**Inheritance in Java:**

Inheritance is one of the fundamental concepts of object-oriented programming (OOP),it allows one class to acquire the properties and behaviors that is (fields and methods) of another class. The class that is being inherited from is called the superclass or parent class, and the class that inherits is called the subclass or child class. The subclass inherits the super class using extends keyword.

**Real-life based Example:**

Suppose there's a family with a grandfather, a father, and a son. Each generation inherits certain traits from the previous one. For instance, the grandfather passes on his genetics to the father, who in turn passes them on to his son.

In this scenario:

Grandfather is the superclass, passing on traits.

Father is the subclass inheriting traits from the grandfather and adding his own.

Son is another subclass inheriting traits from both the father and grandfather.

**Programming based example:**

Suppose:

* We have a Shape class with a method draw() that prints "Drawing a shape."
* There's a Circle class that extends the Shape class. This means Circle inherits the draw() method from Shape.
* Inside the Circle class, we override the draw() method to print "Drawing a circle."

**Example In Android Development: (Not Required for now)**

In Android development, inheritance is used to create new UI components based on existing ones. For example, you can create a new button with special features by inheriting from the standard button. This new button inherits all the behavior of the standard button while allowing you to add or modify functionality as needed. This approach helps in building consistent and customized user interfaces efficiently.

Under:

**Types of Inheritance in Java:**

Java supports various types of inheritance:

1. Single-level Inheritance: where a subclass can inherit from only one super-class.
2. Multilevel Inheritance: A subclass can inherit from another class, which itself is a subclass of another class.
3. Hierarchical Inheritance: Multiple subclasses can inherit from the same superclass, creating a hierarchical relationship.
4. Multiple Inheritance (Through Interfaces): A subclass can implement multiple interfaces, allowing it to inherit the behavior from multiple sources. Java does not support multiple inheritance but it is impleamented through interface
5. Hybrid Inheritance (Through Interfaces): It is a combination of multiple and hierarchical inheritance. Java achieves hybrid inheritance using interfaces.

**Super Keyword In Inheritance:**

Inheritance invloves super Keyword in java

In Java, the super keyword is used to refer to the superclass. It allows you to access the superclass's members (fields and methods) from the subclass. You can use super to call the superclass's constructor explicitly or to invoke overridden methods from the superclass.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Diamond Problem:**

The diamond problem in Java occurs when a class inherits from two interfaces that both declare the same default method. This leads to ambiguity about which method implementation to choose. For example, if class A and class B both have a default method with the same name, and class C inherits from both A and B, Java encounters a compilation error due to the ambiguity of method resolution. This problem arises in scenarios of multiple inheritance, causing conflicts in method implementation selection.

**The Object Class:**

The Object class is the root class of all classes in Java, and every class implicitly inherits from it. It is the most fundamental class and provides essential methods, such as toString(), equals(), hashCode(), and more. These methods can be overridden in subclasses to provide custom behaviour.

Note : When asked about OOPs concepts first list them like :

The four fundamental concepts of OOPs are

1. Inheritance

2. Polymorphism

3. Encapsulation

4. Abstraction

Than explain each concept