⟹ Inheritance In Java :-

- Java Inheritance is a fundamental concept in OOPs. It is ~~area~~ mechanism in Java by which one class is allowed to inherit the features (fields and methods) of another class.

In java inheritance means creating new classes based on existing ones. A class that inherits from another class can reuse the methods and fields of that class.

→ Inheritance represents ~~is~~ an IS-A relationship, also known as a parent-child relationships. It signifies that a subclass is type of its super class.

eg- A car IS-A Vehicle.
A Dog IS-A Animal.

⟹ How to achieve inheritance in Java :-

- By using the extend keywords for class inheritance

⟹ Advantage of Inheritance :-

- code Reusability :- Inheritance allow a child class to reuse the code of its parent class. ~~to reuse the code of its~~ The child class can reuse the properties and methods of a ~~subcl~~ parent class redes-

reducing duplication of code.

- **Easy to maintences :-**

→ changes made in the parent class automatically propagate to child classes making maintences easier.

- **Method overriding :-**

Inheritances enables method overriding, allowing a child class to provide a specific implementation of a method already defined in its parent class.

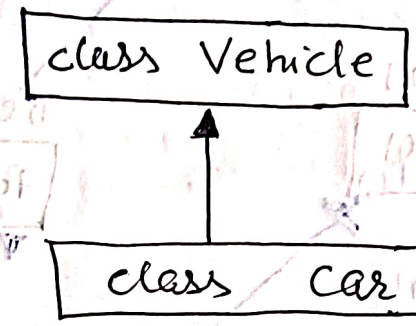
- **polymorphism :-** Inheritances allow for polymorphism, which is ability of an object to take an multiple forms. Inheritance supports runtime polymorphism using method overriding.

- **Abstraction :-** Inheritance in Java allow we to create abstract class that define a common interface or base behavior for a group to related subclesses. This promotes abstraction (hiding implementation details and exposing only essential features) and encapsulation (keep data and method organized and protected).

• Rules in Inheritance :-

1) Single Inheritance :-

→ In single-inheritance, a sub-class is derived from from only one super class. It inherits the properties and behaviour of a single-parent class.



②

→ In Java, private members (like private variable or methods) belong only to the class where they declared. Even if a subclass extend that class, it does not get direct access to those private members.

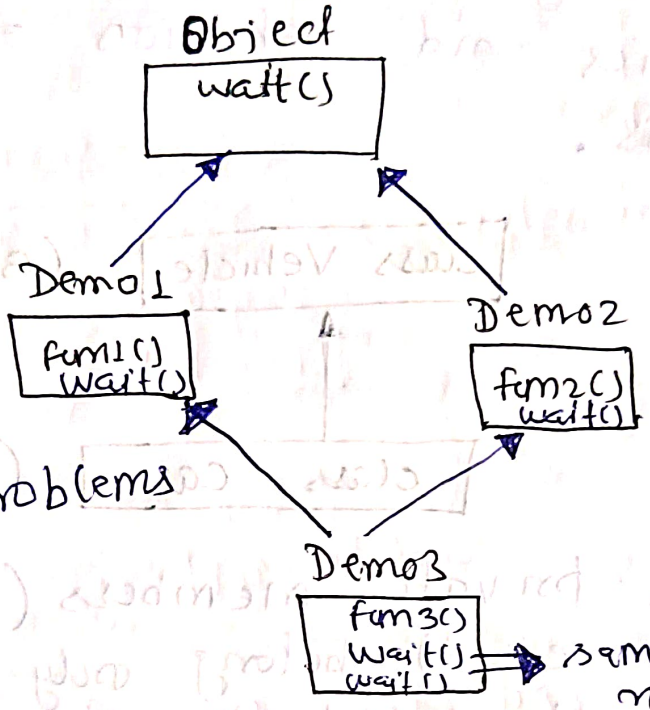
Why

→ Encapsulation :- Java follows the principle of data hiding. Private members are ment to be hidden from outside the class - even from sub classes - to protect internal implementation.

→ Security and Integrity :- Preventing subclass from accessing or modifying sensitive data ensures controlled access using public or protected getter/setter.

Q Multiple Inheritance :-

- Multiple Inheritance is not permitted in Java as it leads to diamond shaped problems which results in ambiguity.



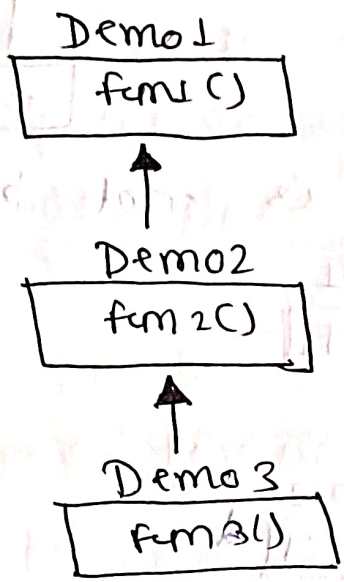
Diamond shaped problems

→ All ^ class except object class will always have one parent class thus we can say that the total Java API is implemented based on inheritance concept.

→ Ambiguity in Java occurs in multiple inheritance when two parent class have the same method. If a child class tries to inherit from both, the compiler cannot decide which method to use. this is called the diamond problem and that's why Java does not support multiple inheritance with class.

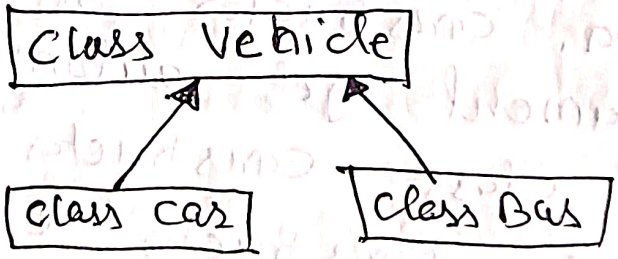
4) Multilevel - Inheritance :-

- One class inherits the properties and behaviors of a parent class and that class is inherited by another class.



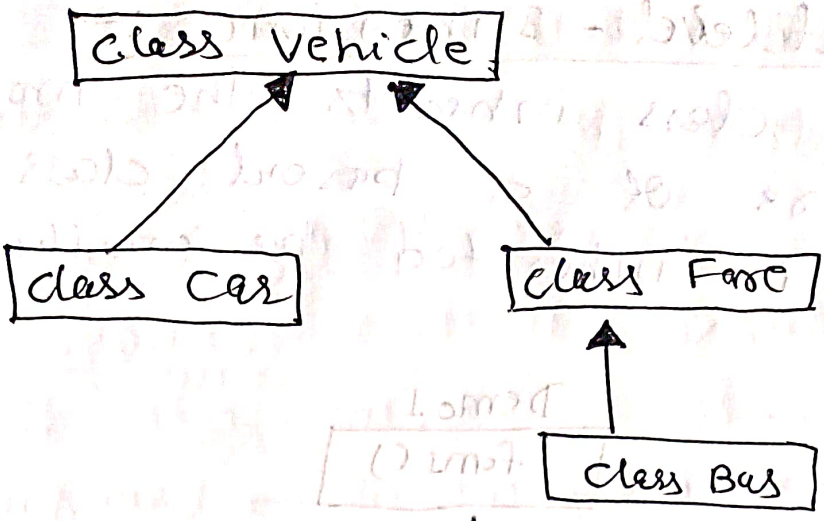
5) Heerarchical Inheritance :-

In heerarchical Inheritance, more than one subclass is inherited from a single base class. i.e more than one derived class is created from a single base class.

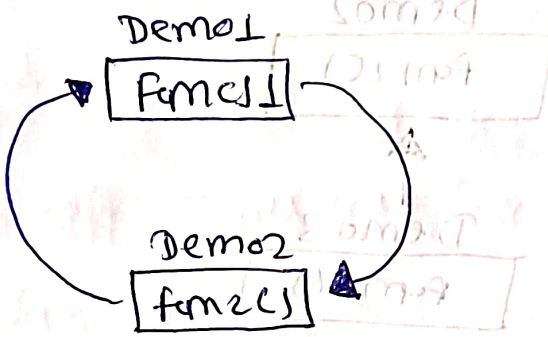


6) Hybrid Inheritance :-

- It is a mix of two or more of the above types of inheritance. In Java, we can achieve Hybrid inheritance only through interfaces if we want to involve multiple inheritence to implement Hybrid inheritence.



→ cyclic Inheritance is not permitted.



④ Constructors do not participate in inheritance.

→ What is constructor chaining?

- In Java, constructor of base class with no argument get automatically called in derived class constructor by a process called as constructor chaining. Constructor Chaining is the processes of child class constructor calling its parent class constructor using super() call.

super()

→ The first line is automatically super() call. Irrespective of whether it is

parameterized constructor or zero parameterized constructor of child class, the super() call will automatically take the control to zero parameterized constructor of parent class.

→ It is used to refer the parent class instance variable

→ It is used to refer to the parent class method.

⟹ Why parent constructor is called?

→ Because the child class object contains the parent class part as well. so the parent class must be properly initialized first that's done through the parent constructor.

→ Default constructor in Inheritence :-

If no constructor is explicitly defined java automatically calls the default constructor of the superclass.

→ In Java, every class implicitly inherits from the class object, unless we explicity extend some other class.

→ java.lang.Object is root class of all classes in Java. If a class doesn't

extend any class, it automatically extend Object.

→ parameterized constructor :—

- the super(parameters) call the parameterized constructors of the parent class. It is part of constructor chaining — where a subclass constructor explicitly call its superclass constructor.

→ first line of constructor only one either super() or this().

→ super() keyword	super keyword
• constructor call	• keyword for accessing parent members.
• calls parent class constructor	• Access variables / methods of parent.
• only inside constructor	• anywhere (method / constr.)
• must be 1st line	• can be used anywhere.

⟹ Disadvantage of Inheritance :-

- **Complexity :-** Inheritance can make the code more complex and harder to understand. This is especially true if the inheritance heerarchy is deep or if multiple inheritence is used.

- **Tight Coupling :-** Inheritance create a tight coupling blw the superclass and subclass making it difficult to make changes to the superclass without affecting the subclass.

group of similar type