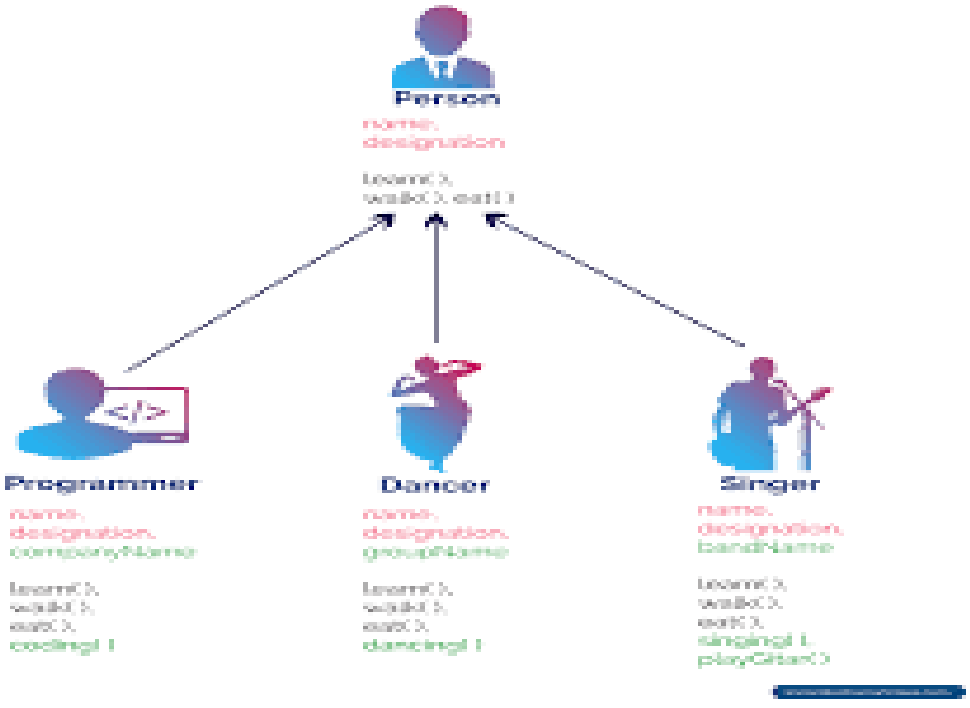
Java Inheritance (Subclass and Superclass)

In Java, it is possible to inherit attributes and methods from one class to another. We group the "inheritance concept" into two categories:

- subclass (child) - the class that inherits from another class
- superclass (parent) - the class being inherited from

To inherit from a class, use the `extends` keyword.

In the example below, the `Car` class (subclass) inherits the attributes and methods from the `Vehicle` class (superclass):



Inheritance Syntax in Java

```
class derived_class extends base_class
{
   //methods
   //fields
}
```

General format for Inheritance

```
class superclass
{
 // superclass data variables
 // superclass member functions
}
class subclass extends superclass
{
 // subclass data variables
 // subclass member functions
}
```

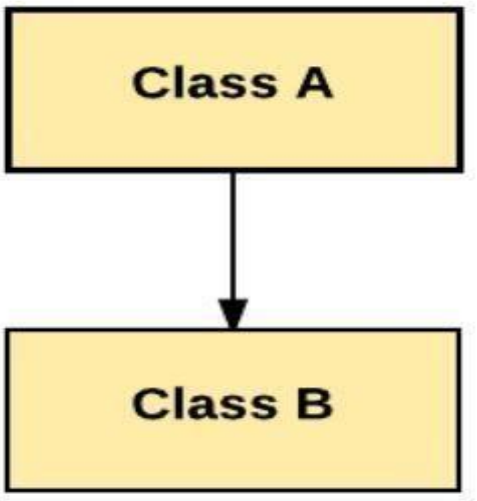
Example

```
class Vehicle {
```

```
  protected String brand = "Ford";         // Vehicle attribute
  public void honk() {                     // Vehicle method
    System.out.println("Tuut, tuut!");
  }
}

class Car extends Vehicle {
  private String modelName = "Mustang";    // Car attribute
  public static void main(String[] args) {

    // Create a myCar object
    Car myCar = new Car();

    // Call the honk() method (from the Vehicle class) on the myCar
object
    myCar.honk();

    // Display the value of the brand attribute (from the Vehicle
class) and the value of the modelName from the Car class
    System.out.println(myCar.brand + " " + myCar.modelName);
  }
}
```

Single Inheritance

As the title indicates, just one class is subject to this kind of inheritance. The parent class gives rise to just one child class. The attributes in this sort of inheritance are only descended from one parent class, at most. Code reuse and the implementation of new features are made easier because the attributes are descended from a single base class. Below is a flowchart of a single inheritance:



In Inheritance, we can access superclass methods and variables. We can also access subclass methods and variables through subclass objects only. We have to take care of superclass and subclass methods, and variable names shouldn't conflict.

Multiple Inheritance in Java

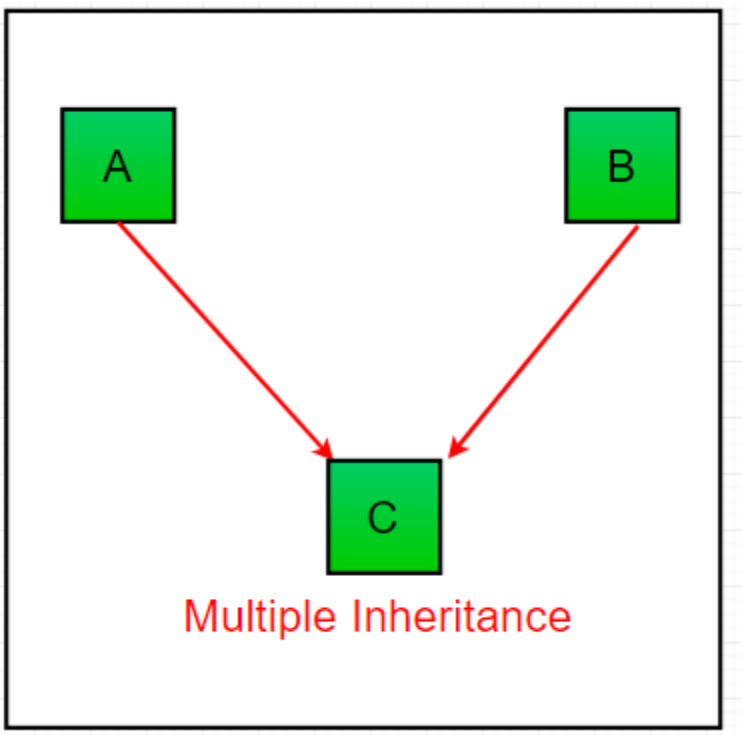
Defining derived class from numerous base classes is known as 'Multiple Inheritance'. In this case, there is more than one superclass, and there can be one or more subclasses.

Multiple inheritances are available in object-oriented programming with C++, but it is not available in Java.

Java developers want to use multiple inheritances in some cases. Fortunately, Java developers have interface concepts expecting the developers to achieve multiple inheritances by using multiple interfaces.

A subclass may inherit features from many parent classes under the concept of multiple inheritance. Contrary to popular belief, multiple inheritances are not the same as multi-level inheritance because the newly derived class in multiple inheritances may

have more than one superclass. There are no limitations, and this newly derived class is free to inherit the features from the superclasses it has inherited from. Interfaces in Java can be used to achieve multiple inheritances.



Ex: class Myclass implements interface1, interface2,….

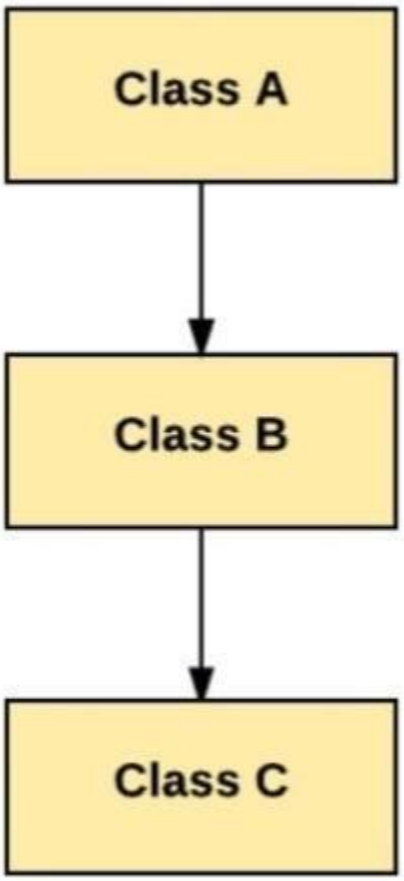
Multi-Level Inheritance in Java

In Multi-Level Inheritance in Java, a class extends to another class that is already extended from another class. For example, if there is a class A that extends class B and class B extends from another class C, then this scenario is known to follow Multi-level Inheritance.

We can take an example of three classes, class Vehicle, class Car, and class SUV. Here, the class Vehicle is the grandfather class. The class Car extends class Vehicle and the class SUV extends class Car.

At least two classes, if not more, are involved in the multi-level inheritance. A subclass that has just been formed becomes the base class for a new class, and one class inherits the features from its parent class.

As the name implies, numerous base classes are involved in multi-level inheritance. As the newly derived class from the parent class becomes the base class for another newly derived class, the inherited features in multilevel inheritance in Java likewise come from several base classes.



Hybrid Inheritance in Java

Hybrid Inheritance in Java is a combination of inheritance. In this type of Inheritance, more than one kind of inheritance is observed. For example, if we have class A and class B that extend class C and then there is another class D that extends class A, then this type of Inheritance is known as Hybrid Inheritance.

Why? Because we clearly observe that there is two kinds of inheritance here- Hierarchical and Single Inheritance.

A hybrid inheritance combines more than two inheritance types, such as multiple and single. Interfaces are the sole means through which it is possible because Java does not enable multiple inheritance. In essence, it combines straightforward, numerous, and hierarchical inheritances.

In the diagram shown below, we see another example of Hybrid Inheritance.