Today's agenda

↳ object oriented Programming

↳ Programming Paradigms → Patterns / model / approaches

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

(i) Procedural Paradigm

↓
old name for method

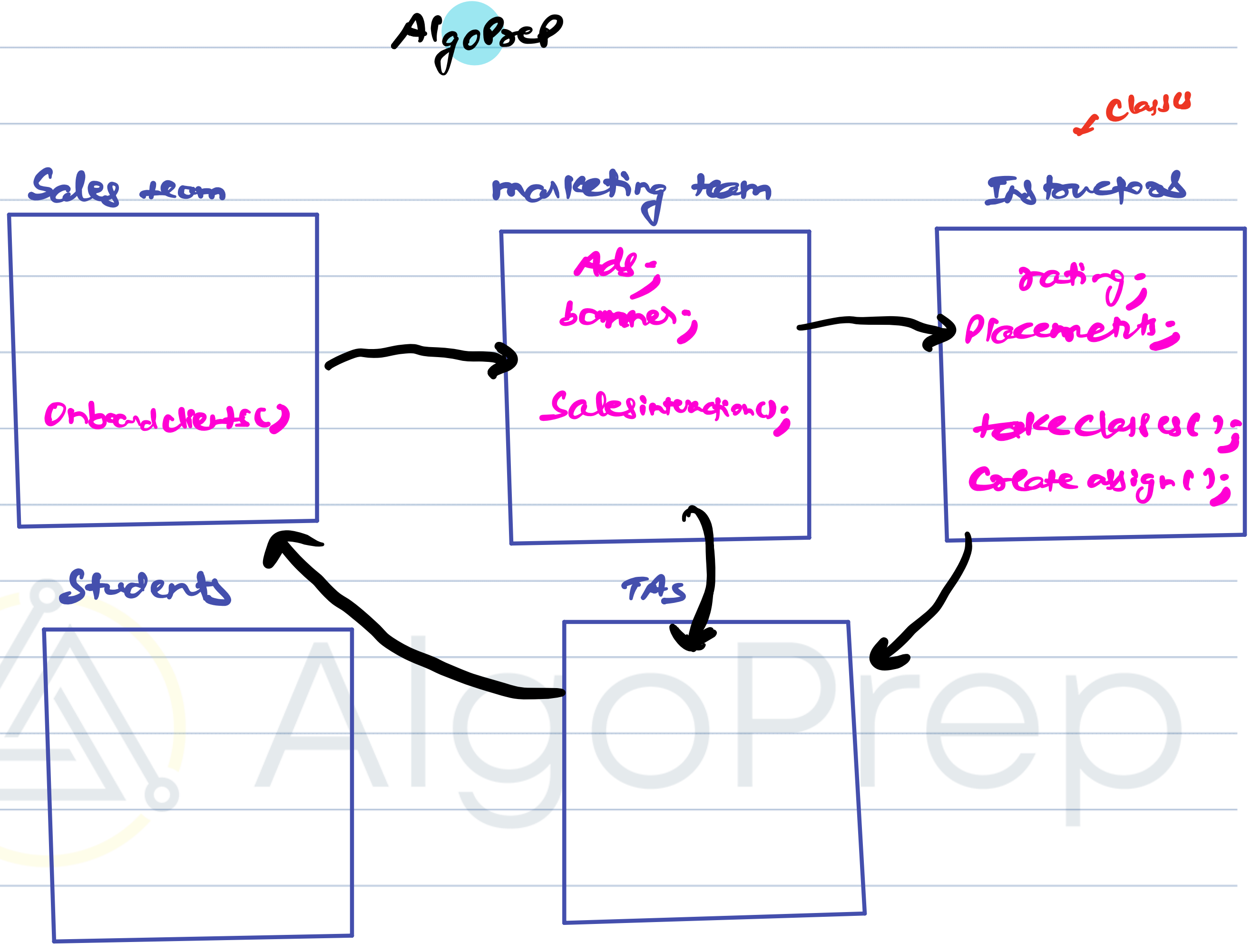
→ bunch of Procedures.
→ those Procedures internally call each other.

```
A() {
    B();
}
```

```
B() {
   C();
}
```

```
C() {
}
```

① OOP (objected oriented Programming)



→ Principal of OOP → Abstraction {values}

(help implement Principal)

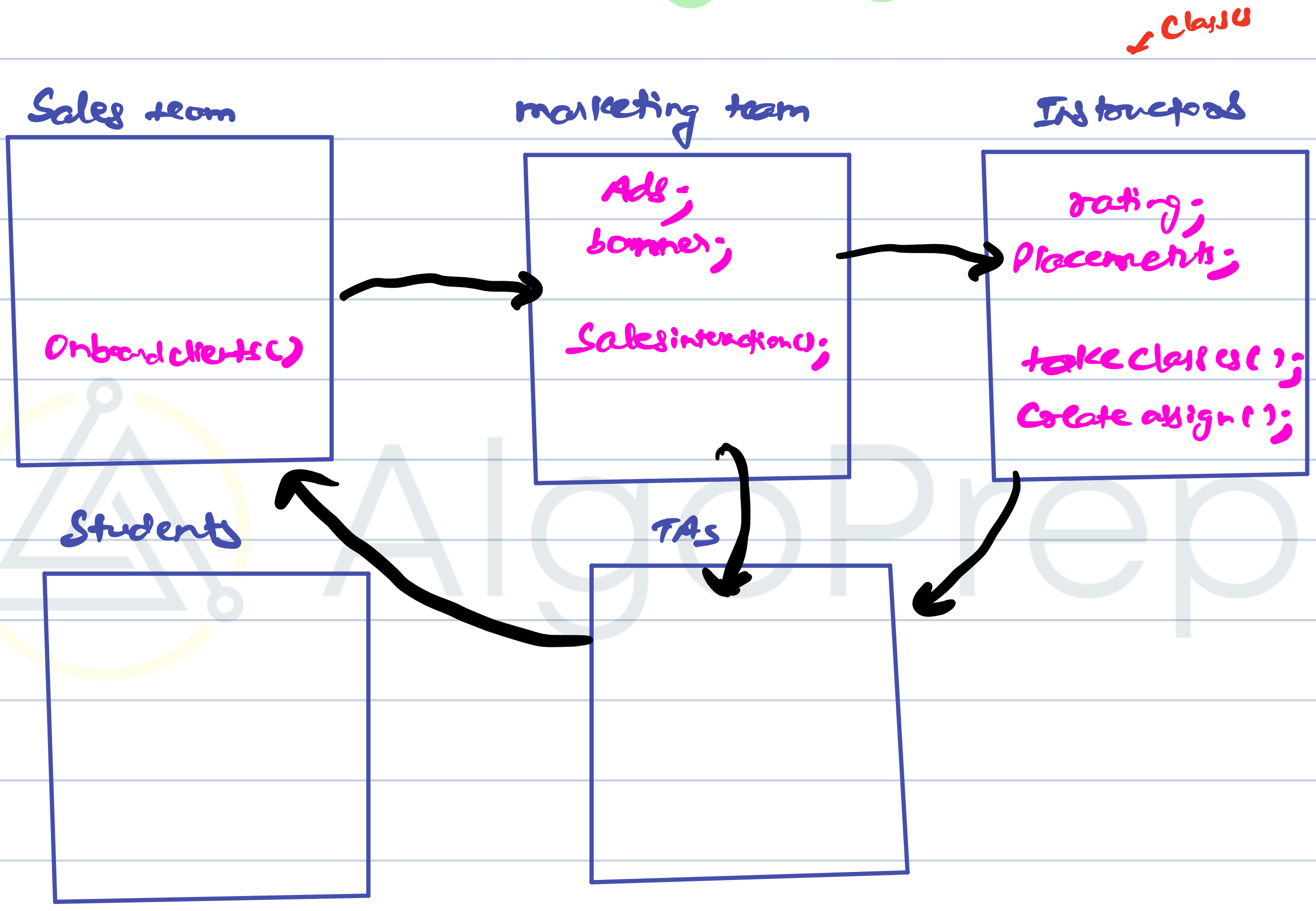
→ Pillars of OOPs

↳ Inheritance, Polymorphism, encapsulation

* Abstraction

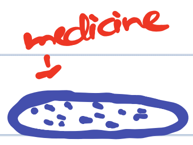
↳ ① a complex system being represented by multiple entities

ⓐ has attributes
or
ⓑ has some behaviour



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

* Encapsulation



→ why are these Capsule for medicine

↓

(i) To hold multiple salts

(ii) Protect medicine from outside environment.

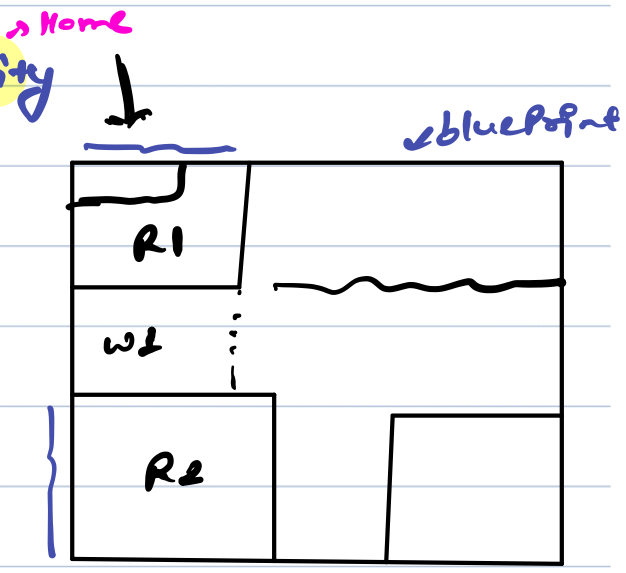
→ encapsulation in oops

↳ (i) Stores the attributes and behaviour together for one entity. ↙ class

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

* **Class**

↳ Bluepoint of an entity



→ It is not a real house.

→ Just a representation of my house.

→ Doesn't take any space.

→ Can create multiple houses using same bluepoint.

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


        rentthehouse ();
         HostParties ();
           sellhouse();
}
```

→ doesn't take any RAM space.

→ multiple instances of one class.

Object

↳ real instance of a class.

↳ occupy RAM memory.

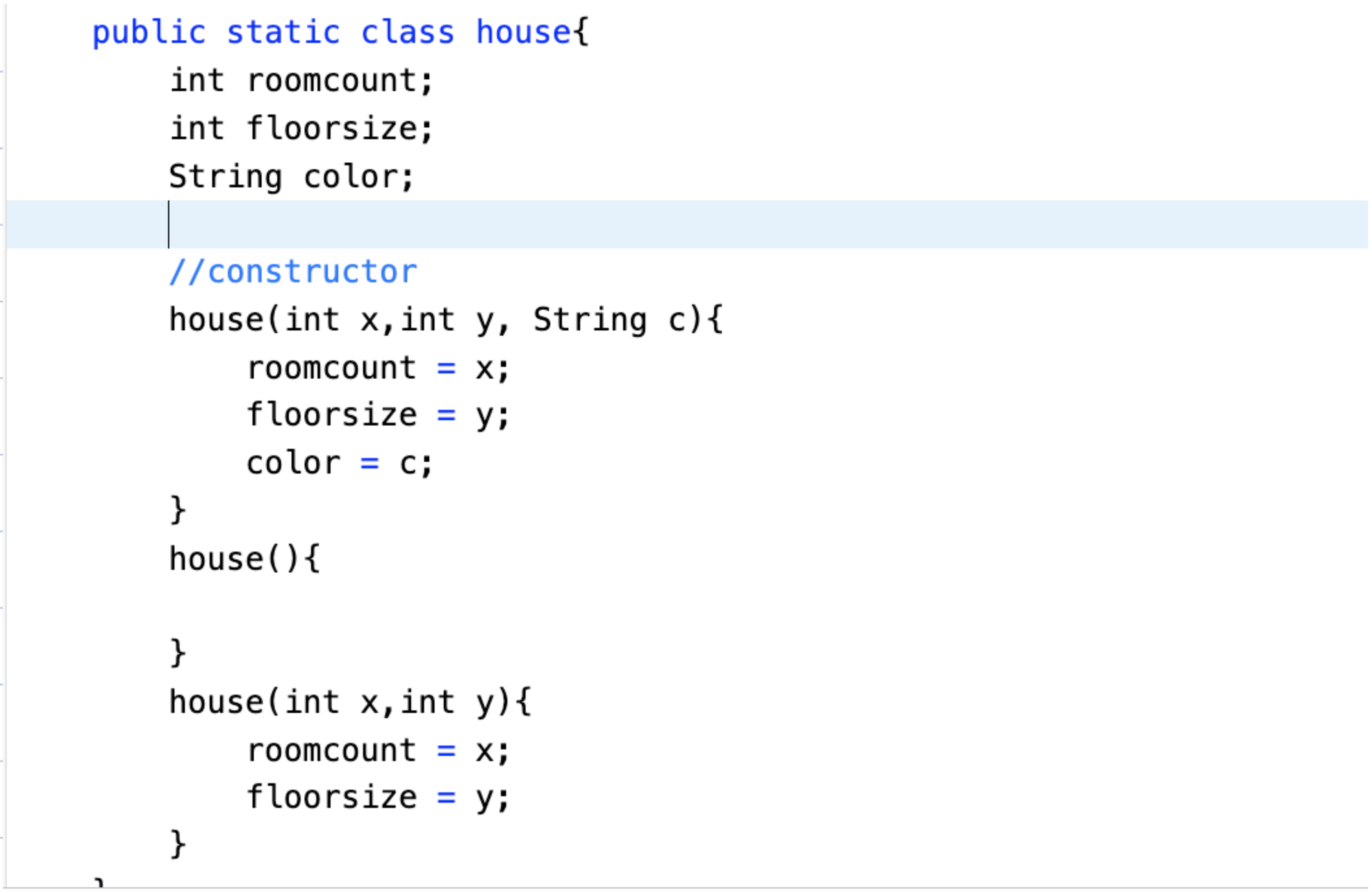
```
→ house h1 = new house();
        h1.roomcount = 5

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

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

name

each objects are completely independent.

each one of them will same diff set of data.

★ Constructor



```
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;
    }
}
```