*Inheritance*

*Inheritance*

* Whatever property having to parent class by default available to child class that means the process of creating the new class by using the existing class functionality called as Inheritance.
* It is called as - (IS-A Relationship)
* The main advantage of IS-A relationship is reusability
* By using "extends" keywords we can implement IS-A relationship.

Example IS-A relationship

class Parent {

}

class Child extends Parent {

}

*Point to be remember*

* 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.
* 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.
* Child class reference cannot be used to hold parent class object.
* Inherit the classes by using extends keywords.
* Why we use inheritance that is for code reusability, reusability means we can reuse existing class features such as variables and method, etc.
* We cannot extend the final class.

*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 look like

Class Parent {

String firstName;

String lastName;

String address;

String city;

String mobileNumber;

}

Class Child extends Parent {

String pancard;

}

**Note**

We cannot assign parent class reference to child 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 A {

}

Class B extends A {

}

Class Test {

Public static void main(string args[]){

A a= new B(); // Here we are assigning the child reference new B () to parent class .

}

**package** com.velocity;

**public** **class** A {

**int** p = 10;

**int** q = 20;

**void** m1() {

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

}

**void** m2() {

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

}

}

**package** com.velocity;

**public** **class** B **extends** A {// used the extends keyword

**int** q = 30;

**int** r = 40;

**void** m2() {

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

}

**void** m3() {

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

}

}

Scenario- 1 by creating the object of A class i.e parent class

**package** com.velocity;

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

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

A a=**new** A();

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

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

//System.out.println(a.r); not able to call child variable

a.m1();

a.m2();

// a.m3(); not able to call child method

}

}

Output

10

20

Class A- m1() method

Class A- m2() method

Scenario- 2 by creating the object of B class i.e child class

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

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

//Scenario-2

B b = **new** B(); // by creating child object

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

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

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

b.m1();

b.m2();

b.m3();

}

}

Output

10

30

40

Class A- m1() method

Class B- m2() method

Class B- m3() method

Scenario- 3 child class reference new B() to parent class

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

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

//Scenario-3

A a = **new** B(); // child reference new B() to parent class .

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

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

//System.out.println(a.r);

a.m1();

a.m2();

//a.m3();

}

}

Output

10

20

Class A- m1() method

Class B- m2() method

Scenario- 4 (3rd and 4th scenario is same)

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

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

A a = **new** A();

B b = **new** B();

a = b;

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

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

//System.out.println(a.r);

a.m1();

a.m2();

//a.m3();

}

}

Output

10

20

Class A- m1() method

Class B- m2() method

Scenario- 5 scenario are same 2nd)

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

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

A a = **new** B();

B b = **new** B();

b = (B) a;

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

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

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

b.m1();

b.m2();

b.m3();

}

}

Output

10

30

40

Class A- m1() method

Class B- m2() method

Class B- m3() method