Inheritance-

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

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 depends upon the below

-To check the access specifiers

-Members does not exist into sub 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.
3. Why we use inheritance that is for code reusability, reusability means we can reuse existing class features such as variables and method, etc.

UML Diagram-

$$\begin{tikzcd}\text{Parent -P} \\ \text{Child – C} \arrow[u]\end{tikzcd}$$

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

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

Sub Class-> Child Class-> Derived Class -> New 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 PAN number then what option we have below-

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

How the class will looks like

```
Class Parent {

String firstname;
String lastname;
String address;
String city;
String mobilenumber;
}
Class Child extends Parent {

String pancard;

}
```

Rules-

We cannot assign parent class reference to child class-

We cannot extend the final class.

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 .

}

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");
     }
}
```

```java
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(); // Not Valid
       }
}
```

Note:

1) Whatever the parent has by default available to the child but whatever the child has by default not available to the parent. Hence on the child reference we can call both parent and child class methods. But on the parent reference we can call only methods available in the parent class and we can't call child specific methods.
2) Parent class reference can be used to hold child class object but by using that reference we can call only methods available in parent class and child specific methods we can't call.
3) Child class reference cannot be used to hold parent class object
4) For all java classes the most commonly required functionality is define inside object class hence object class acts as a root for all java classes.