Thursday, February 12, 2026 7:51 PM

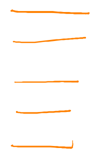
Programs → Set of instructions

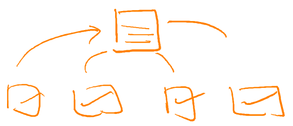
COBOL
MF

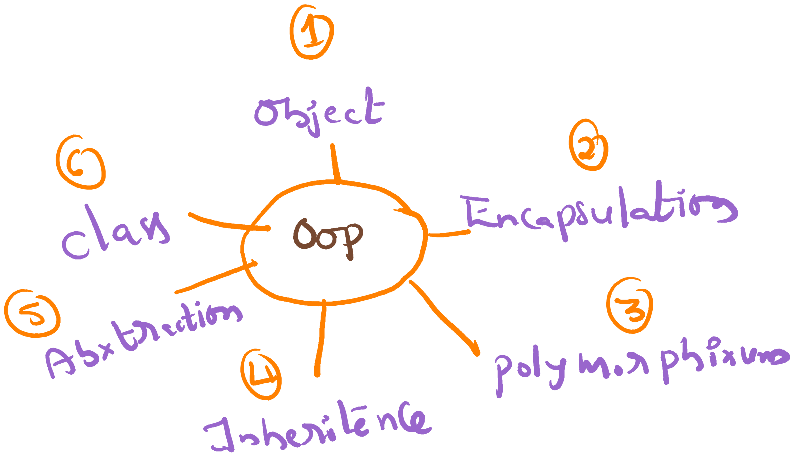
Structured paradigm
start
1000k to 10k ending
→ functions
→ blocks

Procedure Oriented
→ C, ..., C++
→ procedure

Object Oriented
→ classes & objects







Class:- Collection of objects. → blueprint for creating objects.
↓
Set of attributes & methods.

Classes can be created by using Keyword class.

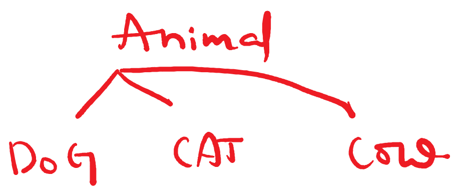
Object: instance of class. → It represents impl of class along with it holds it's own data.

State : attributes reflect properties of object

Behaviour : methods of an object & reflect response of object to other object

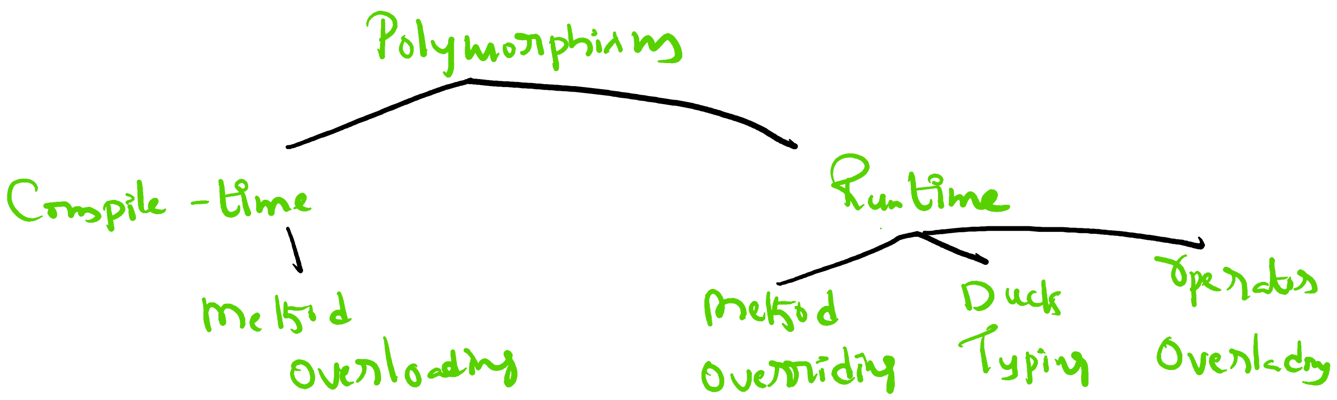
Identity : unique name

Inheritence:- Child class allows to (CC)
↓
Acquire properties and methods of another class (PC)



Polymorphism:-

Same operation, different behaviour.

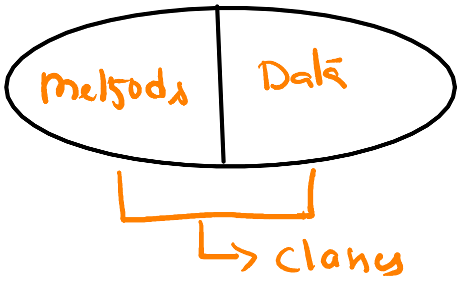


Encapsulation:-

→ bundling of data (attributes) and methods (fun) with in a class.

→ restrict some components, Data privacy,

Encapsulation → classes



Data Abstraction:

→ "what to do" rather how to do?

hides internal imple & expose only necessary functionality.

What is class?

A Class is blueprint of creating an object.

Class :- Blue print of a house

Object:- Actual house built from the blueprint.

programming standards:-

→ Class defines attributes (var) & methods (functions)

→ Object is an instance of class

→ class ✓

Class Cat:
    Species = "mamal"
    def __init__ ( )

→ attributes (var) & methods

→ var → store data with in classes

→ methods define behaviours.

→ Define class:

Syntax
class ClassName:
    <body>

```
import math
✓class Circle:
    def __init__(self, radius):
        self.radius = radius
    def calculate_area(self):
✓       return math.pi * self.radius ** 2
```

__init__ (method) ✓
Constructor ✓
self → refers to current object.
Special method that runs automatically when object is created
→ store the value inside THIS object.
→ Each object has its own copy
C = Circle(5)
C = Circle(2)

```
class Person:
    def __init__(Self, name):
            Self.name = name

    def greet(Self):
        print("Hello, my name is", Self.name)
```

$\rightarrow$ current THIS object

$\rightarrow$ object creation

p1 = Person("ravi")

p1. greet()

$\rightarrow$ method execution