



Today's agenda

↳ object oriented programming



AlgoPrep



## ↳ Programming Paradigms → Patterns / model / approaches

- (i) Procedural → C, C++
- (ii) OOPs → Java, Python
- (iii) functional → JS, C#
- ⋮

### ① Procedural Paradigm

↓  
old name for method

→ bunch of Procedures.

→ those Procedures internally call each other.

```
AC() {  
    B();  
}
```

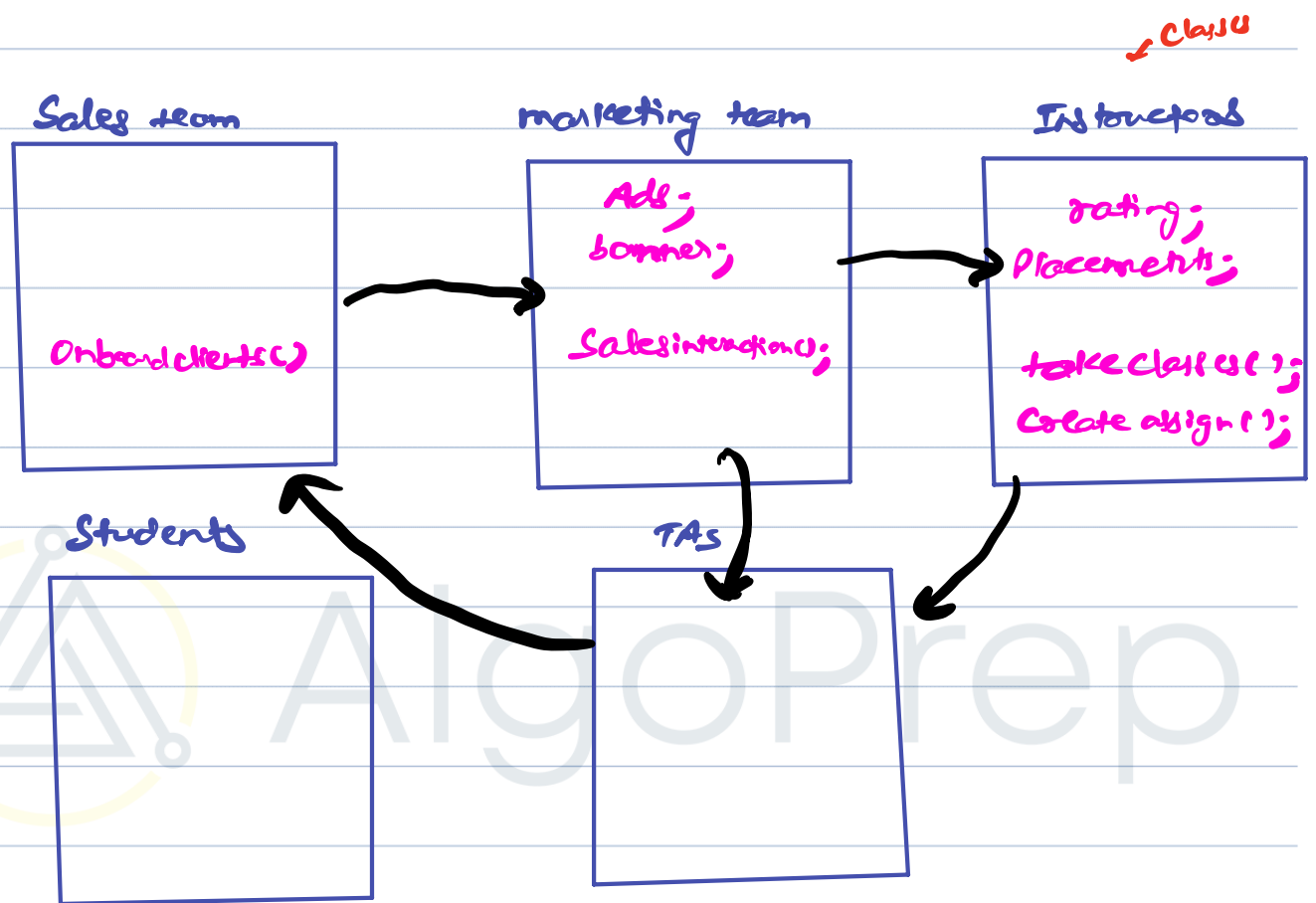
```
BC() {  
    CC();  
}
```

```
CC() {  
    f  
}
```



## ① OOP (Object oriented Programming)

### AlgoPrep



→ Principal of OOP → abstraction (values)

→ Pillars of OOPs (help implement principal)

↳ Inheritance, Polymorphism, encapsulation

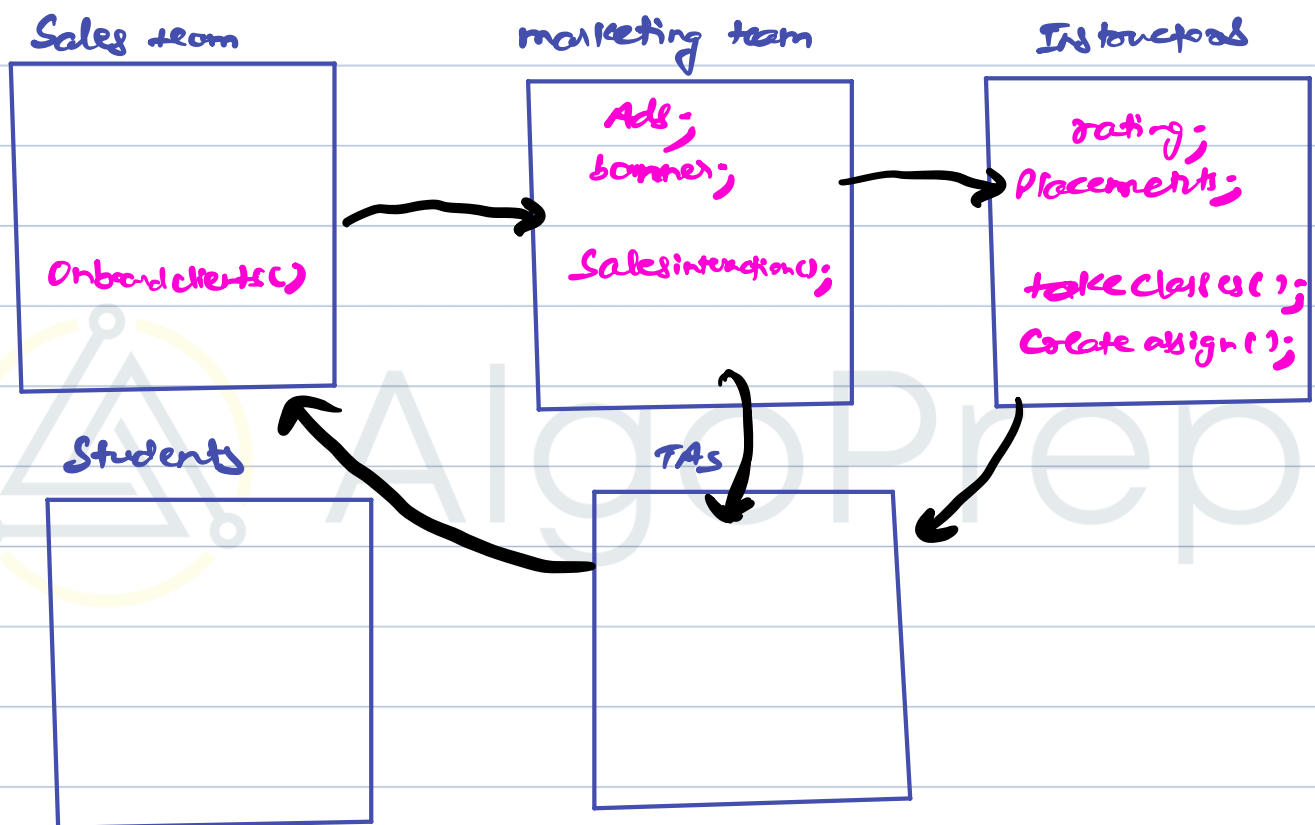


## \* Abstraction

↳ ① a complex system being represented by multiple entities

② has attributes

③ has some behaviours



↳ others don't need to know all the internal working of an entity.



## \* Encapsulation



→ why are there capsules for medicine



- i To hold multiple salts
- ii Protect medicine from outside environment.

→ encapsulation in oops

← class

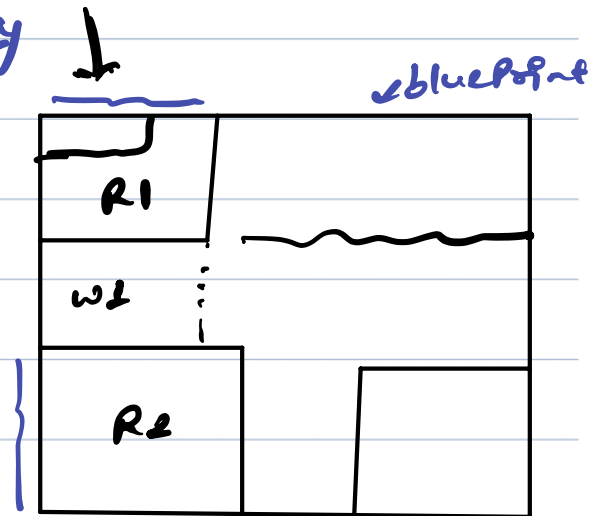
i Stores the attributes and behaviour together for one entity.

ii Protects the entity from illegal access  
↳ using access modifiers.



## \* class

↳ Blueprint of an entity <sup>→ Home</sup>



→ It is not a real house.

→ Just a representation of my house.

→ Doesn't take any space.

→ Can create multiple houses using same blueprint.

```
class house {  
    int roomcount;  
    int floors;  
    int roomsize;  
    String color;  
}
```

→ doesn't take any RAM space.

→ multiple instances of one class.

```
rentTheHouse();  
Host Parties();  
sellHouse();
```



## Object

↳ real instance of a class.

↳ occupy RAM memory.

→ house <sup>house</sup> h1 = new house();  
h1.roomcount = 5

→ house h2 = new house();  
h2.roomcount = 30;

→ house h3 = new house();  
h3.roomcount = 1;

each objects  
are completely  
independent.

each one of  
them will have  
diff set of data.



## ★ Constructors

```
public static class house{
    int roomcount;
    int floorsize;
    String color;

    //constructor
    house(int x,int y, String c){
        roomcount = x;
        floorsize = y;
        color = c;
    }
    house(){

    }
    house(int x,int y){
        roomcount = x;
        floorsize = y;
    }
}
```