INHERITANCE IN JAVA

Velocity

AUGUST 2, 2022

VELOCITY

Pune

Inheritance in java

Inheritance

- The process of creating the new class by using the existing class functionality is called as Inheritance.
- Inheritance means simply reusability.
- It is called as - (IS-A Relationship)

Syntax:

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

Inheritance uses the “extends” keyword to create a derived class by reusing the base class code.

Extends keyword in Java

The extended keyword extends a class and is an indicator that a class is being inherited by another class. When you say class B extends a class A, it means that class B is inheriting the properties (methods, attributes) from class A. Here, class A is the superclass or parent class and class B is the subclass or child class.

Example:

```
class Policy
{
 // policy data variables
 // policy member functions
}
class TermPolicy extends superclass
{
 // term policy data variables
 // term policy member functions
}
```

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

Where TermPolicy IS A Policy.

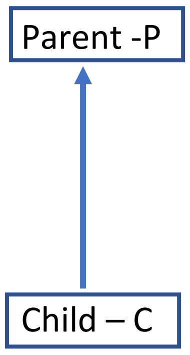
Note-

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.

UML Diagram-



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

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.
4. We cannot extend the final class.

When to use?

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

Business requirement-

Why inheritance?

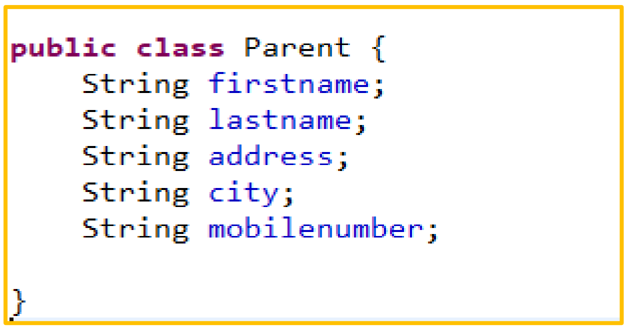
Suppose we have one class which contain the fields like, firstname, lastname, address, city, mobile number and

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

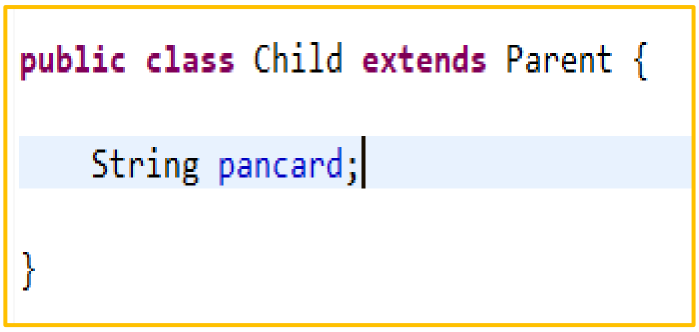
1. Modify the attributes/fields in existing class but this is not a good option as it will increase the testing for that class.

2. Or Add the attributes in the new class, this is the good option because we can also reduce the testing efforts for this.

How the class will look like



```
public class Parent {
    String firstname;
    String lastname;
    String address;
    String city;
    String mobilenumber;

}
```



```
public 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 their eligibility 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 {

}

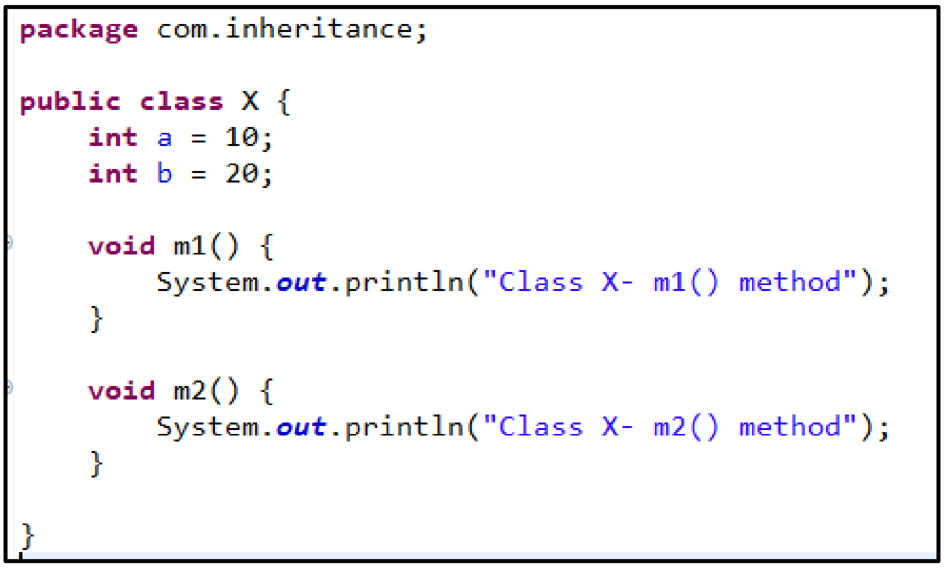
Class Test {

Public static void main (string args[]){

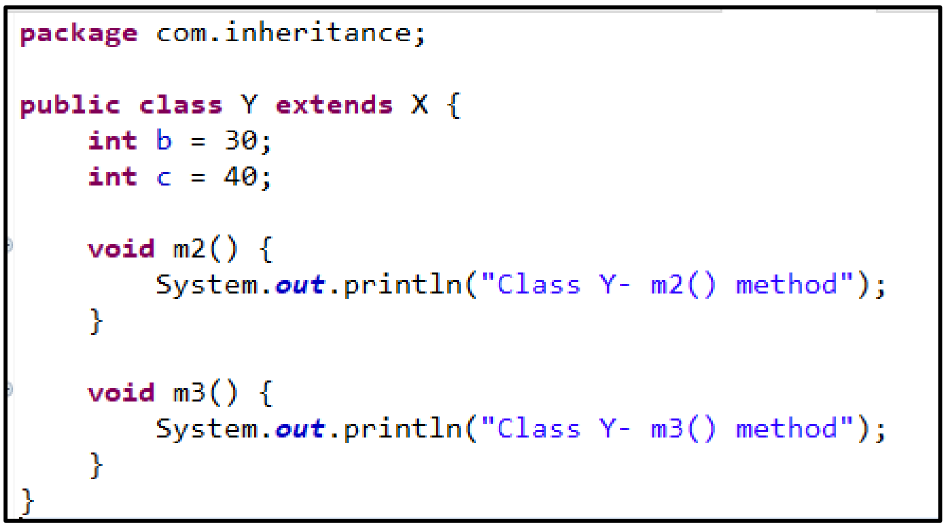
X x= new Y(); // Here we are assigning the child reference **new Y()** to parent class .

}

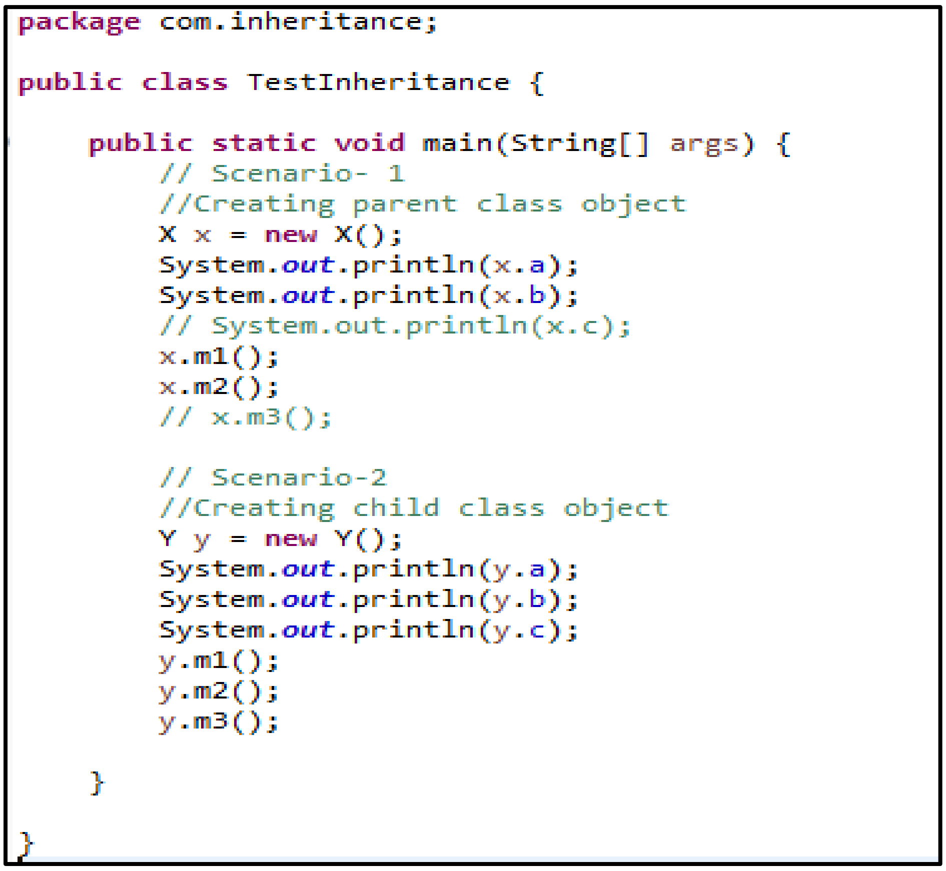
Inheritance Example



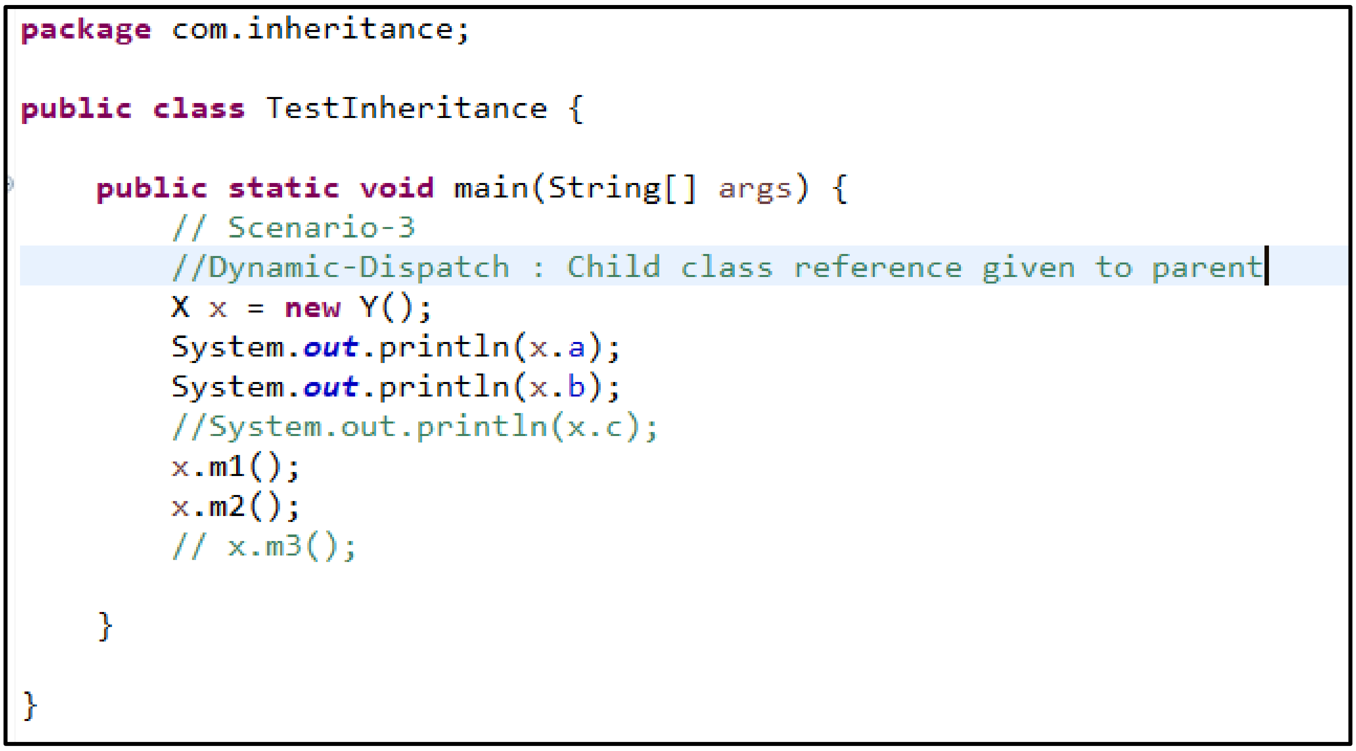
```java
package com.inheritance;

public 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");
    }

}
```



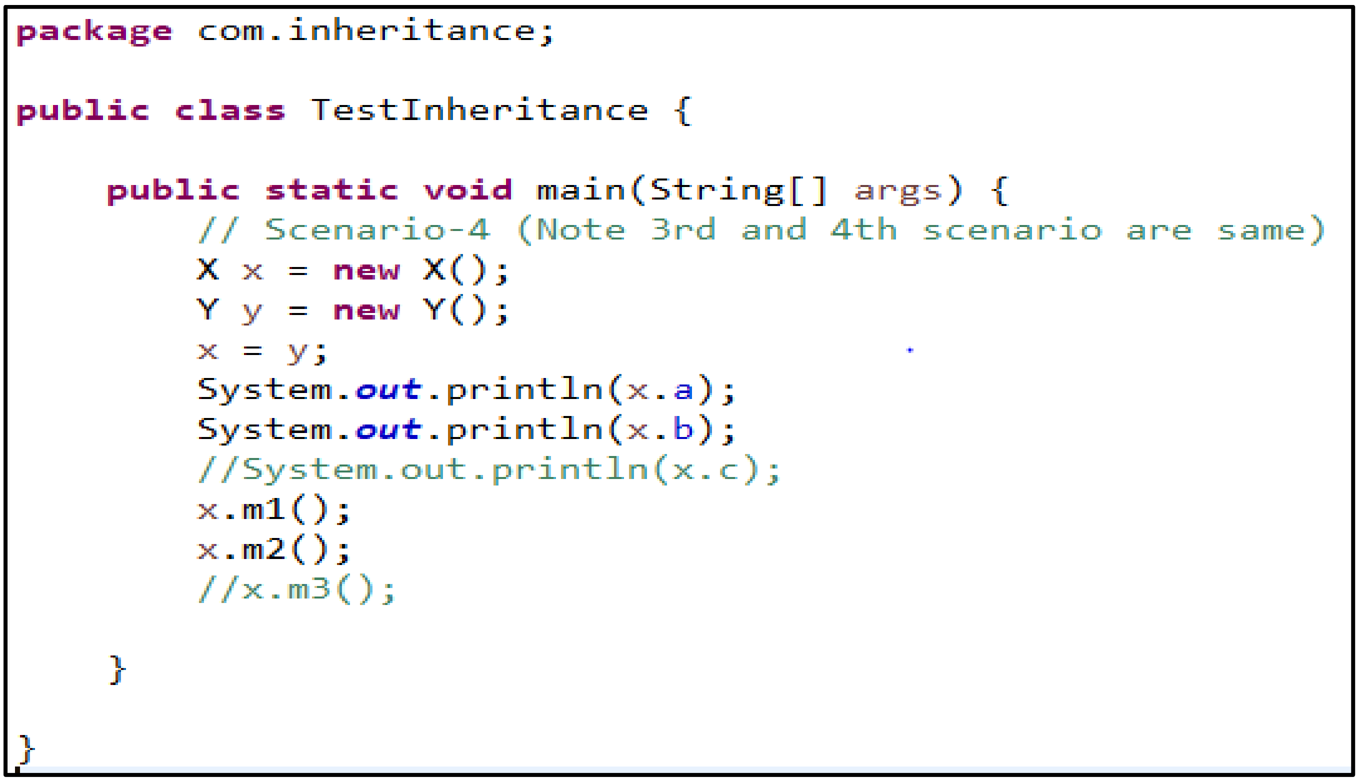
```java
package com.inheritance;

public 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");
    }
}
```



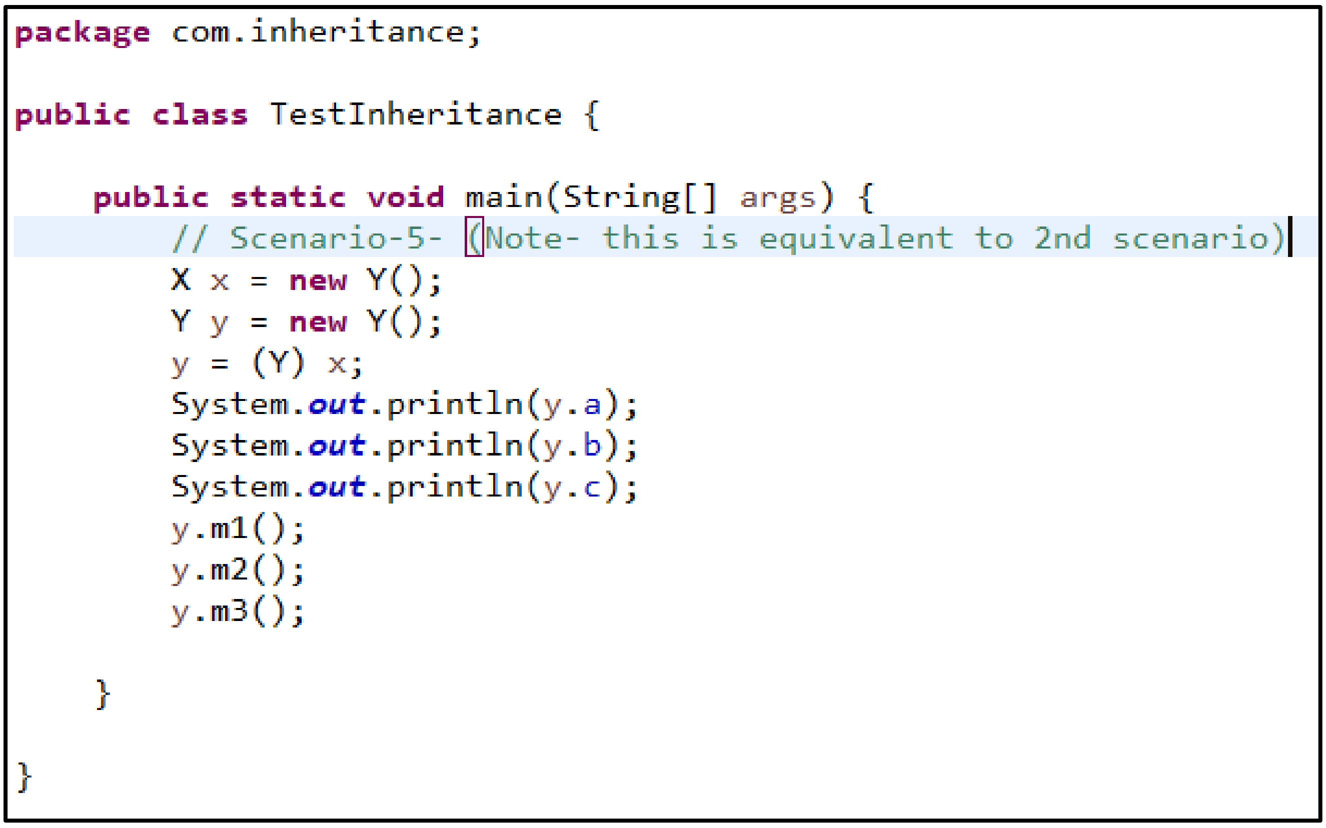
```java
package com.inheritance;

public class TestInheritance {

    public static void main(String[] args) {
        // Scenario- 1
        //Creating parent class object
        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
        //Creating child class object
        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();

    }

}
```



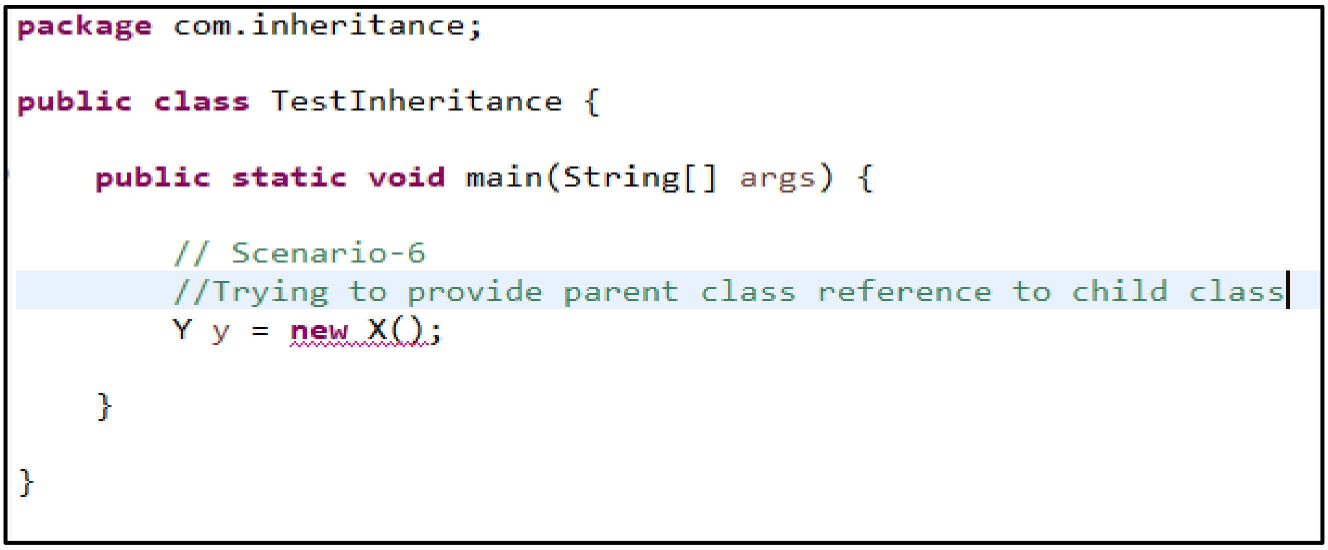
```java
package com.inheritance;

public class TestInheritance {

    public static void main(String[] args) {
        // Scenario-3
        //Dynamic-Dispatch : Child class reference given to parent
        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();

    }

}
```



```java
package com.inheritance;

public class TestInheritance {

    public static void main(String[] args) {
        // 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();

    }

}
```



```java
package com.inheritance;

public class TestInheritance {

    public static void main(String[] args) {
        // 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();

    }

}
```



```java
package com.inheritance;

public class TestInheritance {

    public static void main(String[] args) {

        // Scenario-6
        //Trying to provide parent class reference to child class
        Y y = new X();

    }

}
```