Relationship Between Classes

=> It is the relationship that we provide between multiple classes
=> Types :-
   1. IS-A Relationship (Inheritance)
   2. HAS-A Relationship (Association)
   3. USES-A Relationship (Dependency)

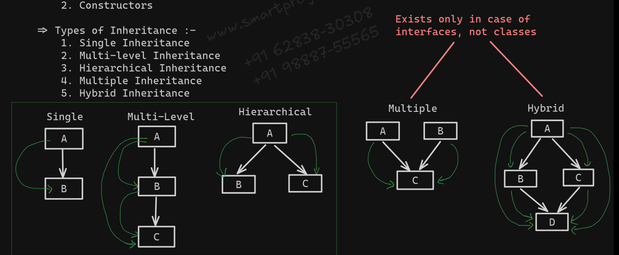
Smart Programming
www.smartprogramming.in
+91 62838-30308
+91 98887-55565

(IS-A Relationship)

Inheritance

=> Inheritance is the process of inheriting the properties of parent class into child class.
=> Real World Example :-
   - Child inheriting Parent properties
=> Inheritance is achieved by :
   - "extends" keyword in case of class
   - "implements" keyword in case of interfaces
=> Syntax :-
```
class Parent
{
    //body
}
class Child extends Parent
{
    // body
}
```
=> Syntax :-
```
interface I1
{
    //body
}
class Child implements I1
{
    //body
}
```
=> Use : Is used for code reusability
=> Example :-
   Program
=> Which members does not inherit :-
   1. Private Members
   2. Constructors

=> Types of Inheritance :-
   1. Single Inheritance
   2. Multi-level Inheritance
   3. Hierarchical Inheritance
   4. Multiple Inheritance
   5. Hybrid Inheritance



=> Disadvantage :-
   -> Classes become tightly-coupled, means if we change any one class, the it will effect the another class

(HAS-A Relationship)

Association

=> It is the concept in which one class uses or interacts with another class by holding a reference to it.

=> Program:
```
class Address
{
    //body
}
class Student
{
    Address addr;   //association

    //body
}
```
=> For eg.
   -> Student HAS-A Address
   -> Car HAS-A Engine

=> Types :-
   1. Aggregation (weak relationship)
      -> Car HAS-A MusicPlayer
   2. Composition (strong relationship)
      -> Car HAS-A Engine

=> Association is highly recommended to use as classes are not tightly-coupled

(USES-A Relationship)

Dependency

=> It is relationship where one class uses another class temporary to perform a specific task
=> Program :
```
class Whiteboard
{
    //body
}
class Teacher
{
    void teachStudents()
    {
        Whiteboard wb = new Whiteboard();
        ----

        //body
    }
}
```
=> Used in frameworks (Spring & Springboot)

Polymorphism

=> Poly means many AND morph means forms
=> It means any entity having multiple forms or can behave in multiple ways

=> Real world example :-
   -> Water - solid, liquid, gas
   -> Sound - animals, river etc

=> It is used to provide the code flexibility

=> Types of Polymorphism :-
   1. Compile Time Polymorphism
   2. Run-Time Polymorphism

Compile Time Polymorphism

=> It is also known as "Static Binding" or "Early Binding"
=> It is achieved by "Method Overloading" or "Operator Overloading"
=> In compile-time polymorphism, Java Compiler decides at compile time that which method is overloaded

=> How to achieve Method Overloading :-
   1. Same name
   2. Same class
   3. Different Parameter
      - Number of parameter
      - Sequence of parameter
      - Type of parameter

Run-Time Polymorphism

=> It is also known as "Dynamic Binding" or "Late Binding"
=> It is achieved by "Method Overriding"
=> In run-time polymorphism, JVM decides at runtime which overridden method it has to invoke based on the actual object (not reference)

=> How to achieve Method Overriding :-
   1. Same name
   2. Different class
   3. Same Parameters
      - Number of parameter
      - Sequence of parameter
      - Type of parameter
   4. IS-A Relationship

=> We use runtime polymorphism to change the parent class method defination

Interview Question

Inheritance
1. What is inheritance in Java, and why is it used?
2. What are the different types of inheritance supported in Java? Which type is not supported directly?
3. What is the super keyword? How is it used in inheritance?
4. How does constructor execution work in inheritance?
5. Can a subclass override a private method of the parent class? Why or why not?
6. What is method overriding, and how does it differ from method overloading?
7. Can a subclass inherit multiple parent classes in Java? Why or why not?
8. How can we prevent a class from being inherited? (final keyword)
9. What happens if a subclass does not define a constructor?
10. What is the difference between is-a and has-a relationships in Java?

HAS-A, USES-A, Composition, and Aggregation
11. What is composition in Java? How is it different from inheritance?
12. Why is composition preferred over inheritance in some cases?
13. Can we replace inheritance with composition? Give an example.
14. How do we implement composition in Java?
15. What happens if an object that contains another object (composition) is destroyed?
16. What is aggregation in Java? How is it different from composition?
17. How is aggregation implemented in Java? Provide an example.
18. How does aggregation affect object lifecycle?
19. Can an aggregated object exist without its parent object?
20. What is the main difference between association, aggregation, and composition in Java?
21. What is a uses-a relationship in Java? How is it different from has-a?
22. How can has-a and uses-a relationships be implemented in Java?

Polymorphism in Java
23. What is polymorphism in Java? Why is it important?
24. What is the difference between compile-time polymorphism and runtime polymorphism?
25. How is method overloading related to polymorphism?
26. How is method overriding related to polymorphism?
27. Can constructors be overloaded in Java? Is it a type of polymorphism?
28. Can we override static methods in Java? Why or why not?
29. What is dynamic method dispatch in Java?
30. How does the JVM determine which overridden method to call at runtime?
31. Can polymorphism be achieved without inheritance in Java? If yes, how?
32. What are the limitations of polymorphism in Java?