**Interview Corner**

1. **What is Java Inheritance?**  
   Inheritance is an OOP mechanism where one class (the *subclass* or *derived class*) acquires the properties (fields and methods) of another class (the *superclass* or *base class*). It models an “is-a” relationship and promotes code reuse and polymorphism.
2. **What are the different types of inheritance in Java?**
   * **Single Inheritance**: A subclass extends one superclass.
   * **Multilevel Inheritance**: A class extends a subclass, forming a chain (e.g., A ← B ← C).
   * **Hierarchical Inheritance**: Multiple subclasses extend the same superclass (e.g., A ← B and A ← C).
   * **Interface Inheritance**: A class implements one or more interfaces (Java’s alternative for multiple inheritance of type).
3. **Why use inheritance in Java?**
   * **Code Reuse**: Write common logic once in the superclass.
   * **Maintainability**: Centralize shared behavior; fixes propagate to all subclasses.
   * **Polymorphism**: Treat different subclasses uniformly through superclass references, enabling flexible and extensible code.
   * **Logical Modeling**: Naturally represent specialized relationships (e.g., Manager is a type of Employee).
4. **How to access any data member or method of the parent class in Java?**
   * **Direct Access**: If the member is public or protected, you can call it directly in the subclass.
   * **super Keyword**:
     + super() invokes the superclass constructor.
     + super.fieldName or super.methodName(...) accesses hidden or overridden members of the superclass.
5. **Is it possible for a class to extend itself in Java inheritance?**  
   No. A class cannot extend itself—doing so would create a cyclic inheritance relationship, which the Java compiler rejects with an error.
6. **What is the output of the program when a derived class extends a base class?**  
   Assuming this typical example:

java

CopyEdit

class Base {

Base() { System.out.println("Base constructor"); }

}

class Derived extends Base {

Derived() { System.out.println("Derived constructor"); }

}

public class Test {

public static void main(String[] args) {

new Derived();

}

}

**Output**:

kotlin

CopyEdit

Base constructor

Derived constructor

The superclass constructor always runs first, followed by the subclass constructor.

1. **Why is multiple inheritance not supported in Java?**  
   Java disallows multiple inheritance of classes to avoid the **“diamond problem”** (ambiguity when two superclasses define the same method). Instead, Java uses interfaces to provide multiple-type inheritance without the pitfalls of stateful multiple inheritance.