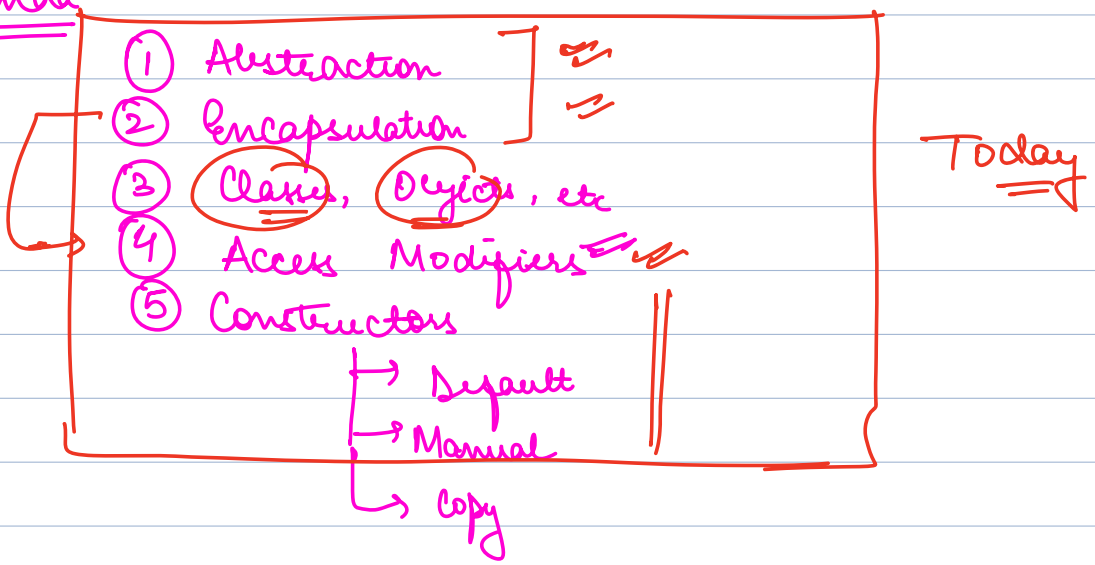
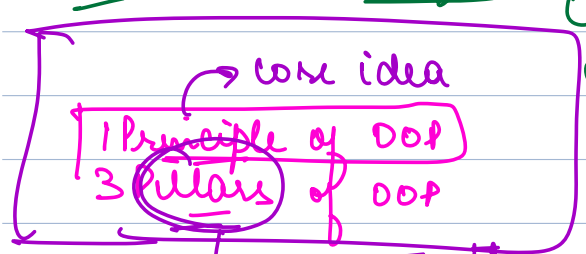
OOP2 ⇒

Agenda



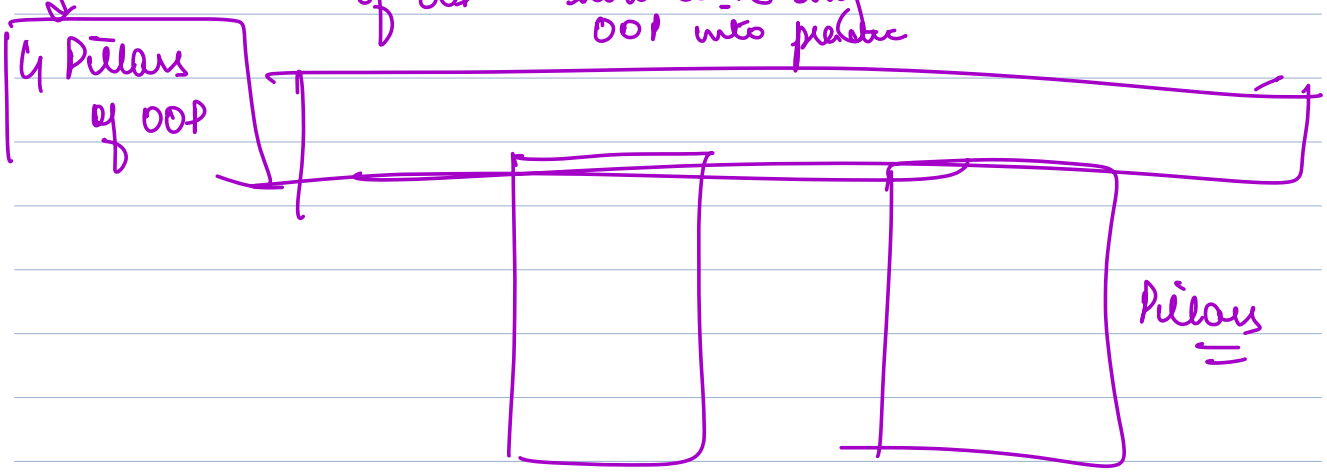
OOP Pillars of OOP

↳ core concepts of OOP
↳ core features / ideas of OOP



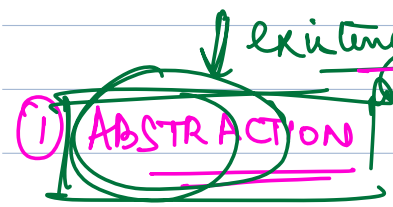
↳ support the idea of OOP ⇒ how to bring OOP into practice

6/4/5/3



1 Principle of OOP ⇒ Abstraction ⇐

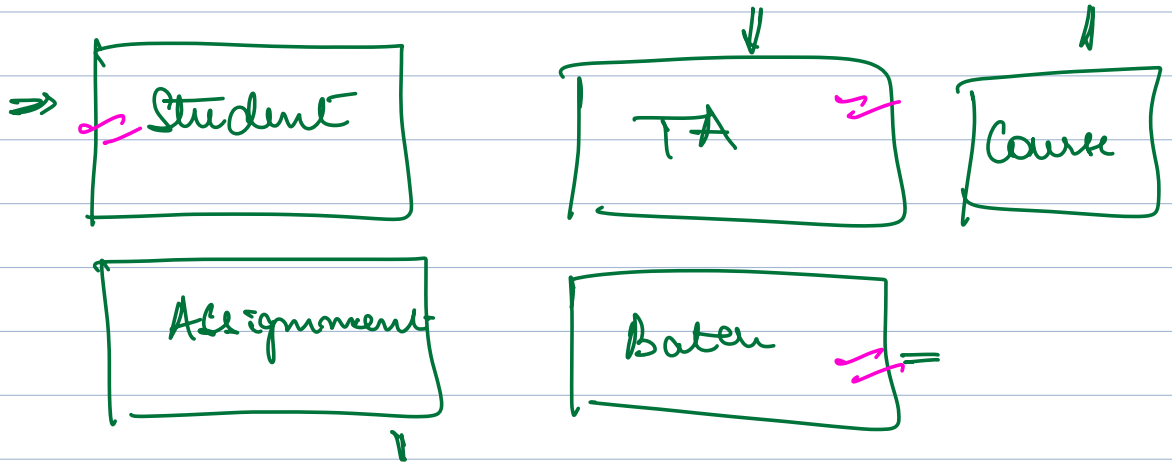
3 Pillars of OOP ⇒
- Inheritance
- Polymorphism
- Encapsulation

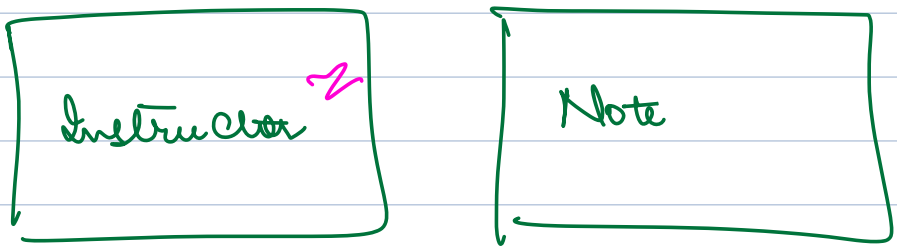


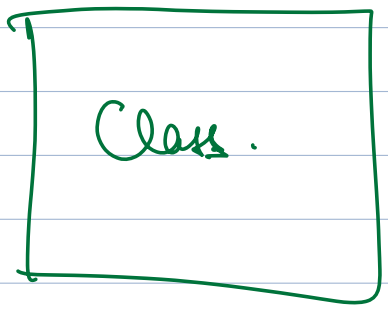
represent a software system in terms of ideas.

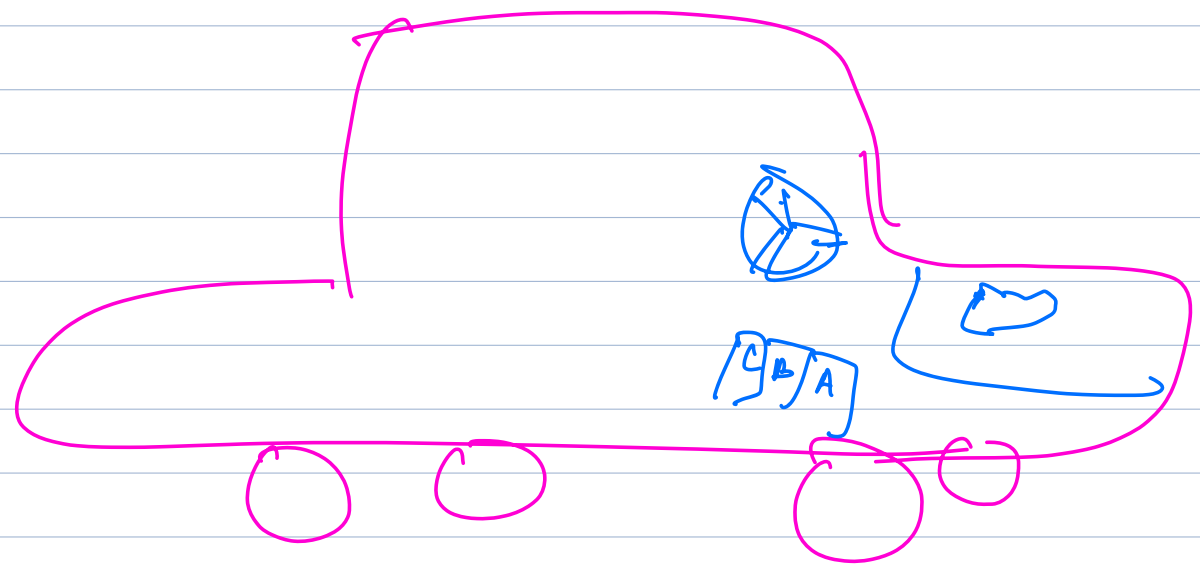
⇒ Whatever S/W system you are implementing there will be many ideas in that (entities / anything about which you have to store info / any thing that can cause) represent that S/W in terms of ideas

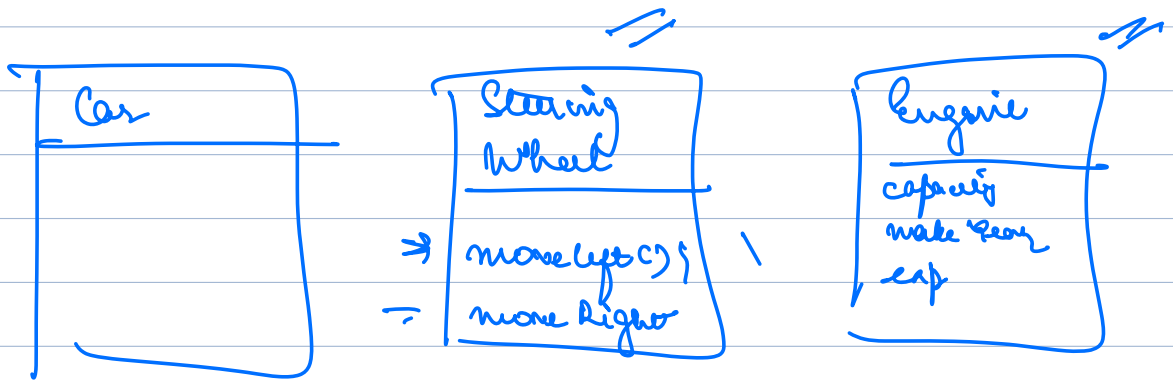
Scaler's S/W system

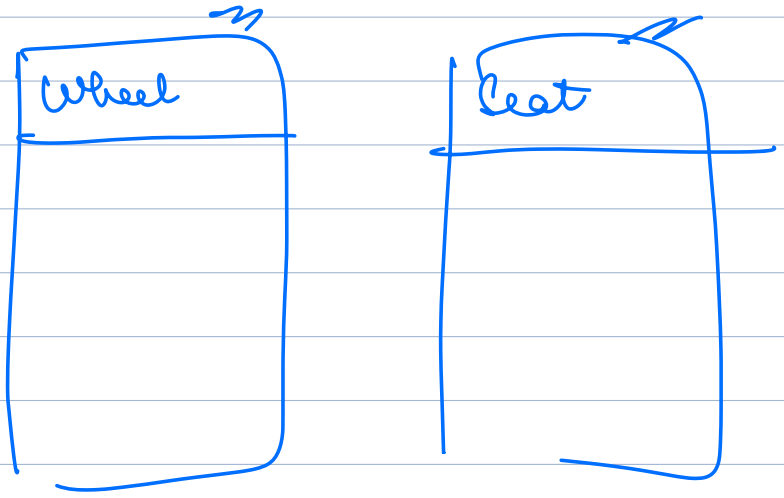



Instructor
Note

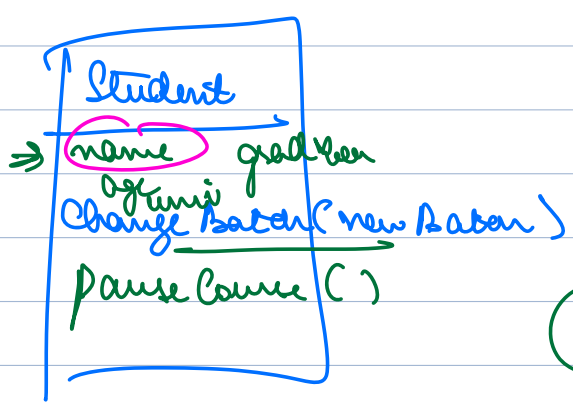

Class.

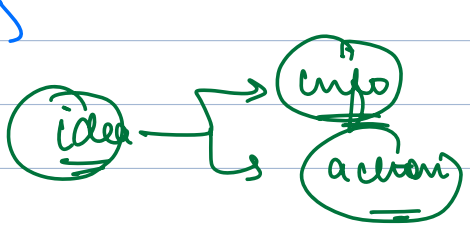

C
B
A


Car
Steering Wheel
moveLeft()
moveRight()
Engine
capacity
make Noise
cap


Wheel
Seat


Driver
wheel. moveLeft()


Student
name
grade
age
changeGrade(newGrade)
pauseCourse()

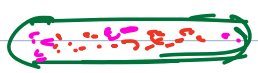

idea
info
actions

Abstraction

① Rep a complex s/w system in terms of ideas:
   a) Anything about which you are storing info
   b) Anything on which you can perform actions.
② Not everyone needs to know details of working of these ideas.

3 Pillars ⇒ Help us represent complex s/w system in terms of ideas.

① Encapsulation



⇒ Holds the medicine together
⇒ Protects the medicine from external environment

Purpose of Encapsulation

① Hold info and actions of an idea together
② Protect the info and actions against illegitimate access.

Student {

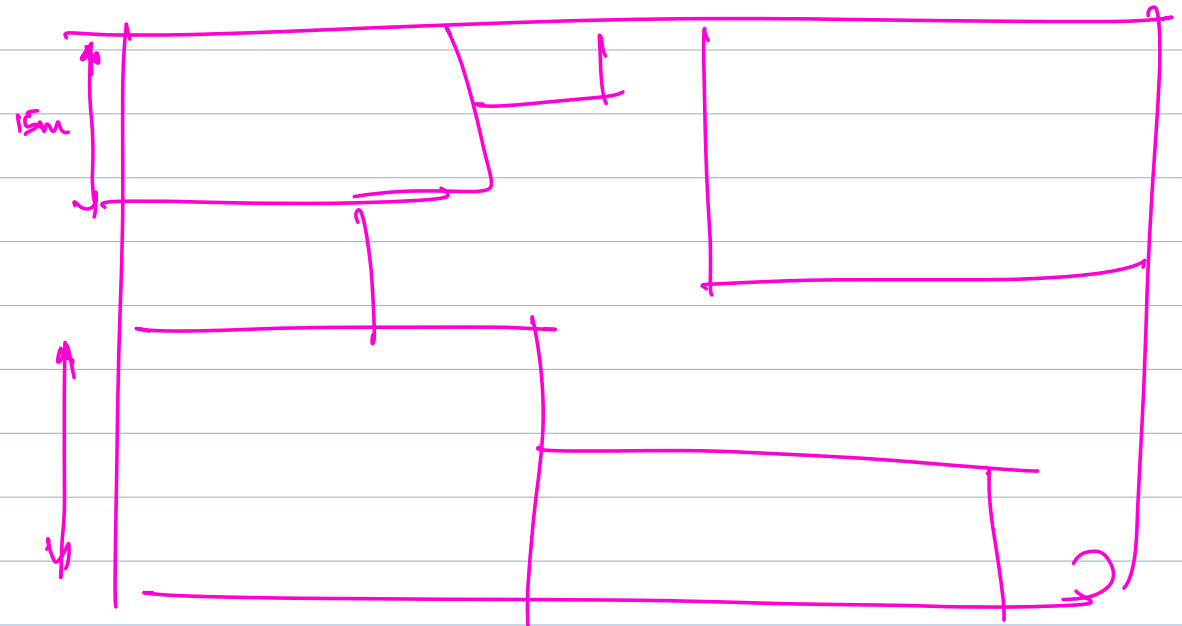
}

Class {

Batch {

}

Class ⇒ Blueprint of an idea



⇒ design of how every house should be it will look like / what all will such a house have.

Blueprint

① 1 Blueprint ⇒ Multiple Houses

② Doesn't occupy any land

class ⇒ idea. (info and action)

name of idea → what every instance of that class should have

```
class Student {
    String name;
    String email;
    String batch;
    double psp;

    void changeBatch(String newBatch) {
        batch = newBatch;
    }

    void applyForJob(int jobID) {
        if (psp >= 80) {
            print("success")
        } else {
            print("failure");
        }
    }
}
```

⇒ From one class you can create multiple instances.

⇒ Class doesn't take RAM space. Instance of class does.

OBJECT

- → Instance of a class.
- → Occupy memory
- → each object is totally isolated from other objects of that class.