Currently :- Senior SWE at Booking. com in Ams.

Previously :- Razorpay (SDE 1)

Samsung Research.

Graduated from Thapar University 2019.

Agenda

→ Intro to OOP

→ Procedural Programming

→ OOPs

Introduction to OOP

Programming Paradigms.

* Procedural Programming → C →
* Functional → Scala, Haskell, Go,
* Object Oriented → Java, Python
* Reactive → rxjs

Procedural Programming.

Procedure $\longrightarrow$ A $\underbrace{\text{set of instructions.}}_{\text{Function}}$

* This is just an oldage name for the functions/methods that we use these days.
* Each procedure may or may not internally call other procedures.

→ Execution of a program starts from main() function.

```
main () {
        a();
        c() -
        b()
}
```

```
a() {
    —— ——
    —— ——
    —— ——
}
```

```
c() {
    ————
    ———
    ———
}
```

What is the problem with Procedural Programming hang.?

→ Sofia is listening to Alok
→ Alok is teaching.
→ Rahul is attending class.
→ Rohit is making notes.
→ Surya is wondering when will the class end.

Subject + verb
someone doing something
entity performing action

Q Write a method called printStudent() with arguments:-

```
printStudent (String name, int age, String gender){
    System.out.println ( name );
    -     -    -    -    ( age );
    -     -    -    -    ( gender );
}
```

struct (structure)

```
struct Student {
    string name;
    int age;
    string gender;
}
```

A struct looks very similar to a class. But there are a few differences.

* A struct has no methods.
* All variables/attributes are public.

something — someone — sofia.

```
printStudent (Student st)
{
    System.out.println ( st.name );
    -    -    -    -    ( st.age );
    -    -    -    -    ( st.gender );
}
```

Someone doing something
↓ — ↓ printStudent ()

Something is happening on someone instead of someone doing something. → **Not natural**

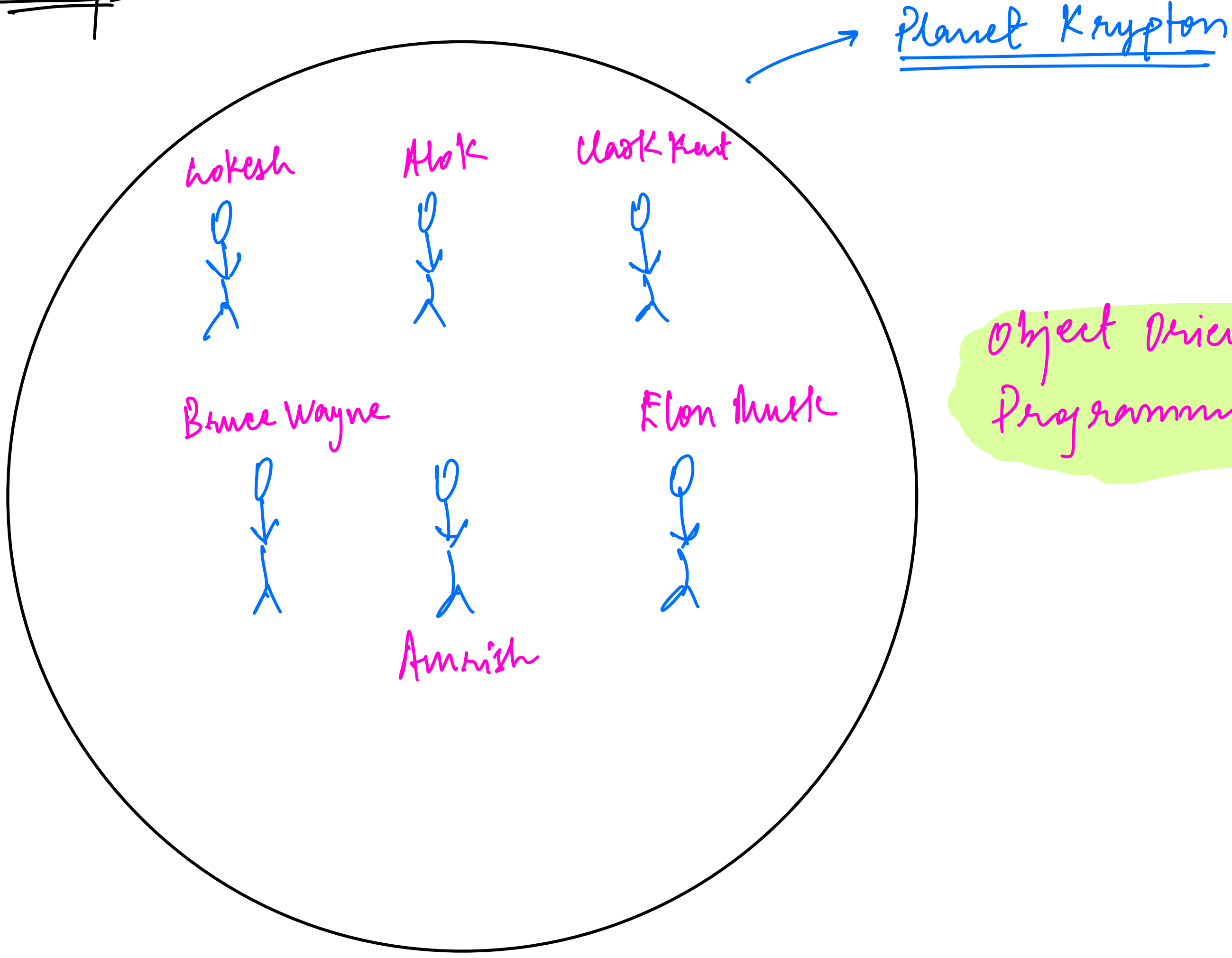
printStudent (Student) v/s student.print ();

Problem:-

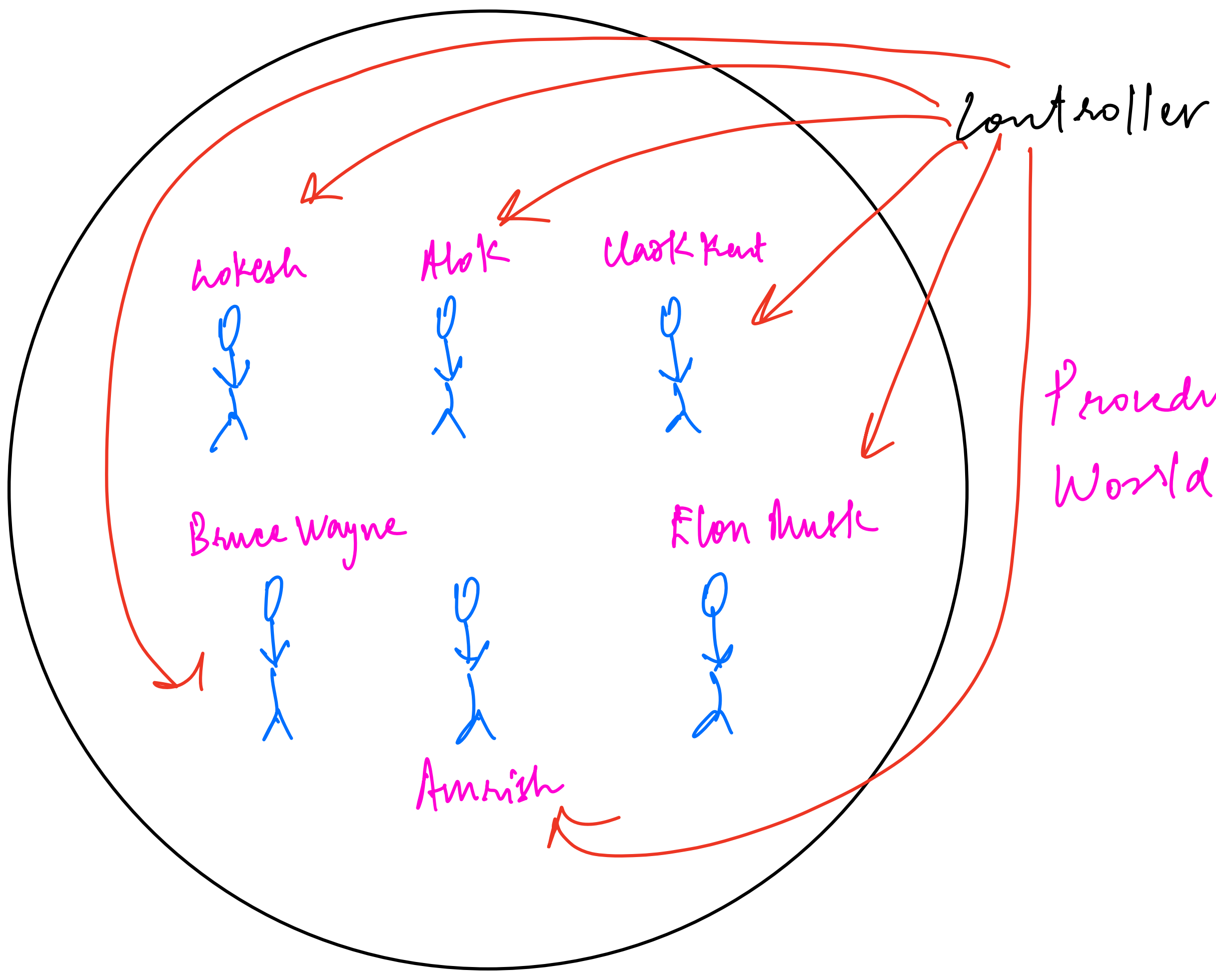
(i) Action is performed on entities.

(ii) Variables are visible / accessible to everyone.

Example



Object Oriented Programming World.



Procedural Programming World.

OOP :- Software systems should consist of entities

```
class Student {
    private String name;
    private int age;
    private String gender;
    void print () {
        sout (name);
        sout (age);
        sout (gender);
    }
}
```

I am not accessible outside this class.

Java

No one uses this. ← {PHP, Go, Perl, Java

Python.

Cons of PPL :-

* Difficult to make sense of a complex problem.
* Difficult to debug.

OOP Introduction

* Entities are the core in OOP
* Every entity has some attribute and behaviour.

4 pillars of OOP
- → Abstraction
- → Inheritance
- → Polymorphism
- → Encapsulation.

Book :-
Java :- A complete Reference.
(optional)

3 pillars and 1 principle of OOP.

Pillar ? → Support system. (Truthful / Help / . .)

Principle ? → Foundation / Fundamentals / (I will be a good person)

We have the pillars to support the principle.

Principle of OOP → Abstraction

Pillars of OOP → Inheritance
Polymorphism
Encapsulation.

Abstraction

abstract

hiding, missing, a general idea,

quality of dealing with ideas.

Represent something in terms of ideas → Real Entity → concept

present their screen, teach, unmute a student.

Founder → Anshuman

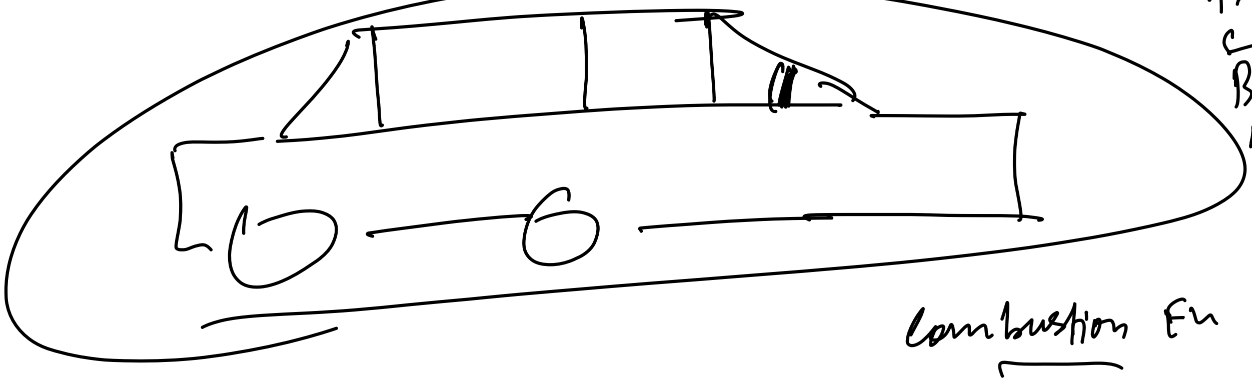
Remote Ins → Alok, — —

Students → Lokesh, Sofia, Rahul, Surya.

send messages,
pause course,
attempt assignments,
change batch.

→ Purpose of Abstraction

↳ others don't need to know the details of the idea.



Car {
fuel
4 ~
C
B
~

Combustion En {

}

Admin Portal

schedule

change class title.

change time.

$\longrightarrow$

class Student {

$\longrightarrow$

batchName

joining date

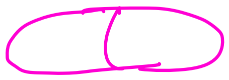
Atten psp

}

Encapsulation

capsule

↳ why we put medicines in capsule



What happens when capsule breaks?

→ Holds the medicine together

→ Protects it from outside environment.

→ Avoids mixing of some powders with other inside the capsule.

Same purpose of encapsulation in OOP.

I am storing Attributes and Behaviour.

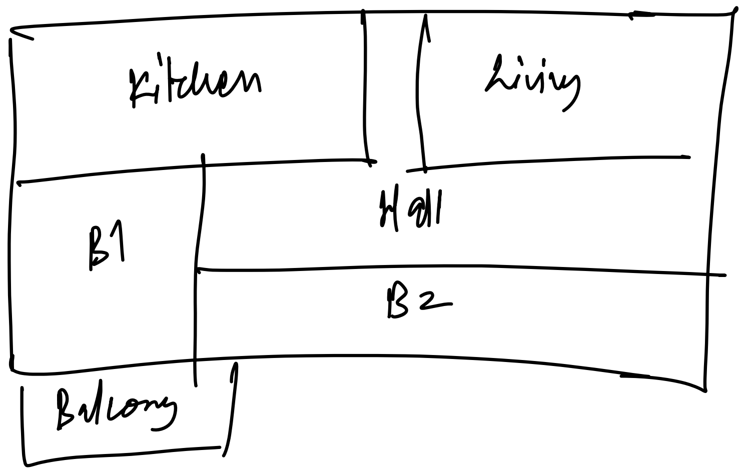
We store attributes and behaviour in a class.

```
class Student {
    private age;
        =
        -
}
```

Terminology

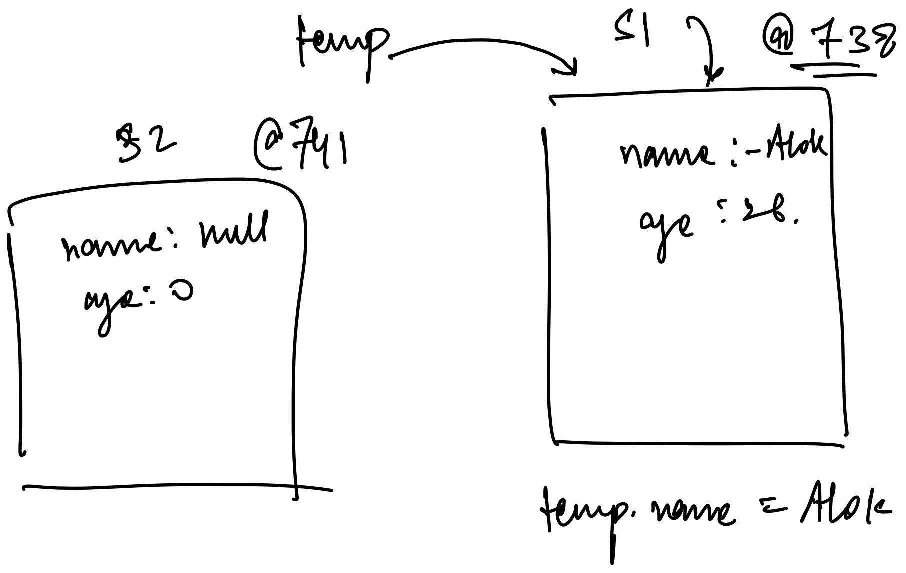
① **Class** :- Blueprint of an idea.

```
class Student {
    int age;
    String name;
    String batch;

    changeBatch (){
    }

    pauseCourse (){
    }

    giveMockInterview (){
    }
}
```



→ class takes no space in memory
→ NOT a real entity
→ Multiple instances of the same class.

2. **Object** :- Real instances of the class.


temp
S1
@738
name :- Alok
age : 28.
S2
@741
name: null
age: 0
temp. name = Alok