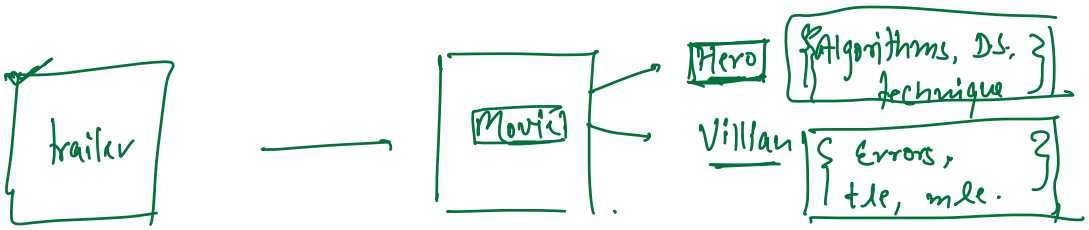
A LITTLE PROGRESS EACH DAY ADDS UP TO BIG RESULTS



→ classes & objects concept ✓

→ OOPS ✗

→ syntax [search]

Class → It is a blue-print.

Eg → floor plan of apartment / building / house.

Object → Real instance of a class.

Eg → Physical apartment / building / house.

→ [For one class, we can have multiple objects.]

Class → **Attributes** to define data.

→ **Methods / Functions** to define functionalities.

```
Class Car {
        name
        color
        price
        mileage
        :
        drive ( ) {     }
        AC ( ) {     }
        music ( ) {     }
        :
}
```

Car : Meenakshi

name → Scorpio
color → Black
price → 16
mileage → 7 km/L

drive ( ) { }
AC { }
:

Car : Atharva

name → Lamborgini
color → Blue
price → 7 cr.
mileage → 3 km/L
:

drive ( ) { — }
AC ( ) { -- }
}

[Same functionalities will be there for all the objects.]

```
class Student {
```

Attributes
```
    String name
    int id
    :
```

functions/methods.
```
    study () {--}
    bunk () {--}
    exam () {--}
    sleep () {--}
}
```

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

{#2368}

object reference of student class.

name → "Gaurav"
id → 123

#2368) Memory address.

```
s1.name = "Gaurav"
s1.id = 123
```

→ [dot to access attributes & methods.]

```
s1.study()
```

```
Student s2 = new Student();
s2.name = "Meenakshi"
s2.id = 456.
```

name = ~~"Meenakshi"~~ "Atharva"
id = 456

#4597

```
Student s3 = s2;
```

[#4597]

shallow copy

```
Student s3; {null}
```

[object reference to student class.]

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

⇓

{Error!! Null Pointer Exception}

print(s3.name) → Meenakshi

s3.name = "Atharva"

print(s3.name) → "Atharva"

print(s2.name) → "Atharva"

```
Student s4 = new Student();
    s4.name = "Abdul"
    s4.id = 789
```

name → ~~Abdul~~ Mayank
id → 789

2378

```
Student s5 = new Student();
    s5.name = "Abdul"   // s5.name = s4.name
    s5.id   = 789       // s5.id = s4.id
```

name → Abdul
id → 789

4523

```
print(s4.name) → Abdul
s4.name = "Mayank"
print(s4.name) → Mayank
print(s5.name) → Abdul
```

Deep Copy

```
int a = 10;
Student s3;
    s3.name = "Harsh"
```

Q) Create a Rectangle class that supports following functionalities →

① Find the area of the rectangle.

② check if the rectangle is square or not.

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

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

l → 4
b → 6

#2546

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

① name of this method must be exactly similar to class Name.

② No return type.
{not even void}

Magic.

```
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.

A = { 2 5 3 6 2 }

B = { 4 5 1 6 2 }

```
ans = 0
for( i = 0 ; i < N ; i++){
    Rectangle r = new Rectangle( A[i], B[i]);
    if ( ! r.isSquare() ){
        ans += r.area();
    }
}
return ans;
```

```
for(i=0; i<N; i++)
    if [A[i] != B[i]){
        ans += A[i]*B[i]
    }
}
```

?

```
ans = 0 , Rectangle r = new Rectangle();
for( i = 0 ; i < N ; i++){
    r.l = A[i]
    r.b = B[i]
    if ( ! r.isSquare() ){
        ans += r.area();
    }
}
return ans;
```

→ Readable.

Re-usable.

Q) Add a method/function to check if area is

① greater than an integer K.

② greater than another rectangle.

this / self → reference of current calling object.

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

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

Method Overloading.

```
boolean isGreaterThan (Rectangle r){
    return this.area() > r.area();
    //return this.isGreaterThan(r.area())
}
```

```
Rectangle r1 = new Rectangle (6,4)
Rectangle r2 = new " (5,5)
print (r1.isGreaterThan(20));
print (r1.isGreaterThan(r2));
```

Method overloading

→ Same function name

→ different parameters

```
→Rectangle r5 = new Rectangle(7,3);
    r5.isGreaterThan(20);
```

Q:) Given N rectangles with length & breadth in A[i] & B[i].
∀ index i, count the no. of squares on left of i such that area of square is greater than the area of current rectangle.

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[] ans = new int[N];

Rectangle[] R = new Rectangle[N];
```

custom-data-type

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

R → [[2, 4], [5, 5], [3, 1], [6, 6], [2, 2]]
0 1 2 3 4

i = 4

count = ~~1~~ 2

```
for( int i = 0 ; i < N ; i++){
    Count = 0
    for( j = 0 ; j < i ; j++){
        if( R[j].isSquare() && R[j].isGreaterThan(R[i]) ){
            count++
        }
    }
    ans[i] = count ;
}
```

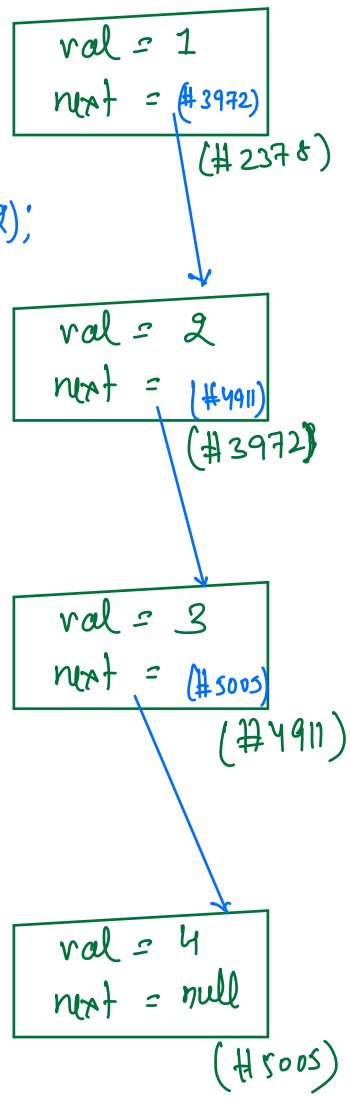
return ans.

Object Reference Inside A class.

```
class Node {
    int val;
    Node next;
        ↳ object reference

    Node (int x)
    {
        val = 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
```

Dis → Linked List.

Trailer → [whole movie is pending]

int a = 10;

int a = "Jitender";

Node a = _____

↑

address of the node
or
reference

```
class Student {
    int id;

}
```

Student s1 = new Student();

print(s1.id);

⇓

o/p ⇒ 0

int → 0

double → 0.0

non-primitive data types → null

boolean → false.

Java - Defaults.