Object-oriented programming

OOP is a programming paradigm.

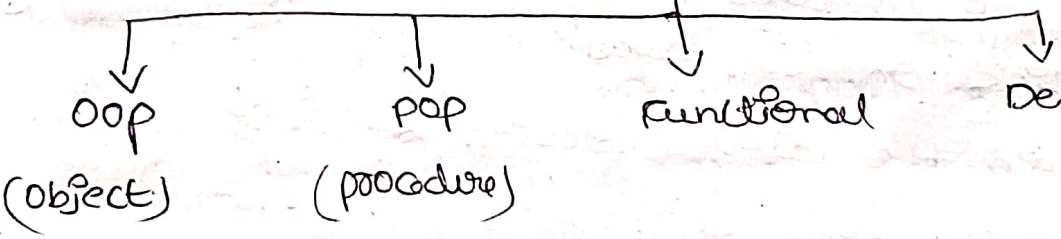
↓

model ⓞⓡ just away of looking at something

Paradigm serves as a model ⓞⓡ framework that shapes how problems are understand & solved.

* In Computer Science, programming paradigm refers to the style ⓞⓡ approch to program

programming paradigm.

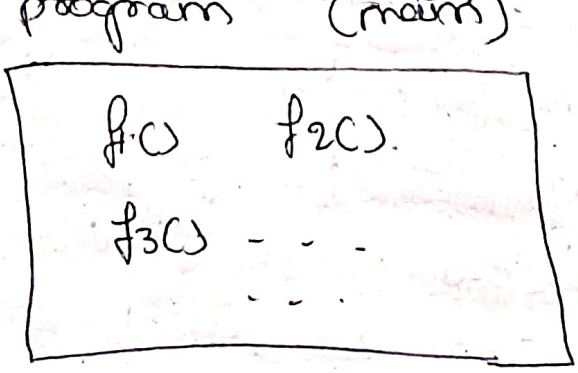


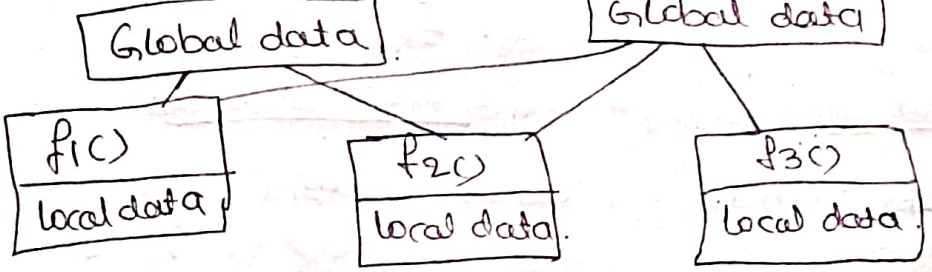
POP (procedure oriented programming) :-

How to do the task.

main focus is on procedure / algorithm / functi

* In each program we can have global data shared by every function.





So there are functions are sharing global data. as well as having their own local data. It means data is not secure here.

Data security is an issue in POP.

Eg:- locker in your house.

If everyone 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

lets take an Ex

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)

Nancy thought what are the things the prgm has to do
→ rotate
→ PlaySound.
} proced user she needs to implmt

[mainly she thought how to do]

Employee 2 (Rahul)

Rahul taught what are the things in the program, Shapes and other things as well

Square	Circle	triangle
rotate()	rotate()	rotate()
PlaySound()	PlaySound()	PlaySound()

Now one more shape added (Amoba) which will rotate like other shapes & but play on .mp3 Sound file.

Nancy update her playSound procedure

```
playSound (Shape num)
// If shape is not amoba
// play AIF Sound
// else
// play .mp3 Sound
```

but

Amoba was not to be rotated. In the way other shapes rotate How both of them implmted the rotate():- Determine the rectangle that surrounds the shape, Calculate the center point of the rectangle & rotate the shape around the point

Rahul added one more class named Amoba.

Amoba
rotate()
PlaySound()

here rahul didn't touch the already tested code & easily add new class (Scalability, extensibility)

but

how ameoba was supposed to rotate :-

around this point not around center point.

In nancy pgm
Again a lot of code was affected, recompiled. as tested pm Nancy's approach.

```
rotate (shapenum, x, y).
// if shape is not amoba
   // rotate around center
   // point
// else
   // use x & y points as
   // the rotation point as
   // offset & rotate.
```

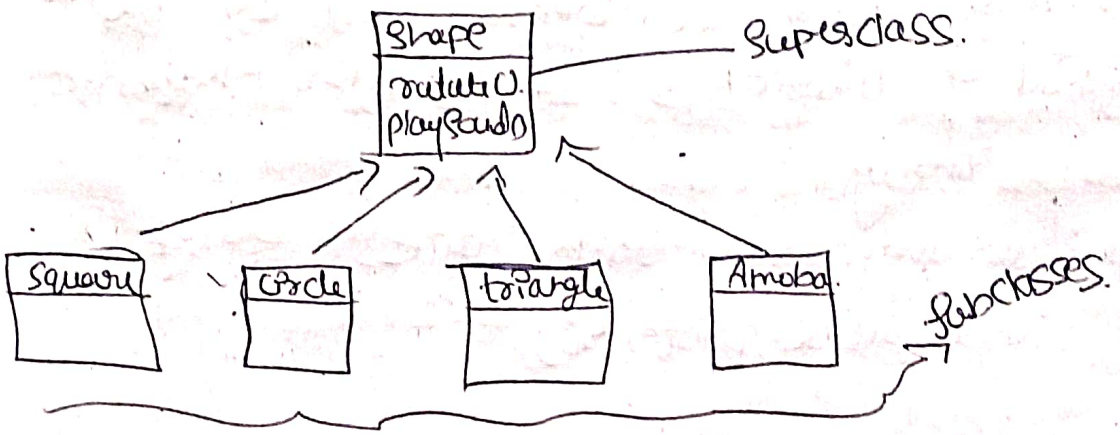
Rahul modified the code of ameoba only without affecting the already tested code

Ameoba.

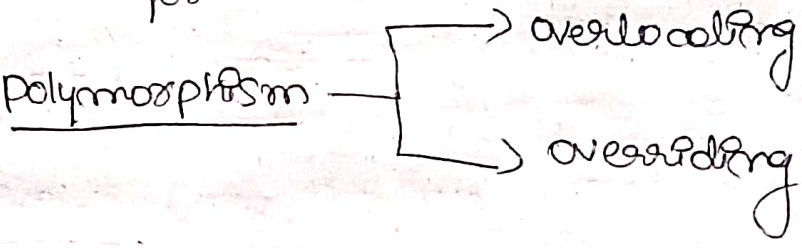
int x
int y.
rotate()
// code
playSound()
// code

But Now nancy said there is a lot of duplicate code in rahul's code because in each class, same name & methods & class are being implemented.

Rahul said don't worry I will fix that using another feature of OOP i.e., Inheritance.



But one more problem is for Amoeba both & PlayCard are different. Don't worry OOP h solution for this too :-



For Amoeba we can redefine our rotate() & PlayCard's methods. (This is called overriding)

Nancy has one more question:- how do you tell an Amoeba to do something

using objects when its time for, say the triangle to rotate, the program code invoke (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 behaviors.

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

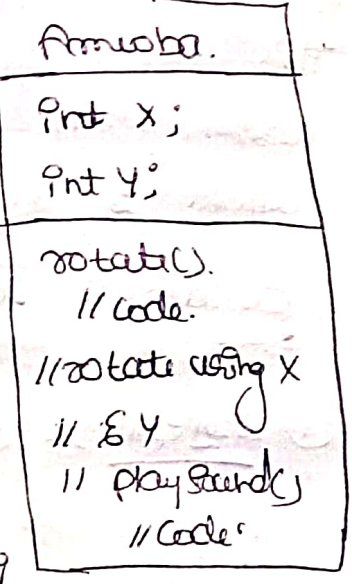
The data x & y are being used by the rotate method of their class. It is also known as data hiding.

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

Hence data security is there in oop approach.

In oop, abstraction hides the complex implementation details and only expose the necessary parts, this mirrors how people interact with real-world systems without needing to know the internal details.

Eg:- a coffee machine
a car.
ATM.



Class.

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

Classes & Objects

* As we know, Java is a object oriented programming language. It is heavily OO not purely OO.
[OO means object oriented]

* OOP languages helps to model real world problems. real world scenarios in more natural way. means oop allows us to structure our code in a way that closely resembles how we see & 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 point will clarify this idea

① objects :- In OOP we represent real-world entities as objects

Eg:- a student, a dog, a car, (or) a bank account can be modeled as objects each object has properties (attributes) and behaviors (methods) that corresponds to the characteristics and actions of that entity.

Eg:- College management System.

Student: Attributes (name, Student, id, marks)
methods (Study, enroll)

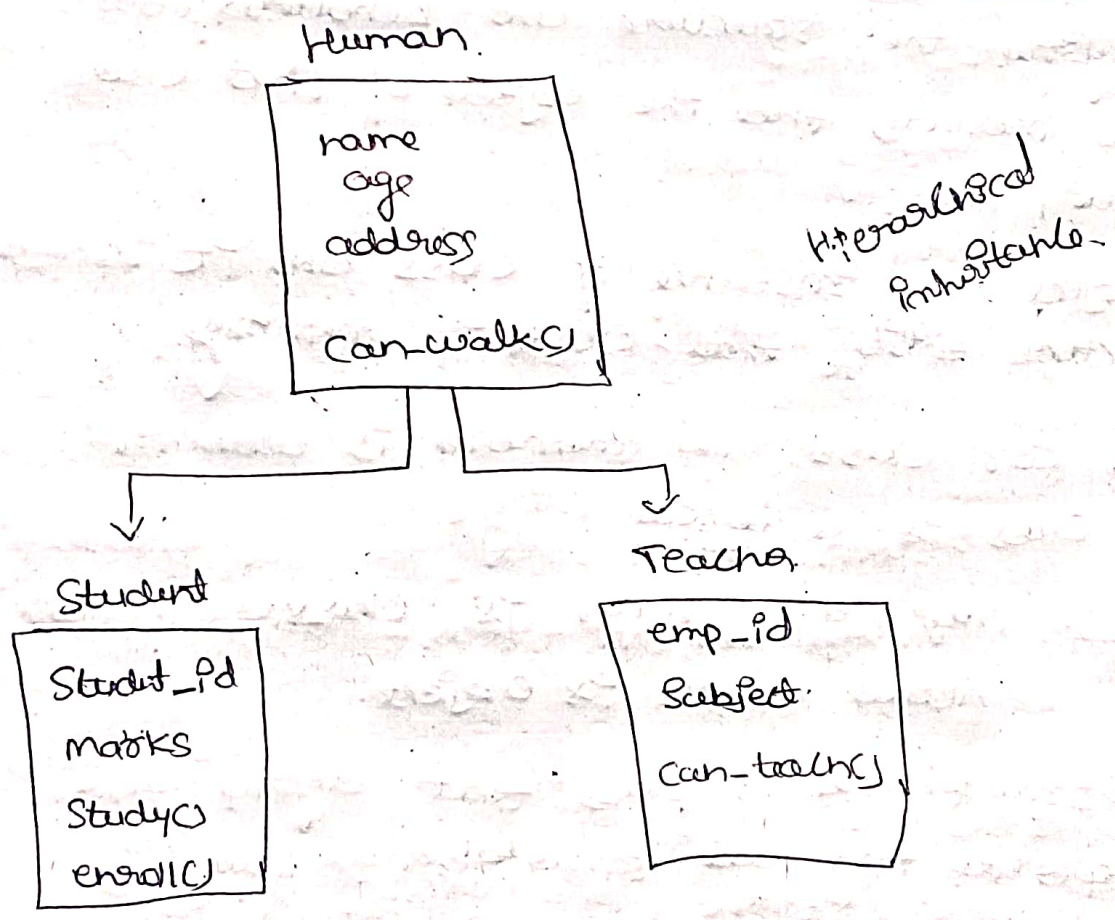
Teacher:- Attributes (name, emp-id, Subject)
methods (teach, assign_HW)

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 inherit traits.

Eg:- We inherit properties from our ancestors. We inherit some features as well as behaviors from our parents



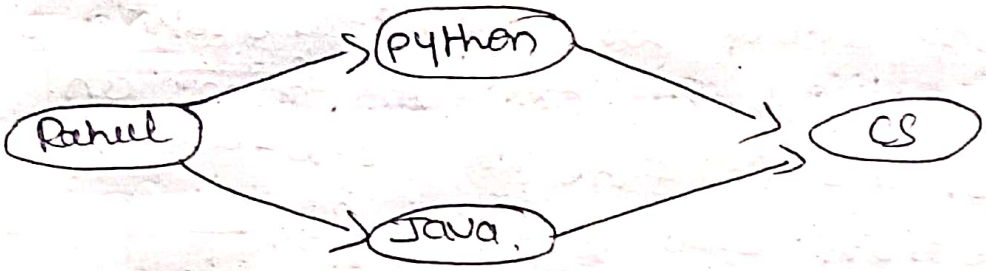
polymorphism:- more than one form.

Eg:- A payment system

a payment processing system can handel different type of payments such as credit card debit card, wallets etc.

we can achieve this using oop concepts.

Eg:- Students 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 behaviours (data & methods) that the objects of that class will have.

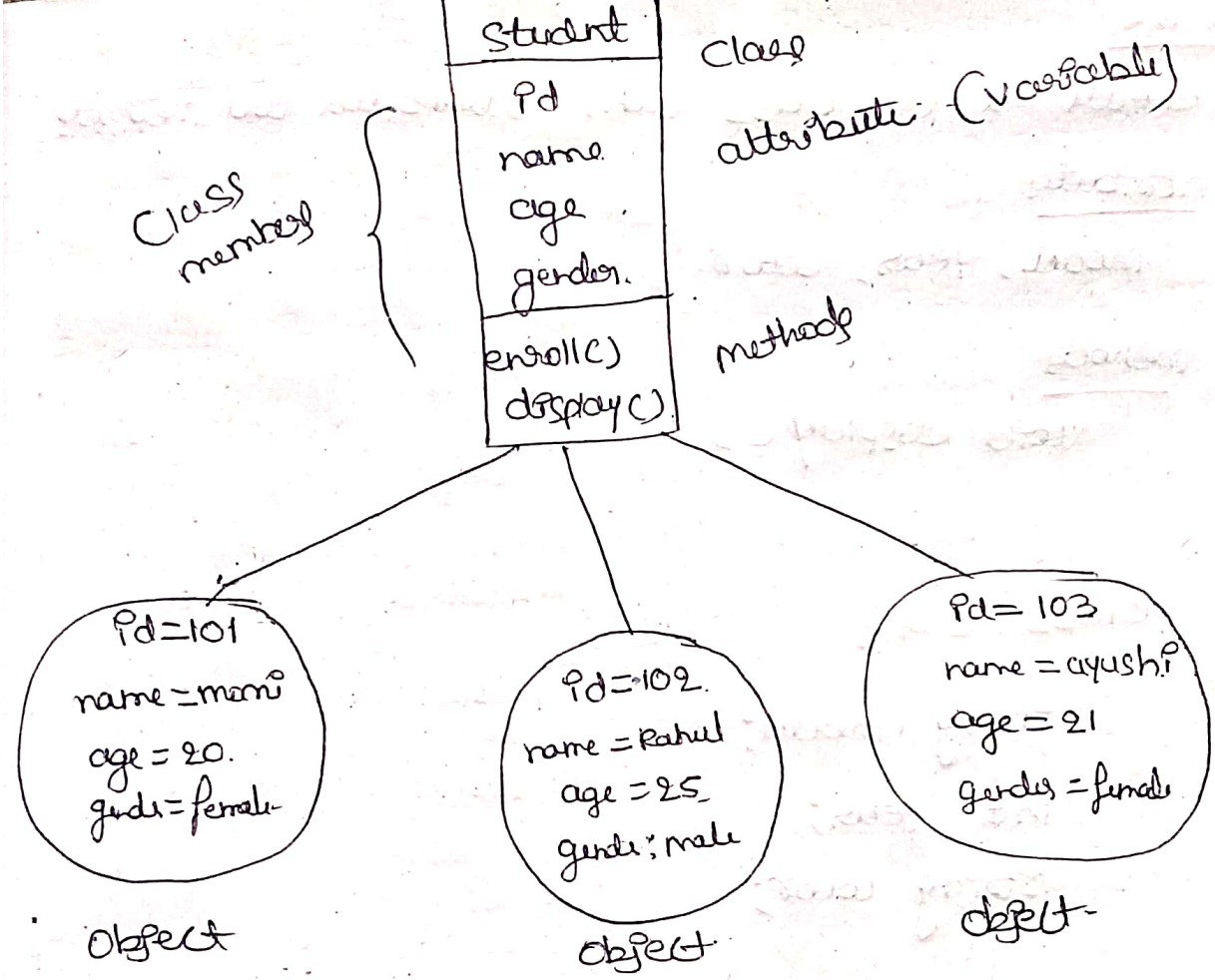
Object is instance of class and it represents actual entity that exists in memory.

Eg:- 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 a brand, model, screenSize & behaviours like makeCall(), SendText().

→ object ⇒ a specific Smartphone Apple iphone 14 with 6inch screen




```
Class Student
{   int Pd;
    String name;
    int age;
    String gender;
void display()
    System.out.println("the Student" + name + "has
                        Pd" + id);
}
}
Class StudentMain{
public static void main(String[] args).
{
    Student S1 = new Student();
     S1. Id = 100;
     S1. name = 'Ram';
     S1. display();
}
```

Instance variable declaration.

Task

create a car class with attributes and methods

Attributes

model, years, Colors.

methods

void display().

```
class Car {
    String model;
    int years;
    String color;
    void display() {
    System.out.println("The car" + model + "
     introduced in" + years + "with" + Colors + "Co
    }
public class Main {
    public static void main(String[] args) {
    Car c = new Car();
     c.model = "Hyundai";
     c.years = 1967;
     c.color = "blue";
      c.display();
    }
}
```