int x, int y){
        l = x;   b = y;
    }
    int area () {
        return l*b;
    }
    boolean isSquare () {
        return (l == b);
    }
}
```

Add a method to check if area is $\rightarrow$ 1) greater than an int K.
2) greater than another Rectangle.

this / self

```
boolean areaGreaterThan (int K) {
    return this.area () > K;
}
```

Method overloading

```
boolean areaGreaterThan (Rectangle R) {
    return this.area () > R.area();
}
```

Q → Given N rectangles with length & breadth in A[ ] & B[ ].
(A[i], B[i]) ⟶ i^{th} rectangle.
∀ index i, count the number of squares on the left of i s.t area of square is greater than the area of current rectangle.

```
// Java
int a[] = new int [N];
Rectangle R[] = new Rectangle [N];
custom datatype

for i → 0 to N-1
    R[i] = new Rectangle (A[i], B[i]);
```

	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

area1 > area2
25 > 3

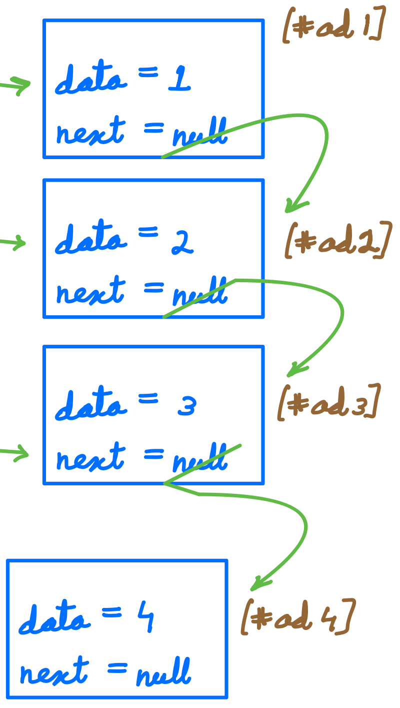
```
for i → 0 to (N-1)
    ans = 0
    for j → 0 to (i-1)
        if ( R[j]. isSquare() && R[j]. areaGreaterThan(R[i]))
        ans += 1
print(ans)
```

Object Reference inside a class

```
class Node {
    int data;
    Node next; //obj. reference
    Node(int x) {
        data = x;
        next = null;
    }
}
```

```
Node a = new Node(1);
Node b = new Node(2);
a.next = b;
Node c = new Node(3);
b.next = c;
Node d = new Node(4);
c.next = d;
```

Linked list



data = 1
next = null
[#ad 1]
data = 2
next = null
[#ad2]
data = 3
next = null
[#ad3]
data = 4
next = null
[#ad 4]