**Inheritance-**

The process of creating the new class by using the existing class functionality called as Inheritance.

Or

It is a mechanism in which one class acquires the property of another class

Inheritance means simply reusability.

It is called as - (IS Relationship)

**Example**- IS Relationship

Class Policy {

}

Class TermPolicy extends Policy {

}

In this example, Policy is super class and TermPolicy is sub class

Where TermPolicy **IS A** Policy.

**Note-**

All the parent members are derived into child class but they are depending upon the below two condition as

-To check the access specifiers

-Members does not exist into sub class.

**UML Diagram-**

Parent -P

Child – C

Where p is parent class and c is the child class.

**Note**- Below are different names for super and sub class

Super Class->Parent Class->Base Class-> Old Class

Sub Class-> Child Class-> Derived Class -> New Class

**Note**-

1. Inherit the classes by using extends keywords.
2. Whenever we create the object of subclass then all the member will get called super class as well as sub class. Because reason is that super class members automatically inherited into sub class that’s why.
3. Why we use inheritance that is for code reusability, reusability means we can reuse existing class features such as variables and method, etc.
4. We cannot extend the final class.

**When to use?**

If we want to extends or increase of features of class then go for inheritance.

**Business requirement-**

**Why inheritance?**

Suppose we have one class which contain the fields like, firstname, lastname, address, city, mobile number and

In future we got the requirement to add the email then what option we have below-

1. Modify the attributes/fields/variable in existing class but this is not good option it will increase the testing for that class.
2. Add the attributes/fields/variable in the new class, this is the good option we can also reduce the testing efforts for this.

How the class will look like

Class Parent {

String firstname;

String lastname;

String address;

String city;

String mobilenumber;

}

Class Child extends Parent {

String email;

}

**Note**

We cannot assign parent class to child class- it means Child c=new Parent(); Here we cant write new Parent();

All the members of super class will be directly inherited into sub class and they are eligible and depends on access specifiers only.

**Dynamic dispatch-**

The process of assigning the child class reference to parent class called as “Dynamic dispatch.”

**Example-**

Class X {

}

Class Y extends X {

}

Class Test {

Public static void main(string args[]){

X x= new Y(); // Here we are assigning the child reference new Y() to parent class as X.

}

Inheritance Example-

**Scenario 1**

**package** com.inheritance;

**class** X {

**int** a = 10;

**int** b = 20;

**void** m1() {

System.***out***.println("Class X- m1() method");

}

**void** m2() {

System.***out***.println("Class X- m2() method");

}

}

**package** com.inheritance;

**class** Y **extends** X {

**int** b = 30;

**int** c = 40;

**void** m2() {

System.***out***.println("Class Y- m2() method");

}

**void** m3() {

System.***out***.println("Class Y- m3() method");

}

}

**package** com.inheritance;

**public** **class** TestMain {

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

//Scenario- 1

X x=**new** X();

System.***out***.println(x.a);

System.***out***.println(x.b);

System.out.println(x.c);

x.m1();

x.m2();

x.m3();

//Scenario-2

Y y = **new** Y();

System.***out***.println(y.a);

System.***out***.println(y.b);

System.***out***.println(y.c);

y.m1();

y.m2();

y.m3();

//Scenario-3

X x = **new** Y();

System.***out***.println(x.a);

System.***out***.println(x.b);

//System.out.println(x.c);

x.m1();

x.m2();

//x.m3();

//Scenario-4 (Note 3rd and 4th scenario are same)

X x = **new** X();

Y y = **new** Y();

x = y;

System.***out***.println(x.a);

System.***out***.println(x.b);

System.out.println(x.c);

x.m1();

x.m2();

x.m3();

//Scenario-5- Note- this is equivalent to 2nd scenario

X x = **new** Y();

Y y = **new** Y();

y = (Y) x;

System.***out***.println(y.a);

System.***out***.println(y.b);

System.***out***.println(y.c);

y.m1();

y.m2();

y.m3();

//Scenario-6

Y y= new X();

}

}