Class starts in 5 minutes

Agenda

2:30 hrs

↳ Intro to OOPs

Intro to OOPs

What paradigm of P.L.:

↳ Procedural → C
↳ Object Oriented → Java, Python, C++
↳ Functional → Haskell, Java, C++
↳ Reactive

Procedural Programming

↳ A set of instructions => function.
↳ Procedure => old age name for functions

↳ We organize our code into a bunch of procedures.

↳ A special function from where flow starts => main.

```
main(){      a(){      b(){
  a()          b()       c()
}            }           d()
                       }
```

Problem with Procedural P.L.:

↳ Students are learning
↳ Instructor teaching
↳ Teacher is listening to lecture

subject + verb

Real world ⇒ entities perform some actions.

print Student (String name, int age, String gender)

struct Student { ⇒ Similar to a class but no methods & all variables are public
    String name;
    int age;
    String gender;
}

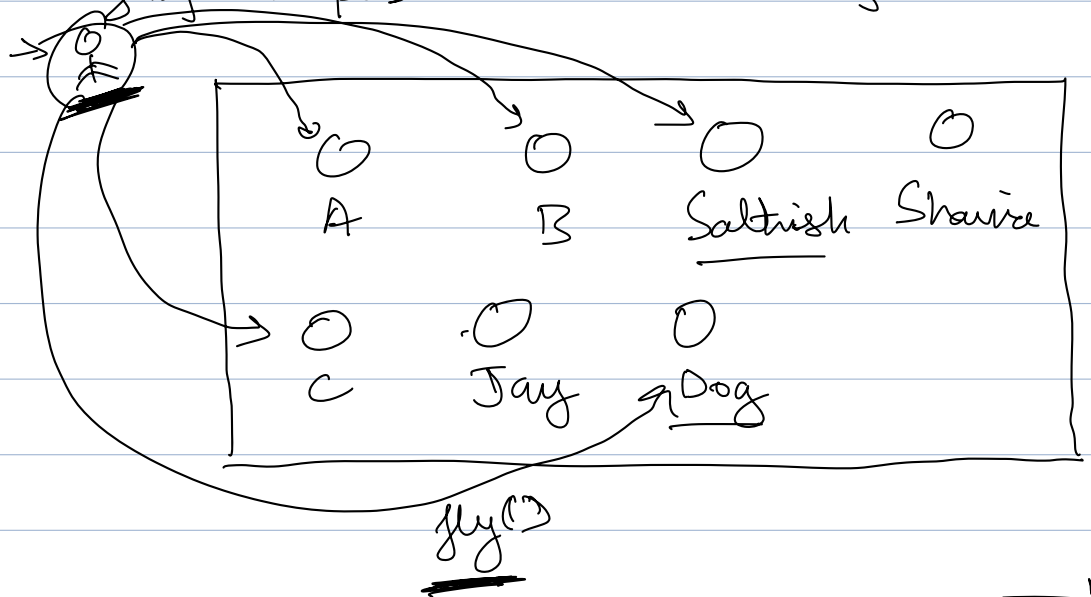
action

printStudent (Student st) {
    print (st.name)
    " (st.age)
    " (st.gender)
}

Real world ⇒ entities perform some action
Procedural world ⇒ Action is being performed on entities

① printStudent(st) vs ② st.printStudent()

Everything is public about an entity ⇒



~~fly Dog~~ bark(Dog dog){

}

dogFly(){

s(Dog :-

}

OOPs

↳ S/W system should consist of entities and each entity contends its attributes and has some behaviour.

↳ Student {

Stry name

int age

:-:

print() {

sout(—)

sout(—)

}

}

Student.print();

vs

printStudent(student);

OOPs

↳ entities are the core of OOPs.

↳ attributes & behaviour

"pillars of OOPs"

3 pillars and 1 principle ←

↳ fundamental / guideline

↳ something on which our concept is based on.

To provide support

Principle of OOPs ⇒ Abstraction.

↳ Build your s/w system around abstraction and use the pillars (Inheri, enca, poly) to implement abstraction

Principle: I will be a good person]

Pillars: honest, hard work, . . .]

Abstraction

↳ Representing something in terms of ideas.
↓
attr & behaviour.

Student & Instructor Mentors Class Batch

Benefit

↳ Others don't need to know the details of the idea

↳ Rep a complex s/w system in terms of ideas.
↳ Others don't need to know about the details of the idea.

Break: 8:11 → 8:17 6 minutes

Encapsulation

↳ 1st pillar of OOPs. that helps us implement abstraction.

Encapsulate

Capsule

↳ To hold diff medicines together. ←
↳ To protect medicine from outside env.

✓ (1) Store attr and beh. together of an idea.
↳ class

✓ (2) Protect the attr & behav from illegitimate access from other classes.

Student
private int age

class => Blueprint of an idea.

class Student {
int age;] -> attr
String name;]
≡
pauseCourse();
giveMockInterview();] -> Beha.
}

Object => A real instance of a class.

Student is a class
Jay is an object of Student
Fabio " " " " "
Tarun " " " " "