# **Inheritance**

The process by which one class acquires the properties (data members) and functionalities(methods) of another class is called inheritance. The aim of inheritance is to provide the reusability of code so that a class has to write only the unique features and rest of the common properties and functionalities can be extended from another class.

Child Class:  
The class that extends the features of another class is known as child class, sub class or derived class.

Parent Class:  
The class whose properties and functionalities are used(inherited) by another class is known as parent class, super class or Base class.

Example:

ParentClass.java

**package** FPPackage;

**public** **class** ParentClass {

String name = "Subbu";

String role = "Manager";

String company = "IBM";

**public** **static** **void** displayDetails() {

ParentClass pc = **new** ParentClass();

System.***out***.println("Name is "+pc.name);

System.***out***.println("Role is "+pc.role);

System.***out***.println("Company is "+pc.company);

}

}

ChildClass.java

**package** FPPackage;

**public** **class** ChildClass **extends** ParentClass{

**public** **static** **void** main(String[] args) {

ChildClass cl = **new** ChildClass();

System.***out***.println("Name is "+cl.name);

System.***out***.println("Role is "+cl.role);

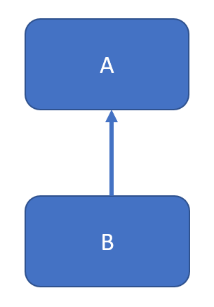
System.***out***.println("Company is "+cl.company);

}

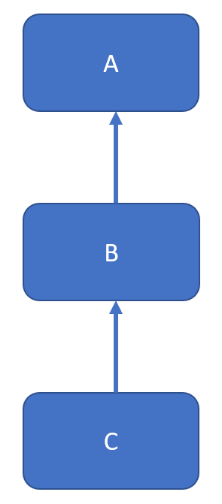
}

**Types of Inheritance:**

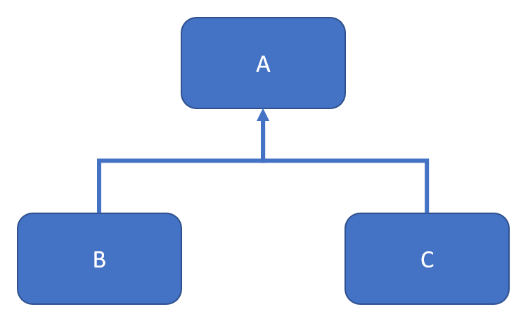
**Single Inheritance:** When a class extends to another one class only, then it is called single inheritance.



**Multilevel Inheritance:** In Multilevel inheritance, we have a base class and we extend this class from a child class and this class will be extended from another child class.

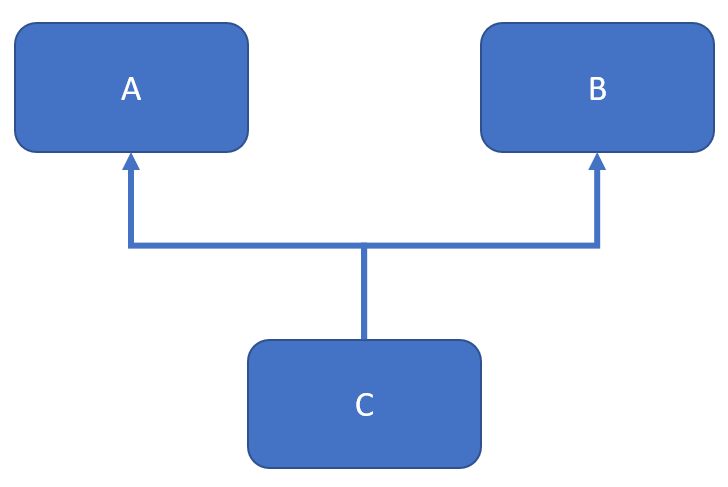
****

**Hierarchical Inheritance:** In hierarchical inheritance a single class will be extended by multiple classes.

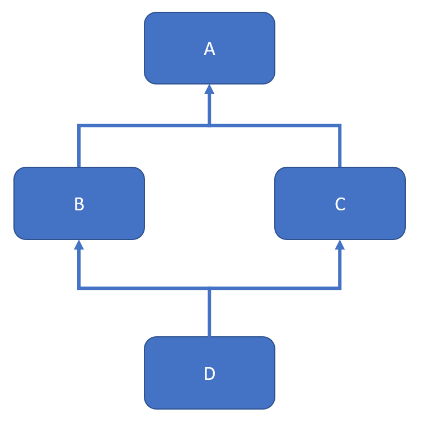


**Multiple Inheritance:** In multiple inheritance a single class extends multiple classes.

Java doesn’t support multiple inheritance for classes. It is supported for interfaces only.



**Hybrid Inheritance:** It’s a combination of single and multiple inheritance.



**Constructors and Inheritance:**

When the constructor of a child class is invoked, it by default invokes the constructor of the parent class.

**Parent Class:**

**package** FPPackage;

**public** **class** ParentClass {

String name = "Subbu";

String role = "Manager";

String company = "IBM";

ParentClass(){

System.***out***.println("Parent Class Constructor");

}

**public** **static** **void** displayDetails() {

ParentClass pc = **new** ParentClass();

System.***out***.println("Name is "+pc.name);

System.***out***.println("Role is "+pc.role);

System.***out***.println("Company is "+pc.company);

}

}

**Child Class:**

**package** FPPackage;

**public** **class** ChildClass **extends** ParentClass{

ChildClass(){

System.***out***.println("Child Class Constructor");

}

**public** **static** **void** main(String[] args) {

ChildClass cl = **new** ChildClass();

System.***out***.println("Name is "+cl.name);

System.***out***.println("Role is "+cl.role);

System.***out***.println("Company is "+cl.company);

}

}

**Inheritance and Method Overriding:**

If we declare the same method of parent class in child class, it is called method overriding.

We can call the child method class using child class object.

We can call the parent class method using super keyword.

Example:

ParentClass:

**package** FPPackage;

**public** **class** ParentClass {

ParentClass(){

System.***out***.println("Parent Class Constructor");

}

**public** **void** displayDetails() {

System.***out***.println("I am parent class");

}

}

ChildClass:

**package** FPPackage;

**public** **class** ChildClass **extends** ParentClass{

ChildClass(){

System.***out***.println("Child Class Constructor");

}

**public** **void** displayDetails() {

System.***out***.println("I belong to child class");

**super**.displayDetails();

}

**public** **static** **void** main(String[] args) {

ChildClass cl = **new** ChildClass();

cl.displayDetails();

}

}