lecture 6

Object-Oriented Programming :-

OOP is a programming $\underbrace{\text{paradigm}}$,

$\downarrow$ model or just a way of looking at something

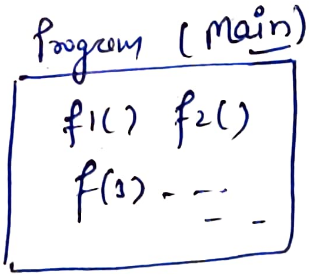
Paradigm serves as a model or framework that shapes how problems are understood & solved.

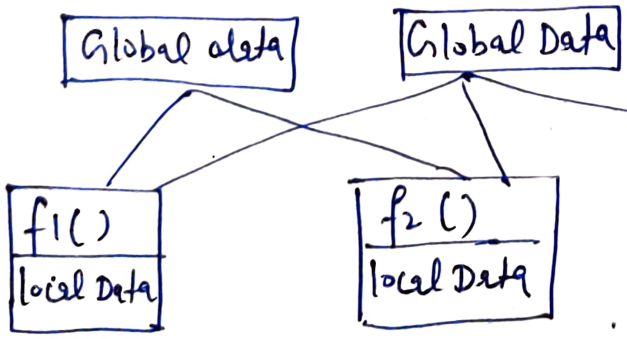
$\rightarrow$ In Computer Science, programming paradigm refers to the style or approach to programming.

$$\begin{tikzcd}
 & & \text{Programming paradigm} \arrow[dll] \arrow[dl] \arrow[d] \arrow[dr] & \\
\text{OOP (object-oriented)} & \text{POP (Procedure-oriented)} & \text{Functional} & \text{Declarative}
\end{tikzcd}$$

POP :- how to do the task.
- main focus is on procedure / algorithm / functions

In Each Program we can have global data shared by every function





So these are functions are sharing global data as well as having their own local data. It means Data is not secure here. $\boxed{\text{Data Security is an issue in POP.}}$

e.g! - locker in your house.

If everybody can access this then money is not secure

→ Second thing in POP because of interdependency of functions it is not well suited for large scale applications/projects. So it is not well suited for the projects/applications who regularly needs some updates or change.

let's take an example:-

There are 2 employees who are given a task as follows:-

Task:- There will be shapes on a GUI, a square, a circle & a triangle. when the user clicks on a shape, the shape will rotate clockwise 360° and play an AIF sound file specific to that shape.

employee 1 (Nancy)	employee 2 (Rahul)
Nancy thought what are the things the program has to do. → rotate → PlaySound } procedures she needs to implement [mainly she thought how to do]	Rahul thought what are the things in the program:- shapes & other things as well.

Square	Circle	triangle
rotate()	rotate()	rotate()
Playsound()	PlaySound()	playsound()

Something added Now one more shape added (Amoeba) which will rotate like other shapes but play an .mp3 sound file

Nancy updated her playsound() procedure	Rahul added one more class named Amoeba

```
playSound(shapeNum)
  // if shape is not amoeba
  // play AIF Sound
  // else
  // play .mp3 sound
```

Amoeba
rotate()
PlaySound

here Rahul didn't touch the already tested code & easily added new class. [scaleability, extensibility]

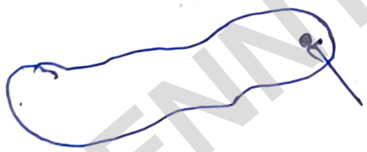
Something again changed :-

→ Amoeba was not to be rotated in the way other shape rotates.

How both of them implemented the rotate() :- Determine the rectangle that surrounds the shape, Calculate the centre point of the rectangle & rotate the shape around the point

De'

how Amoeba was supposed to rotate :-



Again a lot of code was affected, recompiled as tested in Nancy's approach.

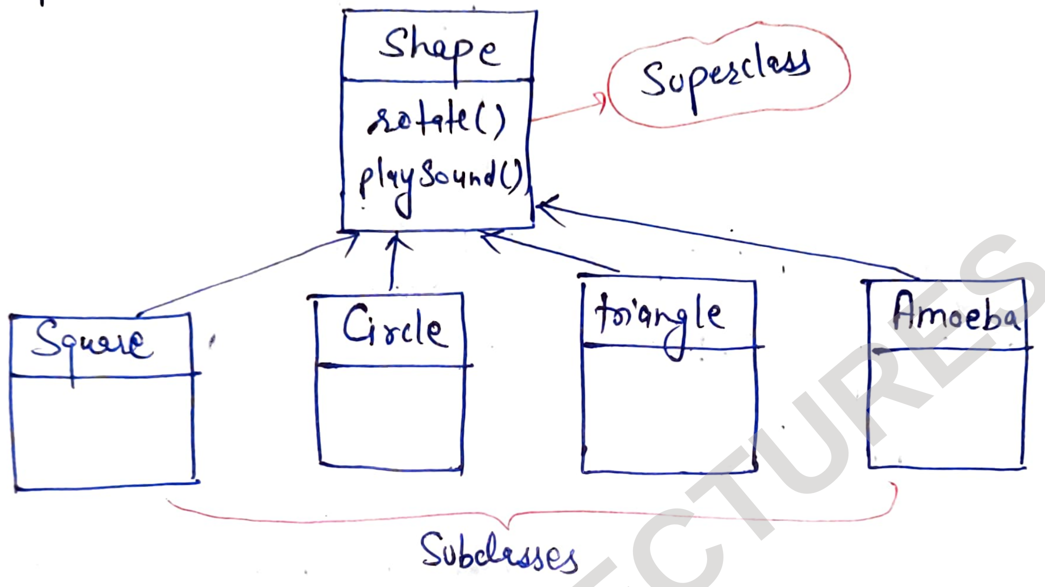
```
rotate(shapenum, x, y)
  // if shape is not amoeba
  // rotate around centre point
  // else
  // use x & y points as the
  // rotationpoint offset &
  // rotate
```

Rahul modified the code of amoebe only without affecting the already tested code.

Amoeba

int x;
int y;
rotate()
  // code
PlaySound()
  // Code

But now Nancy said there is a lot of duplicate code. in Rahul's code. because In each class same 2 methods are being implemented.
Rahul said Don't worry I will fix that using another feature of OOP i.e. Inheritance, and how :-



But one more problem is for Amoeba rotate & playSound are different. Don't worry OOP has solution for this too:-

Polymorphism → overloading
↳ Overriding

For Amoeba we can redefine our rotate() & playSound() methods [This is called overriding]

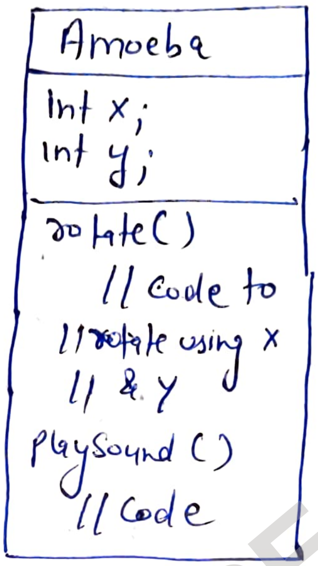
Que:- Nancy has one more question:- how do you tell an Ameba to do something?

Ans:- Using objects. When it's time for, say, the triangle to rotate, the program code invokes (calls) the rotate() method on the triangle object.
So when you need to add something new to the program, just write a new class for the new object type, so the new objects will have their own behaviour

→ Now here we put the data & the methods working on that data in a single unit (class) So this is known as $\boxed{\text{encapsulation.}}$

The data x & y are being used by the rotate method of this class. It is also known as data hiding

We can hide data with access specifiers (private, public, protected)



```
Amoeba
-----------------
Int x;
int y;
-----------------
rotate()
    // Code to
  //rotate using x
  // & y
Playsound ()
   // Code
```

Class

Hence Data security is there in OOP approach.

In OOP, abstraction hides the complex implementation details and only exposes the necessary parts. This mirrors how people interact with real-world systems without needing to know the internal details (working)

e.g:- a coffee machine
a Car
ATM.

NOTE :- A class is not an object but it is used to construct them

Classes & Objects :-

- As we know Java is an Object-Oriented Programming language. It is heavily OO not purely OO.
  [OO means Object Oriented]

- OOP languages helps to model real world Problems or real world scenarios in more natural way means OOP allows us to structure our code in a way that closely resembles how we see and interact with real world.
  OOP helps us create software that mirrors real life. Instead of just writing lines of code, we can group related information & actions into objects that represents real things like cars, students, doctors etc. This makes our code easier to understand and work with because it reflects how we think about & interact with the world around us.
  following points will clarify this idea:

① Objects :- In OOP we represent real-world entities as objects e.g.- a student, a dog, a car or a bank account can be modeled as objects. Each object has properties (attributes) and behavior (methods) that corresponds to the characteristics and actions of that entity.

e.g:- College Management System

classes & objects

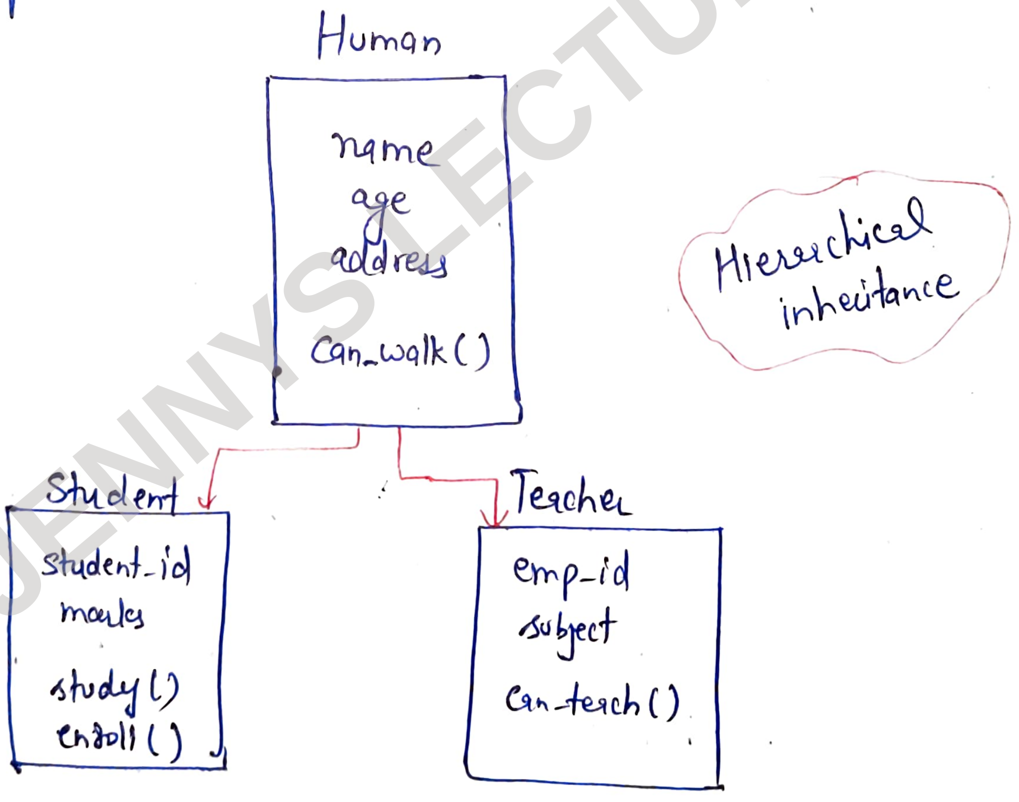
Student :- Attributes(name, Student-id, marks)
methods (study, enroll)

Teacher :- Attributes(name, emp-id, subject)
methods (teach, assign_homework, assign-grades)

Course :- Attributes(c_name, c-code), methods(add, remove)

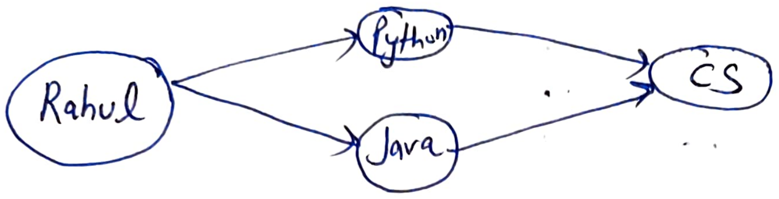
Encapsulation:- It means the data (attributes) and methods (functions) related to an object are bundled together. This mirrors how real world objects have their own state and behaviors allowing for better organization and reduced complexity.

Inheritance:- OOP allows for inheritance where a new class can inherit properties from an existing class. This is same in real-world also. Real-World entities also inherits traits. e.g:- We inherit properties from our ancestors.
We inherits some features as well as behaviour from our parents.



Polymorphism:- more than one form.
eg:- A Payment System
↳ a payment processing system can handle different type of payments such as credit card, debit card wallets etc.
We can achieve this using OOP concepts:

e.g.:- Student registering in any course offered by department
Customer buying some products



So here Student is a class & Rahul is object of that class.

Class is a blueprint or template that defines the structure and behaviour (data and methods) that the objects of that class will have.

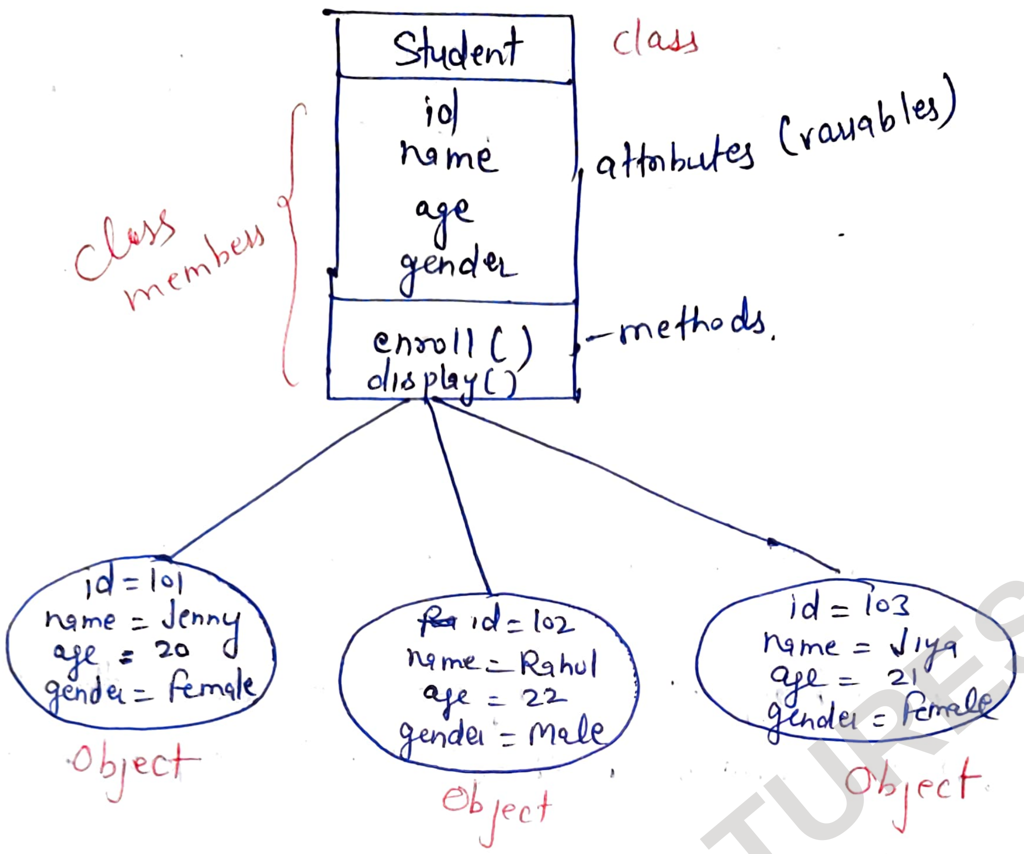
Object is instance of a class and it represents actual entity that exists in memory.

IMP → For class no memory is allocated. Class is a logical thing.

eg:- Map of a house is Class
Actual House is Object

→ Smartphone is class that defines general properties that all smartphones have like brand, model, ScreenSize & behaviour like makeCall(), sendText()

→ Object ≐ a specific smartphone
Apple Iphone 14 with a 6.1 inch screen



```
class Student
{
        int id;          } instance
        string name;     } variable declarations
        int age;
        string gender;
        void display()
method    {   System.out.println("The student "+ name + "has id"+id);
definition }
}
Class StudentMain
{
    public static void main(String[]args)
    {
        Student s1 = new Student();
        s1.id = 100;
        s1.name = "Jenny";
        s1.display();
    }
```