Lecture :- OOPS-1 [help in DS]

- Programming paradigm
- Procedural programming
- OOPS
- Access Modifiers

Programming paradigms

Qu. What is programming paradigms ?

Ans Style or standard way of writing a programme.

Example Making a burger

Gouthama	Akhilesh	Kavish
Deep fries bun	Medium fries bun	Low fry bun
" " potato	" " potato	" " potato

Mcd. Standard way of creating burger

Without programming paradigm, the code will be :-

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

Types of programming paradigm

Imperative programming

It tells the computer how to do the task by giving a set of instructions in a particular order, i.e. line by line.

Example

```
main() {
    int a = 20;
    int b = 10;
    int sum = a+b;
    print(sum);
    int diff = a-b;
    print(diff);
    int c = 100;
    int d = 40;
    int sum2 = c+d;
}
```

redundant

redundant

Procedural programming

It splits the entire program into small procedures / functions which are reusable code blocks.

Example

```
int add ( a , b ) {
    return a+b;
}
int diff ( a , b ) {
    return a-b;
}
main() {
    int a = 20;
    int b = 10;
    print( sum (a, b) );
    print ( diff (a, b) );

    int c = 100;
    int d = 40;
    print ( sum ( c , d ) );
}
```

Object oriented programming

It built the entire program using classes and objects.

Declarative programming

In this paradigm, you specify "what" you want the program to do without specifying "how" it should be done.

Example `select * from customers`

What: fetch all rows of customers

How: Not known.

2) `int sum = MathApacheUtils.xyz(a,b);`

Procedural programming

Qu: Add three numbers.

```
void main() {
    int a=1;
    int b=2;
    int c=3;
    int sum = addThreeNO(a,b,c);
    print(sum);
}

int addThreeNo(a,b,c);
    int sum = addTwoNO(a,b);
    int total = addaTwoNO(sum,c);
    return total;
}

int addTwoNO (a,b){
    return a+b;
}
```

Problems of procedural programming.

Sentence :- subject + verb. [entity performing action]

```
printstudent(name, age, gender) {
    print(name);
    print(age);
    print(gender);
}
```

Q Is there any way in procedural programming to combine these set of attributes?

Ans structure.

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

→ similar to class

- in java, a struct has no method / functions
  c++, " " can have " "
- All fields are visible to everyone

```
struct student {
    string name;
    int age;
    string gender;
    string password;
}
```

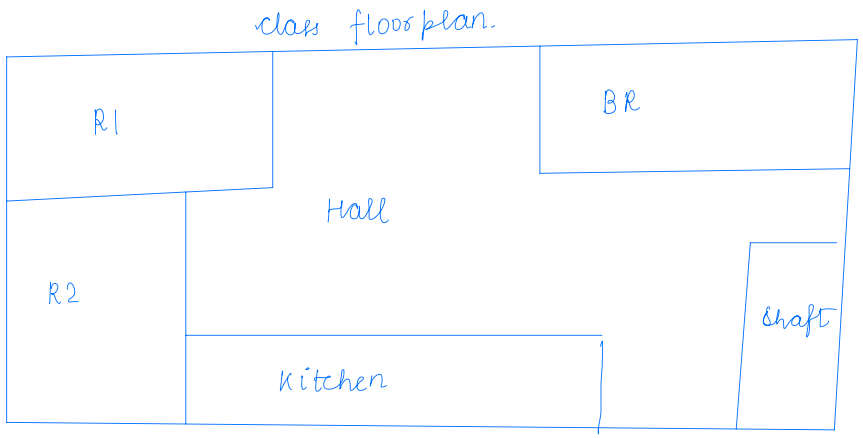
Object oriented programming

1. Entities are core of OOPS.
   Classes
2. Every entity has some attribute and behaviour.
   fields functions.

In object oriented programming, we build the entire programme using classes and objects [entity]

Class Blueprint of an idea.

Example floor plan of an apartment.



Pillars of OOPs

Acc to me. 3 pillars of oops
1 principle " ".

Principle :- concept / fundamental foundation
pillar :- support to hold things together

Principle of OOPs
- Abstraction

pillars of OOP
- Inheritance
- Encapsulation
- Polymorphism

Java : The complete reference

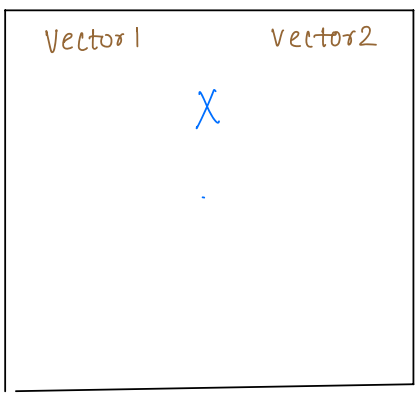
Example:
Principle :- I will be a good person.

How to be a good person?
- I will be truthful
- I will be hardworking
- I respect everyone

Abstraction

- hiding
  - privacy of data variables etc.
- representing in term of ideas

Ex: Mathematical application




```
class MathUtils {
    Vetor crossProduct( VectorV1,
                        VectorV2),
    "   dotProduct( VectorV1,
                    VectorV2)
```

Break: 8:31 - 8:40

Encapsulation

Q Does the word encapsulation contains the word that you use in daily life?

Capsule

Purpose of making capsule and not normal medicines?

If capsule breaks away, what will happen?

→ It will flow away. Prat purpose:- Holding the medicine together.

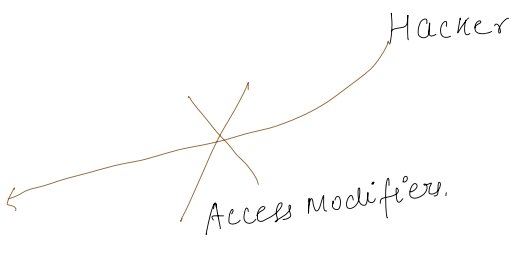
→ Protects medicine from outside environment

Q What we really store in programming?

Data:- Attributes + Behaviours.

Encapsulation allows us to store attributes and behaviours which we can bind together [purpose]

```
class
~~struct~~ student {
    string name;
    int age;
    string gender;
    string password;
}
```



Access Modifiers

- public { public string name }
- private { private string password }
- protected { protected int age }
- default { int psp }

	class	package	subclass (same pkg)	subclass (diff pkg)	Diff project
public	✓	✓	✓	✓	✓
protected	✓	✓	✓	✓	X
default / no modifier	✓	✓	—	—	X
private	✓	X	X	X	X

Static keyword

- Any member belonging to class

Scope of variable (h/w)

- Class / static scope
- Instance "
- Method / local "
- Block "

Thankyou