OOPS

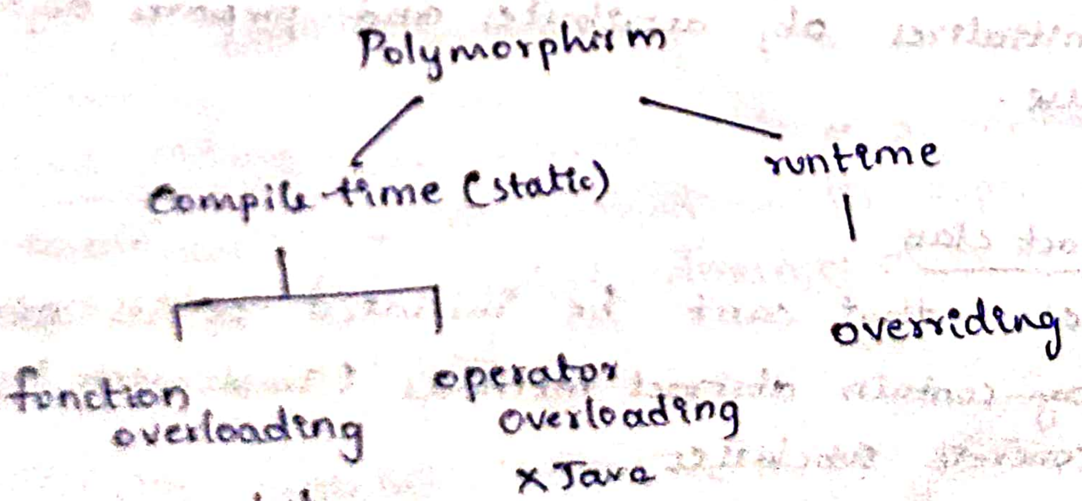
OOP programming paradigm that organizes code into objects, which are instances of classes.
It encapsulates data and behavior together, promoting modularity and reusability.

Encapsulation : concept of bundling data(attributes) and methods(functions) that operate on that data into a single unit (class)
hides internal details of an object and provides controlled access through methods.

Inheritance : mechanism where a new class (subclass or derived class) inherits properties and behaviors from an existing class

promotes code reuse
supports hierarchial relationship

Polymorphism : having many forms
to perform single action in different ways.
In other words, polymorphism allows you to define one interface and have multiple implementations



have same name but different parameters

Overloading	Overriding
→ compile-time polymorphism	→ run-time polymorphism
→ occurs within class	→ performs in 2 class with inheritance relationships
→ same name different signatures	→ same name same signature
→ return type can/can't be the same parameters → change	→ return type - same/covariant
→ static binding	→ dynamic binding
→ private and final methods can be overloaded	→ cannot be overridden
→ poor performance	better
→ arg list - different	→ arg list - same

Constructor

special method that is automatically called when an object is created

initializes obj attributes and prepares object for use.

Abstract class

class that can't be initiated on its own

may contain abstract methods (implement by its concrete subclasses.

Abstraction - process of simplifying complex systems or concepts by focusing on essential details while hiding unnecessary complexities

Interface

defines a set of methods that a class must implement.

allows multiple classes to adhere to same interface, promoting a form of multiple inheritance.

Static data

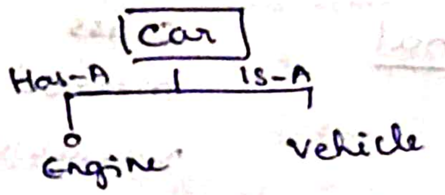
static method belongs to class, not to instance of the class

can be called using class name

often used for utility functions or operations that don't require instance - specific data.

final class - class that cannot be subclassed

Composition - design principle where a class contains objects of other classes as a part of its attributes...



creating complex structures by combing simpler classes

super - used to refer to parent class
- used to call methods & constructors from parent class

method hiding : subclass defines a method with same name as a method in its super class but subclass method doesn't override the superclass method.

Constructor chaining : process of calling one constructor from another within same class or between a subclass and a superclass.

dəamond problem

occurs in lang that support multiple inheritance

$$\begin{tikzcd}
& \boxed{\text{class A}} & \\
\boxed{\text{class B}} \arrow[ur, "\text{extends}"] & & \boxed{\text{class C}} \arrow[ul, "\text{extends}"'] \\
& \boxed{\text{class D}} \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

shallow copy - copies the references of the objects contained within an object

deep copy - results in a completely independent copy of original object and its contained objects.

Virtual method - method declared in a base class that can be overriden by its subclasses

final
- class - can't be subclassed
- method - can't be overridden
- variable - can't be reassigned after it's initial assignment.

SOLID principle

- S Single Responsibility principle
- O Open Closed principle
- L Liskov Substitution principle
- I Iface segregation
- D Dependency Inversion

Destructor - called when an object is about to be destroyed allowing for cleanup tasks.

Encapsulation ② data hiding

helps achieve

① binding the data and code that works on the data in a single unit

practice of making internal data of an object inaccessible to outside world.

Main pillars of OOP — Inheritance — Single / Multiple / Multilevel / Hierarchial / Hybrid
Polymorphism
Encapsulation
Data Abstraction

Single

$$\begin{tikzcd} A \\ B \arrow[u] \end{tikzcd}$$

Multilevel

$$\begin{tikzcd} & A \\ B \arrow[ur] & \\ C \arrow[u] & \end{tikzcd}$$

Multiple

$$\begin{tikzcd} A & & B \\ & C \arrow[ul] \arrow[ur] & \end{tikzcd}$$

Hierarchial

$$\begin{tikzcd} & A & \\ B \arrow[ur] & & C \arrow[ul] \end{tikzcd}$$

hybrid

Polymorphism
Compile time - overloading (phase in which source code is translated into machine code)
find errors (syntax, semantic)
run time - Overriding

phase in which compiled code is executed by the CPU (unexpected conditions like /0 or TLE)

Abstraction display only important information and hiding the implementation details.

pure virtual function - declared in base class with no implementation.

Virtual function - functions in parent class and overridden by subclass

Error - problems that shouldn't be encountered by appli"

Exception - conditions that an appli" might try to catch.

Access specifiers - determine accessibility of methods, classes etc in OOP.