Programming Paradigms

↳ Procedural programming
↳ Object oriented programming

Access Modifiers.

Programming Paradigms

Style or standard way to write a program.

W/o programming paradigm.

1) hers structured
2) Hard to read & understand
3) Hard to test
4) Difficult to maintain

Types of Programming Paradigms.

1) Imperative Programming.

Eg:

```
int a = 10;
int b = 20;
int sum = a + b;
print (sum);

int diff = a - b;
print (diff);
```

---

2) Procedural Programming

Splits the entire program into procedures or functions (section of code that performs a specific task) which are reusable.

```
int a = 10;
int b = 20;
addTwoNumbers (a, b);
substractTwoNumbers (a, b);

void addTwoNumbers (a, b) {
    print (a+b);
}
void substractTwoNumbers (a, b) {
    print (a-b);
}
```

```
}
```

---

3) Object Oriented Programming

4) Declarative Programming.

SQL ⇒ Select * from Customers;

Procedural Programming

```
void addTwoNumbers(a, b) {
    int sum = a+b;
    print(sum);
}

void addThreeNumbers(a, b, c) {
    int sum = a+b+c;
    print(sum);
}
```

```
void main() {
    addThreeNumbers(10, 20, 30);
}
```

Problems with PP.

1) We are studying
2) Ayush is teaching
3) We are being taught progr. para.

Subject ↑ (someone is doing something) ↑ Verb.

```
printStudent(String name, int age, Stig gender){
    print(name);
    print(age);
    print(gender);
}
```

Structure

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

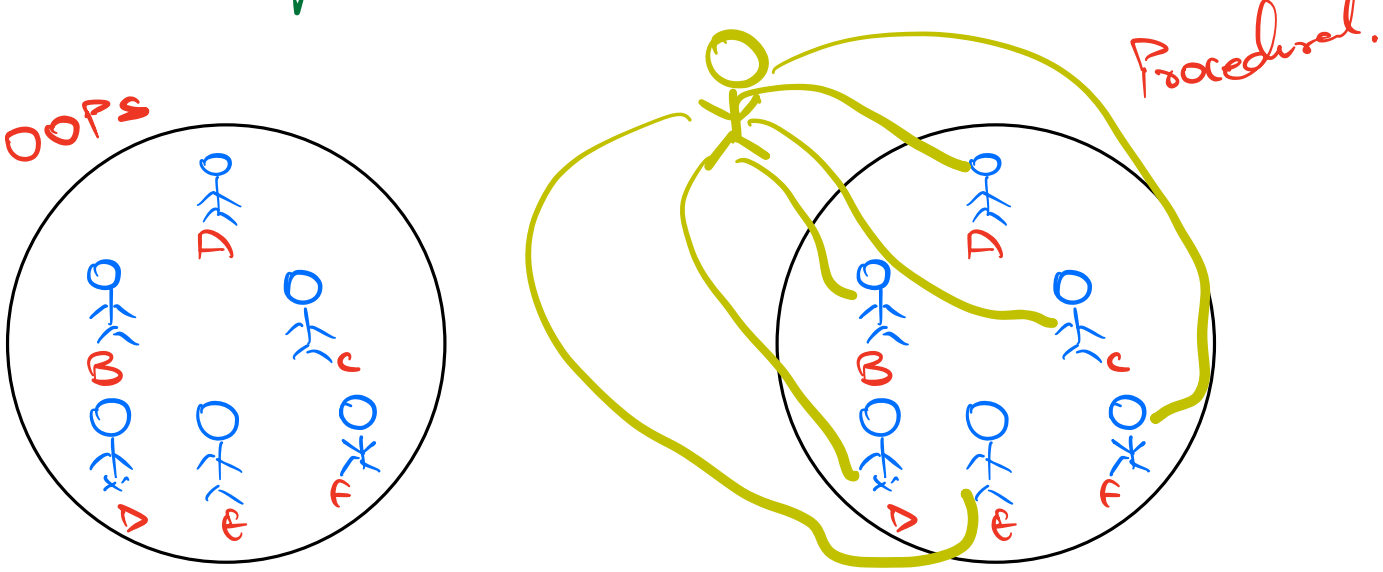
Something — printStudent

Someone — (Student st)

```
{
    print (st.name);
    print (st.age);
    print (st.gender);
}
```

Someone ⇒ Student

Something ⇒ printStudent



OOP :) Software Systems should consists of entities

2) Each entity controls its attribute & also a defined behaviour.

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

        void printStudent() {
            print(age);
            print(name);
        }
}
```

OOPS

⇒ Entities are core in OOPs

⇒ Every entity has some attributes & behaviours.

Classes & Objects (entities)

Class: Blueprint.

↳ floor plan of the appartment.

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

    changeBatch();

    pauseCourse();

    giveMockInterview();
}
```

Object: Real instance of class.

Pillars of Object Oriented Prog.

3 pillars & 1 principle.

- pillars → Support to hold things together.
- principle → fundamental foundation/ concept

Principle ⇒ I will be a good person.

Pillars ⇒ I will be truthful
I will do hardwork (100% PSP)
I will respect everyone

Principle ⇒ Abstraction ✓

Pillars. ⇒ Inheritance
Polymorphism } OOPS 2
Encapsulation. ✓

Abstraction

Representing in terms of ideas.

Abstraction is a way to represent complex software design in terms of idees.

Q What needed to be represented in terms of idees

1) Data
2) Anything that has behaviours.

Encapsulation

Capsule.

Q If capsule seals:

1) It flows aways. : Hold the medicine together

2) Multiple powders present : Prevent mixing w/t each other.

3) Protect medicine from outside world.

Encapsulation in OOPS

1) Store together
   - ↳ Attribute
   - ↳ Behaviours.

2) Where do we store them together

   Class.

3) Protection from outside world.

   Other classes.

   ⇓

   Access Modifiers.

Access Modifiers.

Public
Protected : Inheritence
Private

Default (when you don't use any access modifier)

"this" Keyword.

⇩

refers to the current object

Don't

Correctness > constants