PROGRAMACIÓN ORIENTADA A OBJETOS (parte 2)

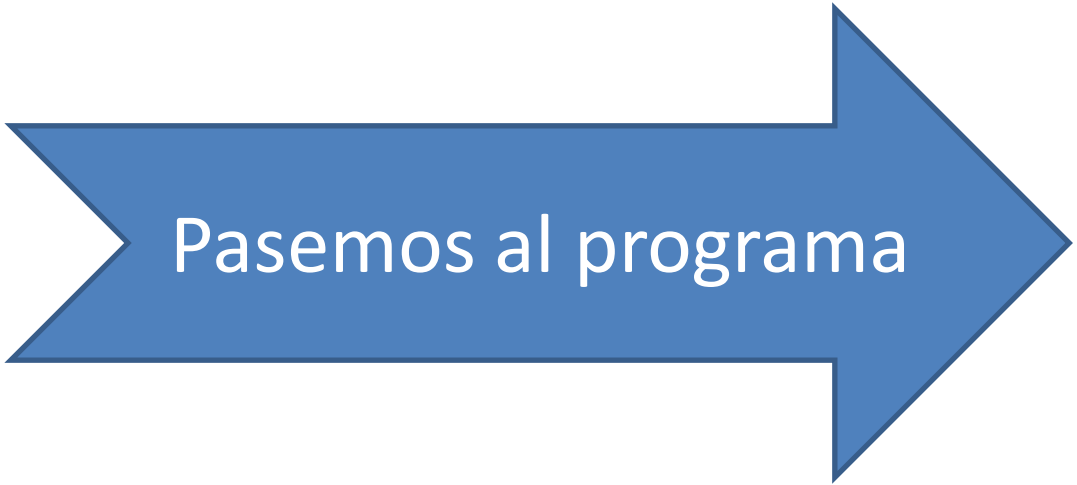
Elaborado por: Juan Miguel Guanira Erazo PUCP

OPERADORES SOBRECARGADOS

Al igual que en la primera parte del curso, se puede implementar sobrecargas de operadores relacionadas a las clases.

La diferencia está en la forma de implementarlas, en esta caso la sobrecarga constituirá un método de la clase.

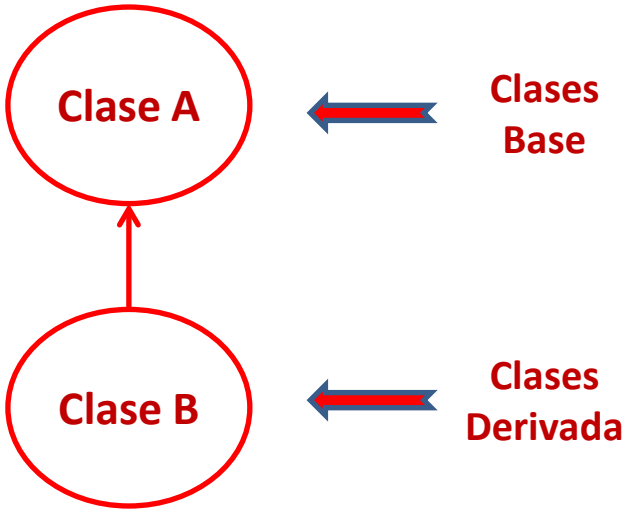
Esto es al escribir `a + b`, el compilador lo interpretará como `a.+(b)`, donde `a` es un objeto de la clase para la que se sobrecargó el operador `+`.


Pasemos al programa

HERENCIA

Como comentamos al inicio del capítulo, Herencia es una propiedad de la Programación orientada a objetos por medio de la cual se puede crear una clase a partir de otra.

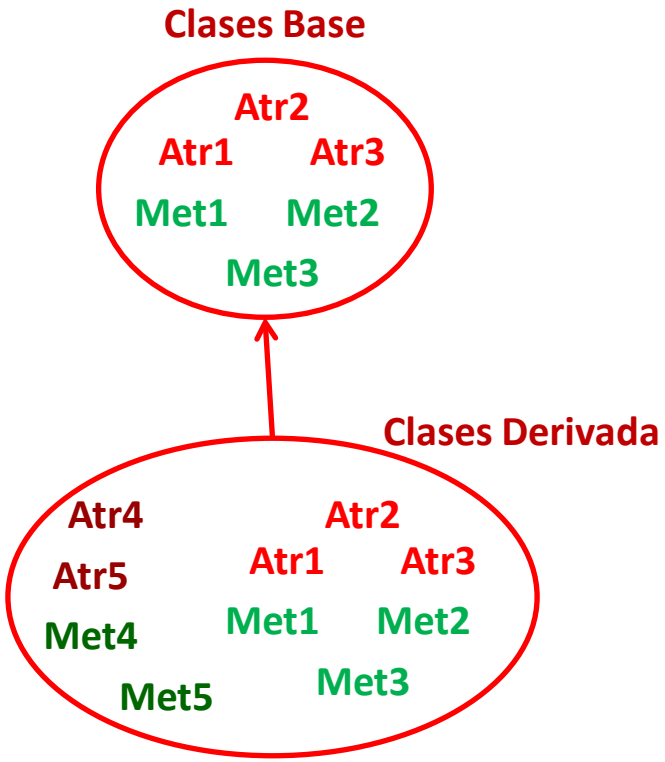
La figura siguiente indica que la clase B está heredando de la clase A.



Características:

Al heredar de una clase base, la clase derivada adquiere todos los atributos y métodos de la clase base.

A la clase derivada se le puede agregar los atributos y métodos que se deseen.

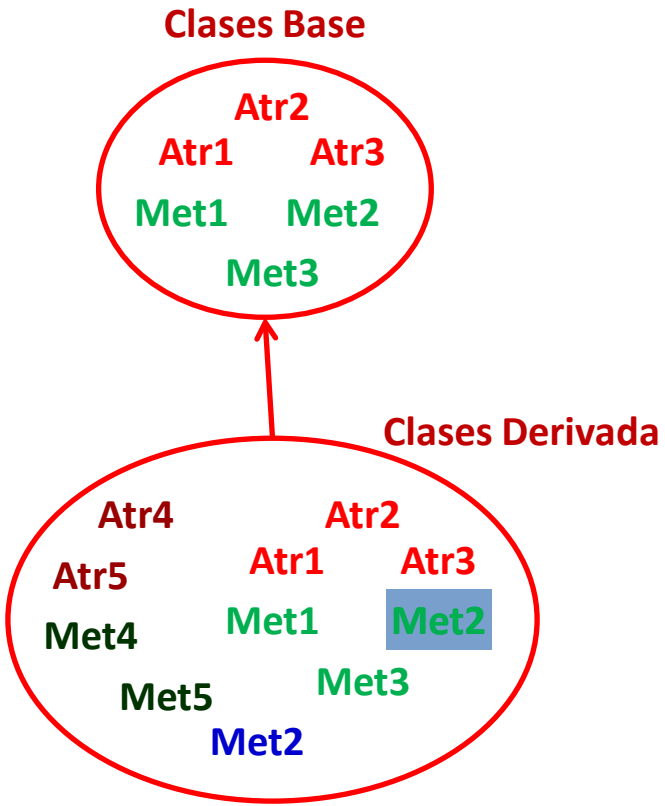


Características:

Inclusive, la clase derivada puede “ocultar” o “sobrescribir” algún método.

Esto no implica destruir el método original, sino ocultarlo.

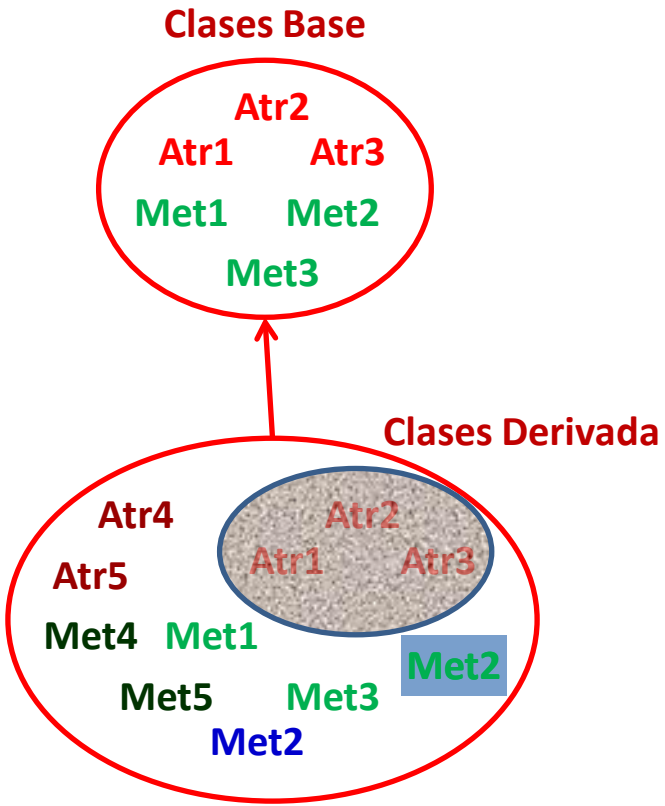
Por defecto se ejecutará el método nuevo.



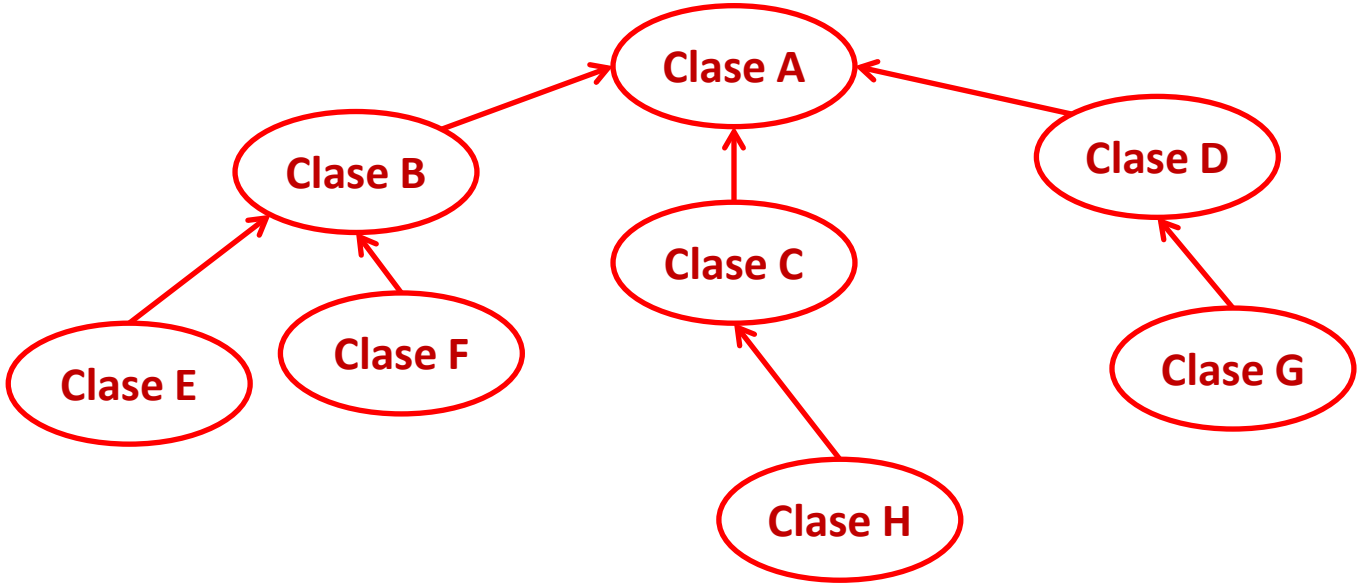
Características:

Desde la clase derivada no se podrá acceder a los elemento que estén en la zona privada de la clase base.

Solo se tendrá acceso por los métodos.

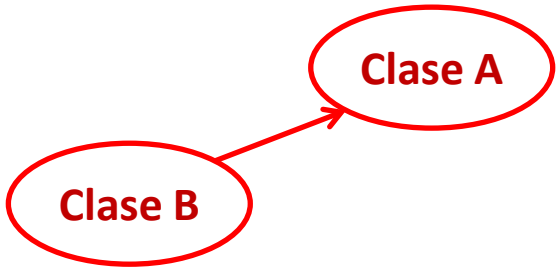


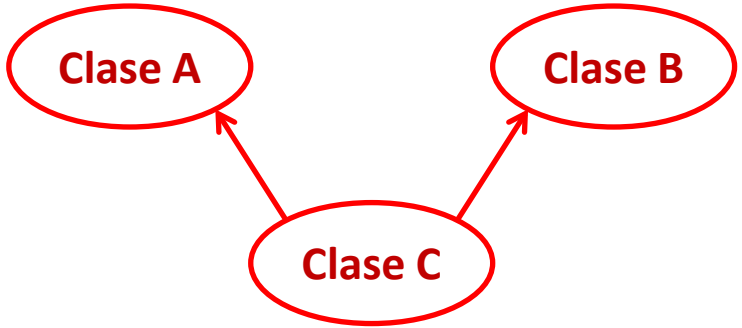
Una clase puede heredar de otra clase y a su vez otra clase puede heredar de la derivada, formándose un "Árbol o Jerarquía de clases".



TIPOS DE HERENCIA:

Cuando una clase hereda de una única clase se dice que la herencia es simple.





Cuando una clase hereda de dos o más clases se dice que la herencia es múltiple.

EJEMPLO DE HERENCIA

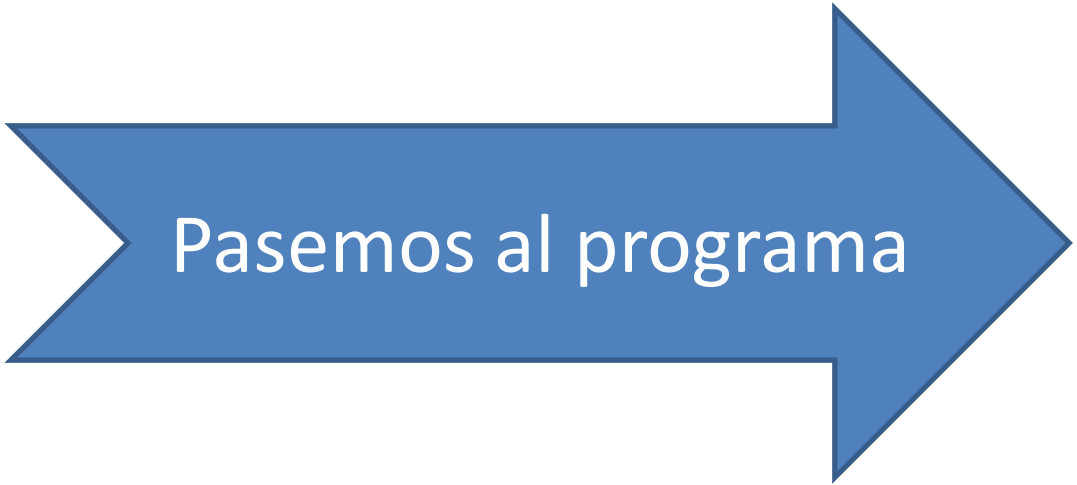
Al igual que con los conceptos básicos, plantearemos un problema simple de herencia.

Primero pensemos en un objetos círculo.

Definido por su radio.

Del cual podemos necesitar obtener su área.

Y su circunferencia.


Pasemos al programa

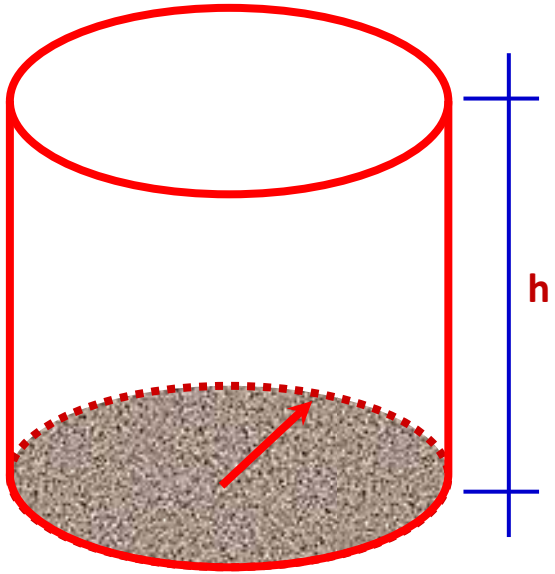
Ahora pensemos en un objeto cilindró.

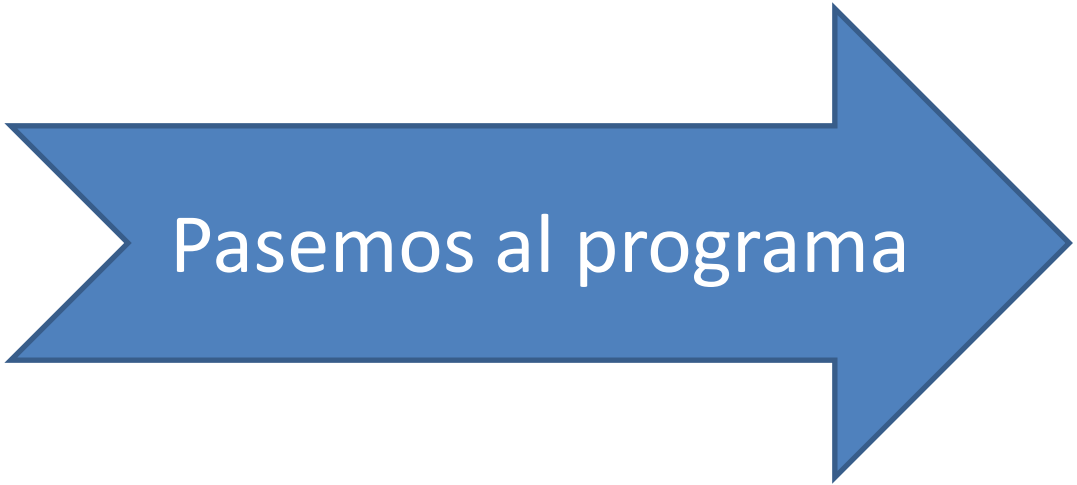
Reflexionemos, ¿por qué crearlo de cero? Si su base es circular.

Podemos crearlo a partir de la clase círculo.

Solo tendríamos que agregarle la altura.

Y calcular su volumen y el área de su superficie a partir de la clase círculo.

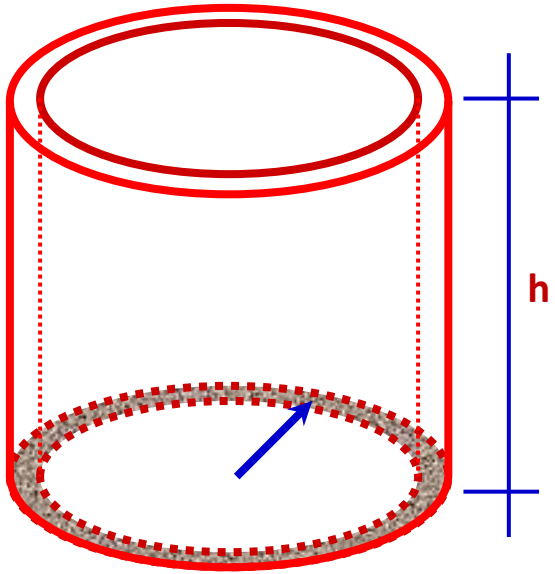


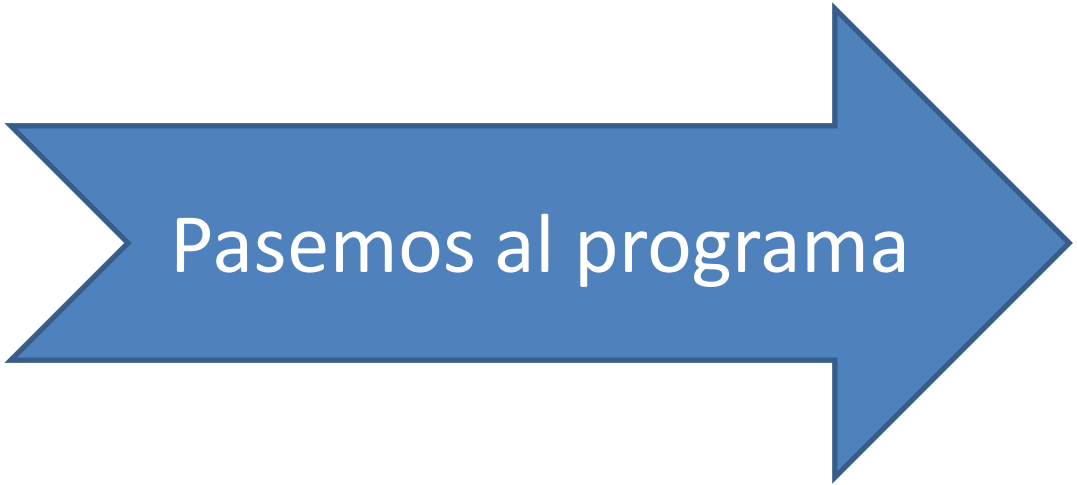

Pasemos al programa

Ahora pensemos en un objeto tubería.

De igual manera lo podemos diseñar a partir de la clase cilindro.

Sólo tendríamos que agregar el radio de la zona hueca.




Pasemos al programa

ZONA “PROTEGIDA” EN LA HERENCIA

Al igual que en una clase existe la zona pública (**public**) y la zona privada (**private**), existe también una tercera zona que solo tiene efecto en herencia.

Esta zona se denomina “protegida” (**protected**).

En esta zona se pueden colocar tanto atributos como métodos.

Características:

Los objetos no pueden acceder a la zona protegida, para ellos es una zona privada.

Una clase derivada si puede acceder a la zona protegida de su clase base.

Se utiliza como una puerta falsa para acceder directamente a los atributos de la clase base.

Implementación:

```
class Base{
    private:
        ...
    public:
        ...
    protected:
        ...
};
```

```
class Derivada:
        public Base{
    private:
        ...
    public:
        Metodo()
        }
```

Puede acceder

ESPECIFICADORES DE ACCESO EN LA HERENCIA

El uso de especificadores de acceso en herencia se emplea para restringir el acceso a los elementos de una clase base las clases que hereden de las clases derivadas de la base.

El especificador de acceso en herencia es el siguiente elemento:

```
class derivada : public base{ …
```

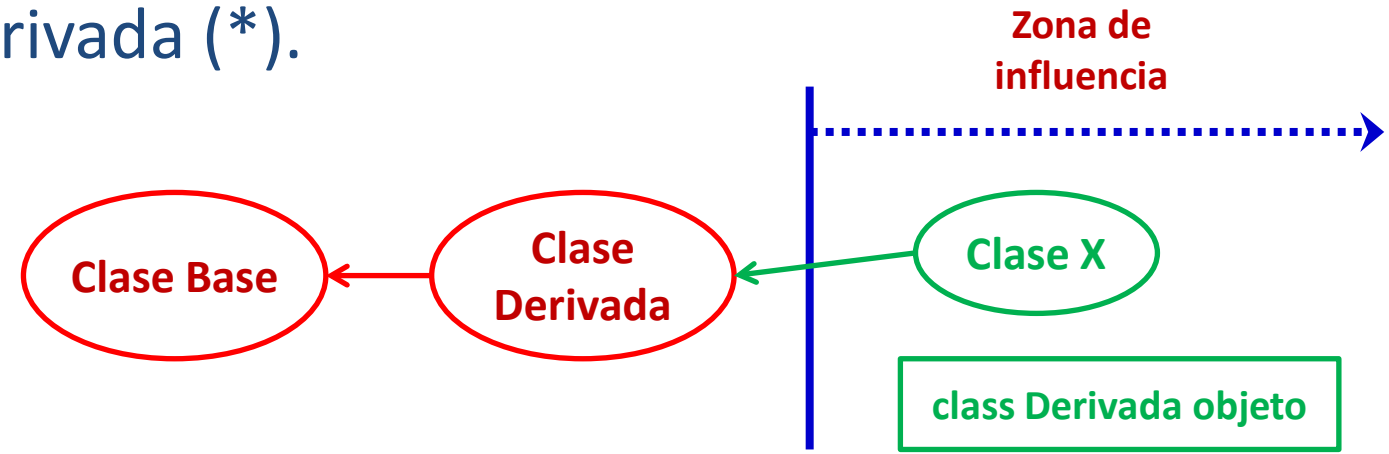
También:

```
class derivada : protected base{ …
class derivada : private base{ …
```

Ese especificador de acceso no afecta ni a la clase base ni a la clase que deriva (*) de ella.

Solo afecta a aquellas clases que derivan a partir de la case derivada (*) y a los objetos que instancian la clase derivada (*).

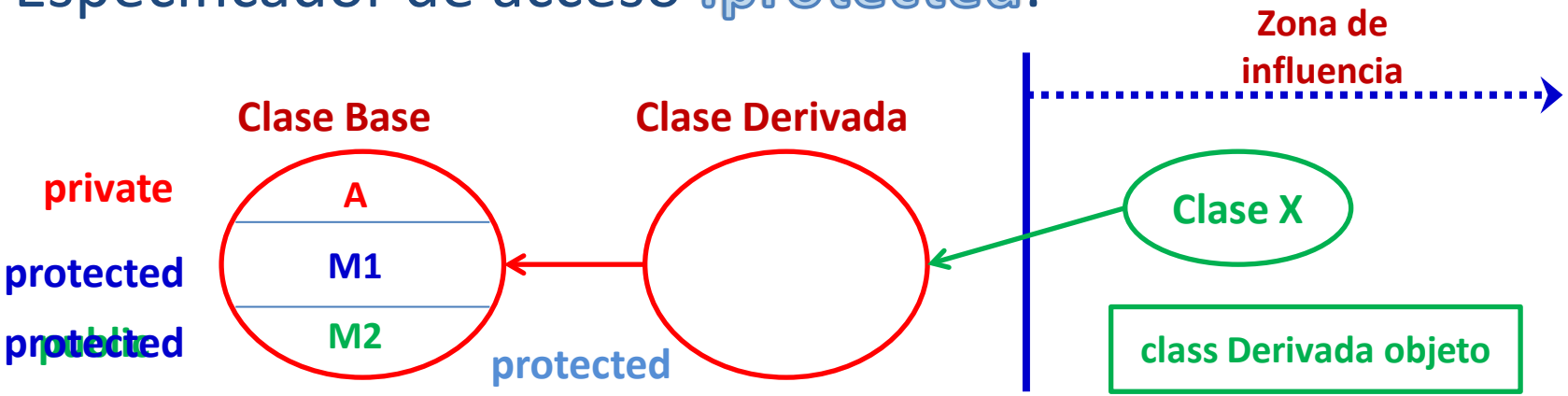
Esto es



Especificador de acceso :public.

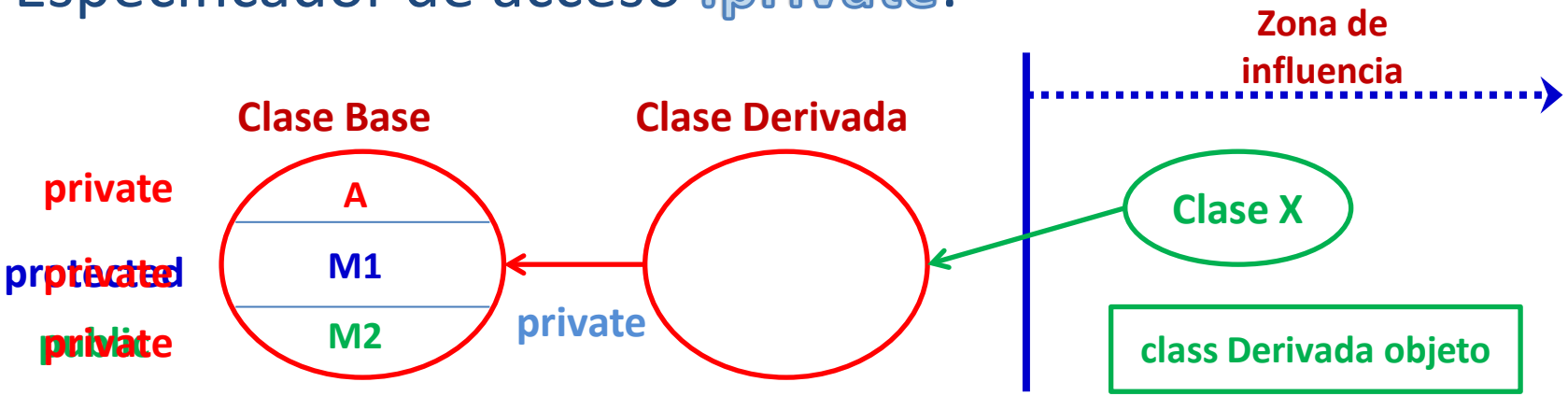
	Clase Derivada	Clase X	class Derivada objeto
A	✗	✗	✗
M1	✓	✓	✗
M2	✓	✓	✓

Especificador de acceso :protected.



	Clase Derivada	Clase X	class Derivada objeto
A	✗	✗	✗
M1	✓	✓	✗
M2	✓	✓	✗

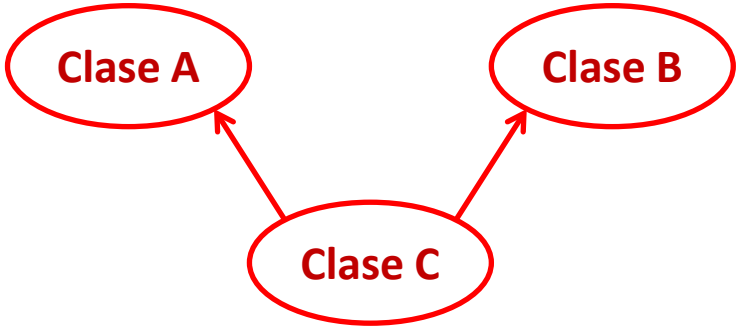
Especificador de acceso :private.



	Clase Derivada	Clase X	class Derivada objeto
A	✗	✗	✗
M1	✓	✗	✗
M2	✓	✗	✗

HERENCIA MÚLTIPLE

Como explicamos al principio, cuando una clase hereda de dos o más clases se dice que la herencia es múltiple.

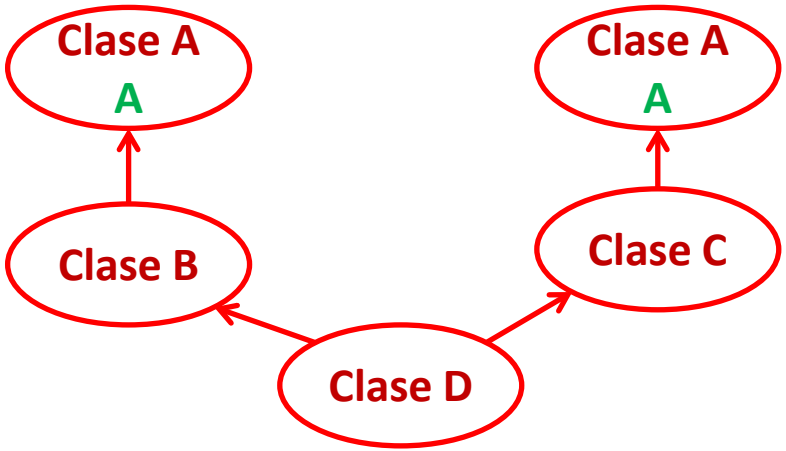


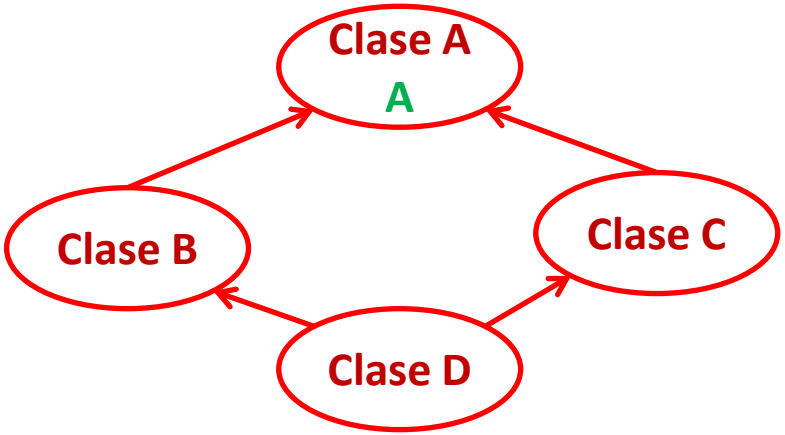
Declaración:

```
class ClaseC : public ClaseA, public ClaseB{ …
```

Problemas con la herencia múltiple:

Los problemas con la herencia múltiple aparecen cuando se forman jerarquías de clases.





class ClaseB : virtual public ClaseA{ …