

What is Inheritance ?

- Inheritance is inheriting the properties of parent class into child class.
- Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object.
- Inheritance represents the IS-A relationship which is also known as a parent-child relationship, for ex :
 - Dog IS-A Animal
 - Sparrow IS-A Bird
 - Car IS-A Vehicle
 - Programmer IS-A Employee
 - Surgeon IS-A Doctor

Advantages Of Inheritance :-

- Code Reusability
- It promotes runtime polymorphism by allowing method overriding

Disadvantages Of Inheritance :-

- Using inheritance the two classes (parent and child class) gets tightly coupled.

For more updates



Our Channel

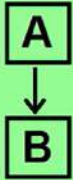


Press the
"Bell Icon"



Types Of Inheritance ?

- **Single Inheritance**



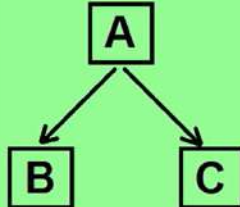
In Single Inheritance one class extends another class (one class only).

- **Multilevel Inheritance**



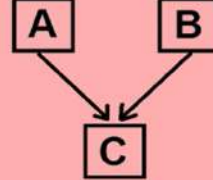
In Multilevel Inheritance, one class can inherit from a derived class. Hence, the derived class becomes the base class for the new class.

- **Hierarchical Inheritance**



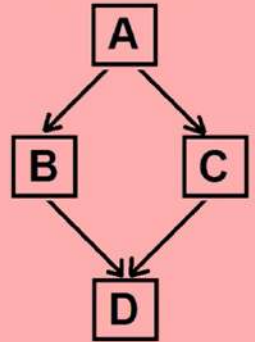
In Hierarchical Inheritance, one class is inherited by many sub classes.

- **Multiple Inheritance**



In Multiple Inheritance, one class extending more than one class. Java does not support multiple inheritance.

- **Hybrid Inheritance**



Hybrid inheritance is a combination of any two inheritances. Java does not support hybrid inheritance.




Important Points Of Inheritance :-

www.smartprogramming.in

- Inheritance is achieved by using “extends” keyword.
- Every class has a super or say parent class i.e. Object class (but object class does not have any parent class)
- Below does not take part in inheritance :
 - **Constructors:** A subclass inherits all the members (fields, methods, and nested classes) from its superclass. Constructors are not members, so they are not inherited by subclasses, but the constructor of the superclass can be invoked from the subclass.
 - **Private members:** A subclass does not inherit the private members of its parent class. However, if the superclass has public or protected methods (like getters and setters) for accessing its private fields, these can also be used by the subclass.
- There can be only one super class, not more than that because java does not support multiple inheritance.

For more updates

 [Subscribe](#)

Our Channel

&

Press the

"Bell Icon"



Relationship Between Classes In Java

- 1. Inheritance (IS-A) :** Child class object carries the body of the Parent class when initiated. Moreover there are certain privileges attach to method overriding to the classes related this way. This relationship exist for code reuse, method overriding and interfacing (through abstract class).

How to Achieve IS-A Relationship : By “extends” keyword.

- 2. Association (HAS-A) :** Association is relation between two separate classes which establishes through their Objects. Association can be one-to-one, one-to-many, many-to-one, many-to-many.

In Object-Oriented programming, an Object communicates to other Object to use functionality and services provided by that object.

There are the two forms of association:

1. Aggregation
2. Composition



Relationship Between Classes In Java

2.1. Aggregation : Without existing container object if there is a chance of existing contained objects , then container & contained objects are weakly associated & this weak association is known as aggregation.

For example : College consists of several professors, without existing college, there may be a chance of existing professor objects, hence college & professor objects are weakly associated & this is known as aggregation.

How to Achieve Aggregation : In aggregation container object holds just reference of contained objects.

For example :

```
final class College
{
    private Professor professor;
    void setProfessor(Professor professor)
    {
        this.professor = professor;
    }
    void teach()
    {
        if (professor != null)
            professor.teach();
    }
}
```

With aggregation, the College also performs its functions through Professor, but the Professor is not always an internal part of the College. Professor may be swapped, or even completely removed. Not only that, but the outside world can still have a reference to the Professor, and tinker with it regardless of whether it's in the College.



Relationship Between Classes In Java

2.2. Composition : Without existing container object if there is no chance of existing contained objects , then container & contained objects are strongly associated & this strong association is known as composition.

For example : College consists of several branches, without existing college, there is no chance of existing branches, hence college & branches are strongly associated & this is known as composition.

How to Achieve Composition : In composition container object holds directly contained objects

For example :

```
final class College
{
    private final Branches branches;
    College(BranchesNames names)
    {
        branches = new Branches(names);
    }
}
```

In the case of composition, the Branches is completely encapsulated by the College. There is no way for the outside world to get a reference to the Branches. The Branches lives and dies with the College.