In Java, if a class does not explicitly define a constructor, the compiler provides a default no-argument constructor. However, if the superclass does not have a no-argument constructor, you must explicitly call the superclass constructor with the required arguments from the subclass constructor.

A screenshot of a computer program

Description automatically generated

If a superclass has a parameterized constructor, the subclass must explicitly call it using super() unless a default constructor is provided.

**Field Hiding**: If a subclass defines a field with the same name as a field in the superclass, it hides the superclass field rather than overriding it.

public class Subclass extends Superclass {

public int fieldName; // Hides the field in Superclass

}

**Static Methods**: Static methods are not inherited in the same way as instance methods. They belong to the class, not the instance.

**Dynamic Method Dispatch**: Java supports polymorphism, allowing a superclass reference to point to a subclass object. Method calls are resolved at runtime based on the object's type.

Rule to be followed: same name, return type, and parameters

Superclass obj = new Subclass();

obj.method();

\*\*@Override 🡪 This helps catch errors at compile-time if the method does not correctly override a method in the superclass.

\*\*\*The "Diamond Problem" in inheritance is a common issue in languages that support multiple inheritance. It occurs when a class inherits from two classes that both inherit from a common superclass, creating an ambiguity. Java avoids the diamond problem by not allowing multiple inheritance with classes, but it can still be encountered when dealing with interfaces.

Consider a scenario where:

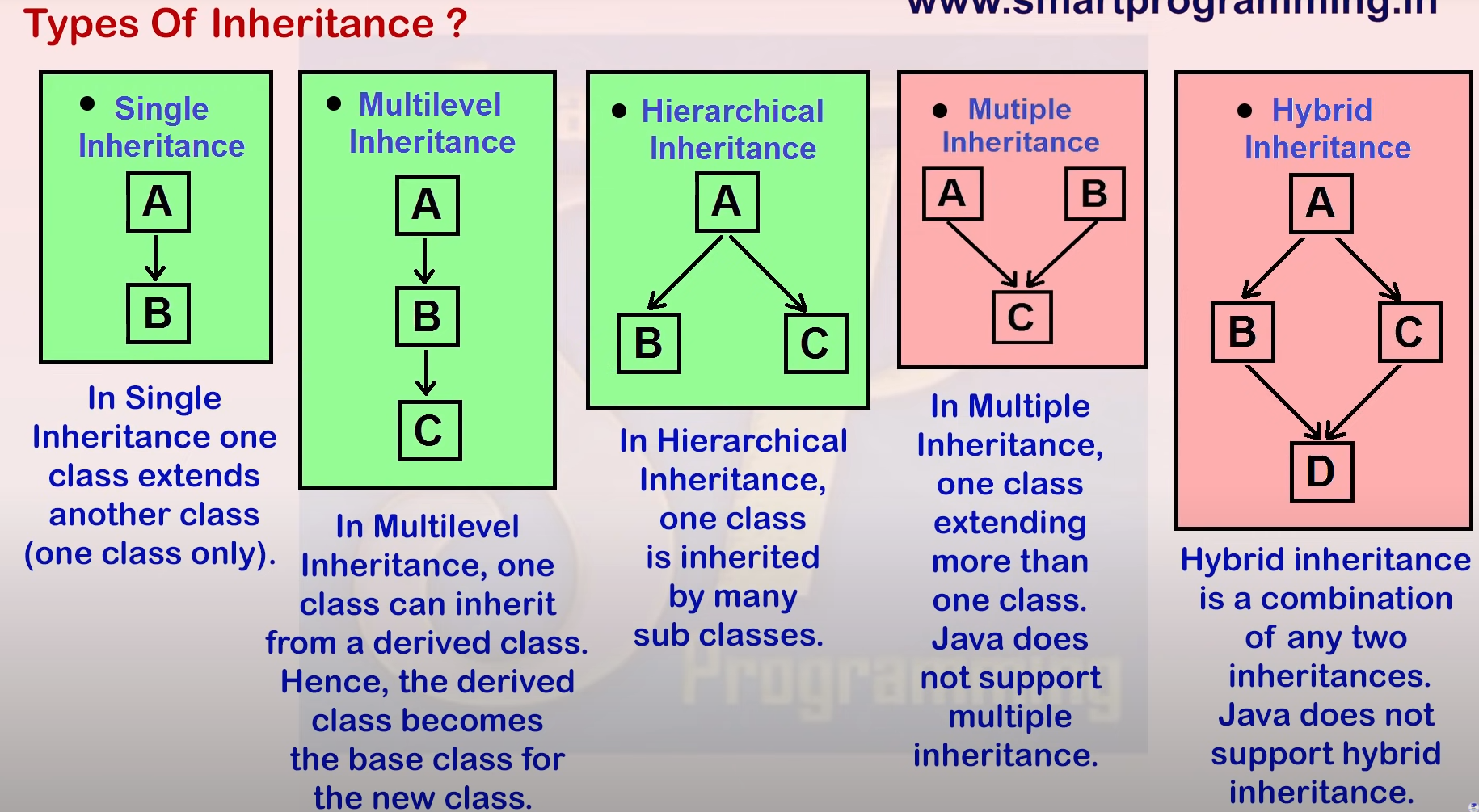
* Class A is the base class.
* Class B and Class C both inherit from Class A.
* Class D inherits from both Class B and Class C.

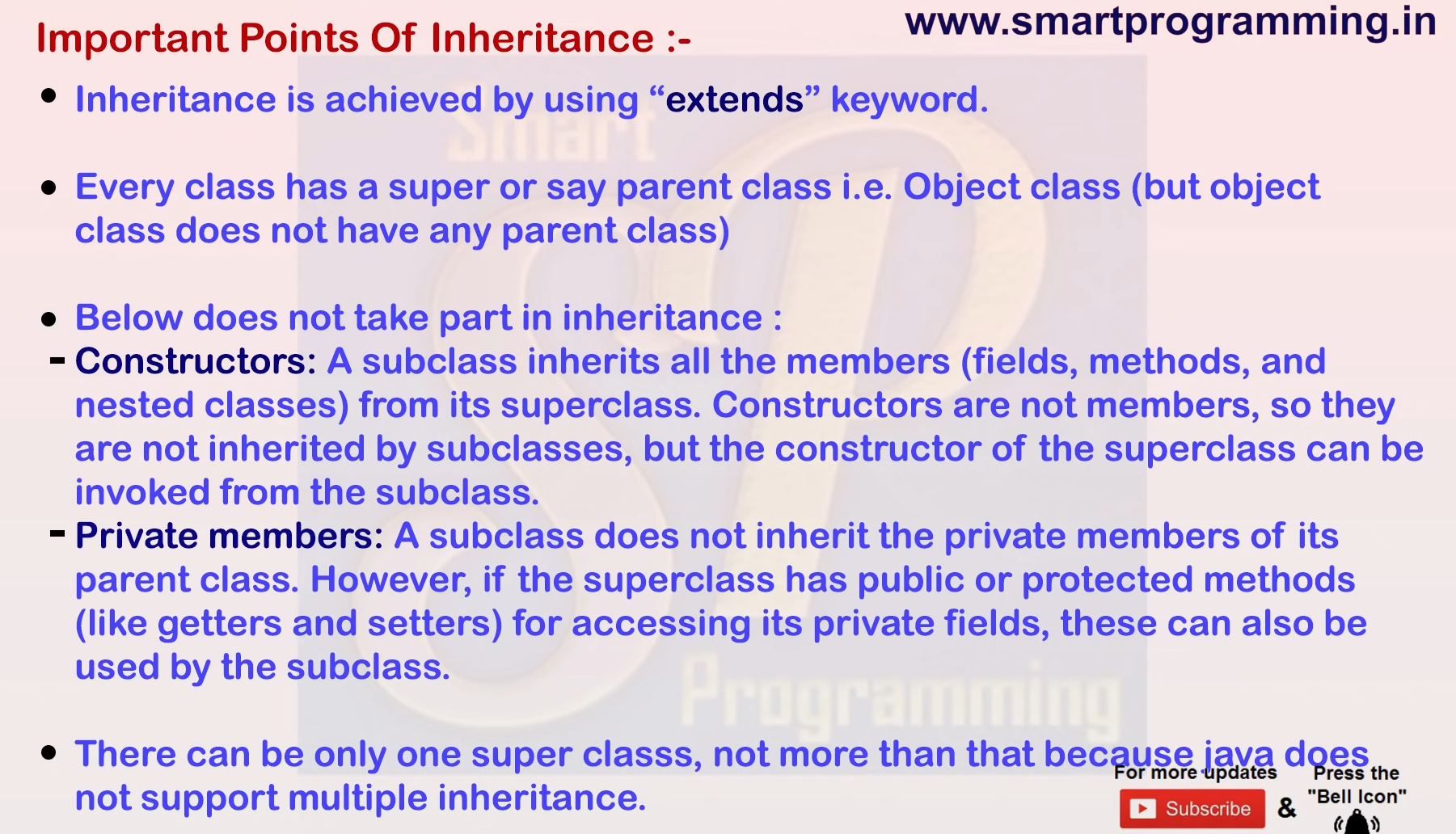
If Class B and Class C both override a method from Class A, and Class D inherits from both B and C, it's unclear which version of the method D should inherit. This creates ambiguity, known as the diamond problem.

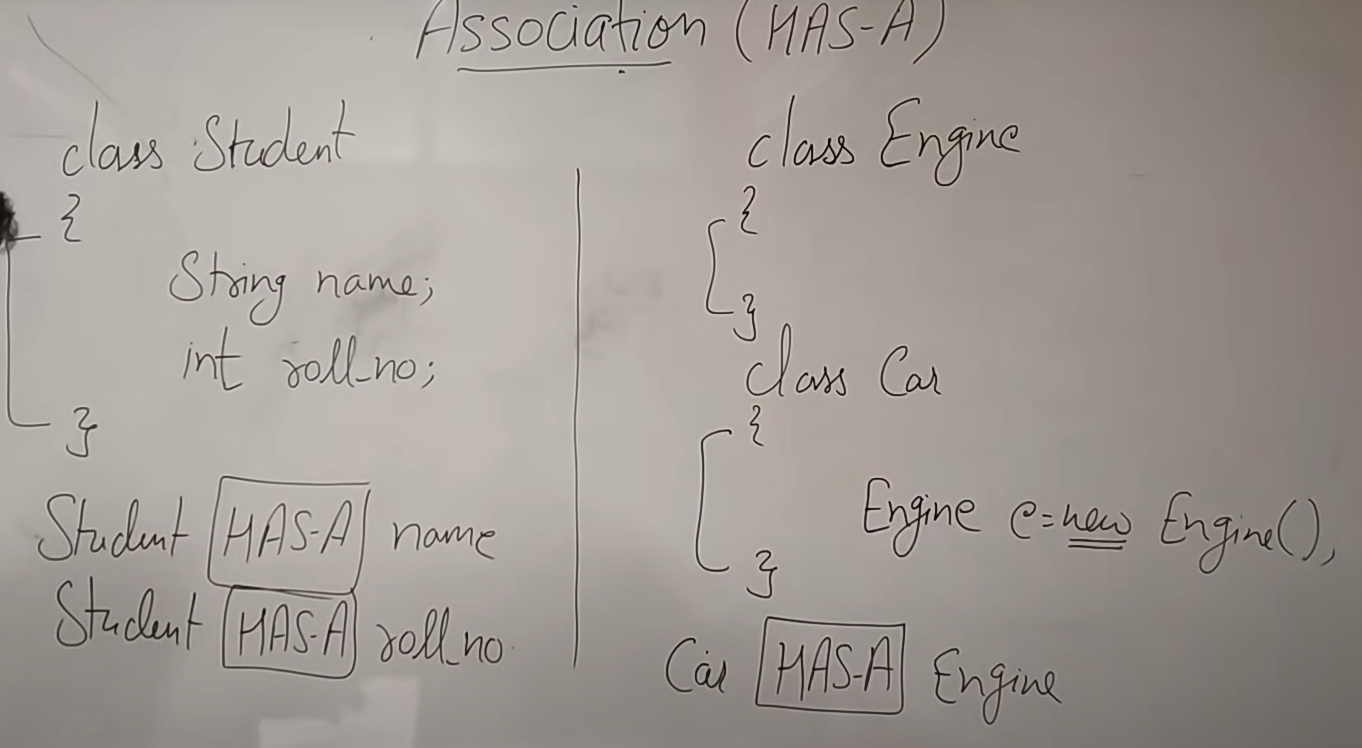
class Vehicle {}

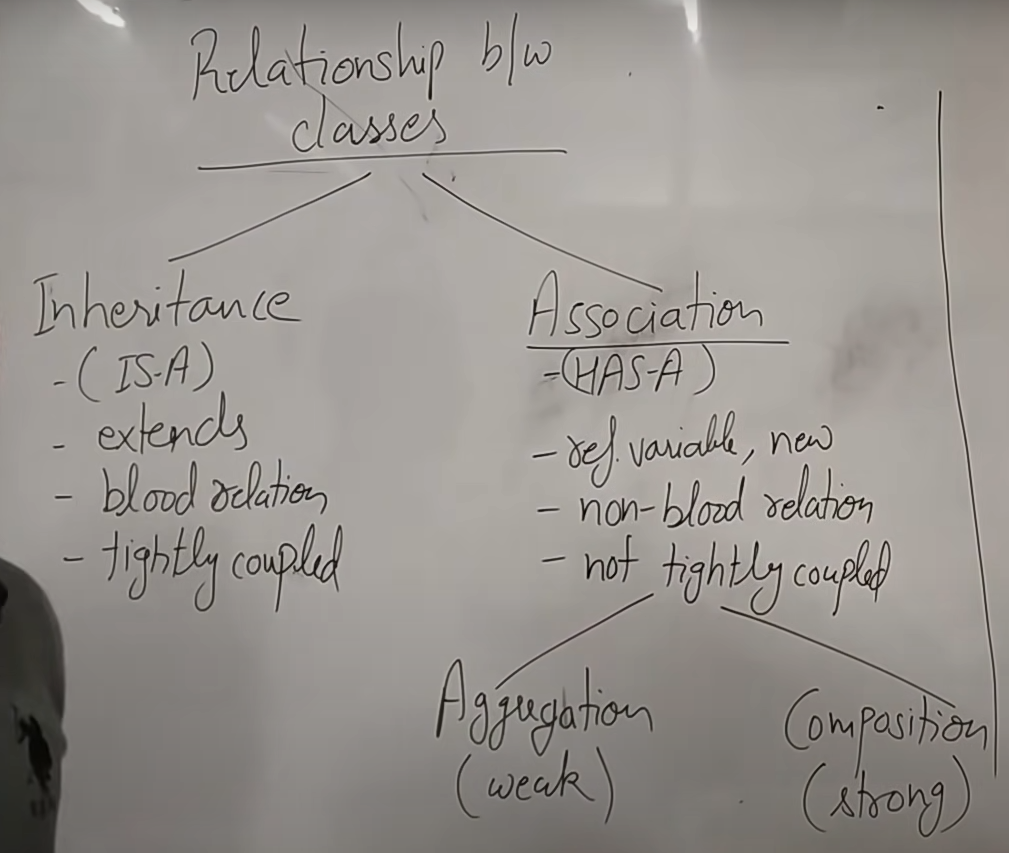
class Car extends Vehicle {}

Car IS-A Vehicle









A whiteboard with writing on it

AI-generated content may be incorrect.

A close-up of a paper

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Q1. Can a subclass access the private members of a superclass?

Answer: No, a subclass cannot directly access the private members (fields or methods) of a superclass. However, it can access them through public or protected getters and setters or via protected methods if they exist in the superclass.

Q2. Can we override a static method in Java?

Answer: No, static methods are not overridden in Java. They are resolved at compile time based on the reference type, not at runtime. However, a subclass can hide a static method from the superclass by defining a static method with the same signature.

Q3. What is the use of the instanceof operator in inheritance?

Answer: The instanceof operator is used to check whether an object is an instance of a specific class or implements an interface. It returns true if the object is an instance of the class or any of its subclasses.

Q4. What is the order of constructor execution in inheritance?

Answer: In Java, when a subclass object is created:

The constructor of the superclass is called first (either implicitly or explicitly using super()).

Then, the constructor of the subclass is executed.

Q5. Can a constructor be inherited in Java?

Answer: No, constructors are not inherited in Java. However, a subclass can call the superclass constructor using super() to initialize the parent class’s fields.

Q6.