Herencia en C++
(Parte 2)
Modificadores de Acceso

Definición del concepto

Los “modificadores de acceso” o también llamados “especificadores de acceso” permiten implementar una de las características mas importantes de la POO conocida como ***principio de ocultación***, con la cual se asigna la visibilidad de los miembros de una clase, logrando seguridad e integridad de los datos en una aplicación.

En C++, y en la gran mayoría de los lenguajes OO se usan los siguientes modificadores de acceso y su símbolo UML en un diagrama de clases :

- **public**: **+**
- **protected**: **#**
- **private**: **-**

Tipos de modificadores

Cabe resaltar que los modificadores de acceso se pueden aplicar a clases simples o clases que estén involucradas en una relación de Herencia.

Asimismo, por defecto, todo atributo, método o constructor de una clase tiene visibilidad **private**.

La siguiente tabla recoge todas las posibles combinaciones de los accesos a las secciones de una clase:

Especificador de acceso	Desde la propia clase	Desde las clases derivadas	Desde el exterior
public	Si	Si	Si
protected	Si	Si	No
private	Si	No	No

Fuente: Herencia en C++

http://informatica.utem.cl/~mcast/ESDATOS/POO/herencia.pdf

Sintaxis

Para implementar los modificadores se sigue la siguiente sintaxis:

```
class NombreClase
{
   public:// Esto es accesible dentro de la clase y por fuera de ella
   //Declaración de atributos, constructores y métodos

   protected:// Esto es accesible solo por las clases derivadas
   //Declaración de atributos, constructores y métodos

   private:// Esto es accesible solo dentro de la clase
   //Declaración de atributos, constructores y métodos
};
```

Ejemplo (1)

Vamos implementar Herencia con una jerarquía de clases de 3 niveles, cada clase declara atributos **public**, **protected** y **private**, y se comprobara el acceso a los miembros de cada clase.

Ejemplo (2)

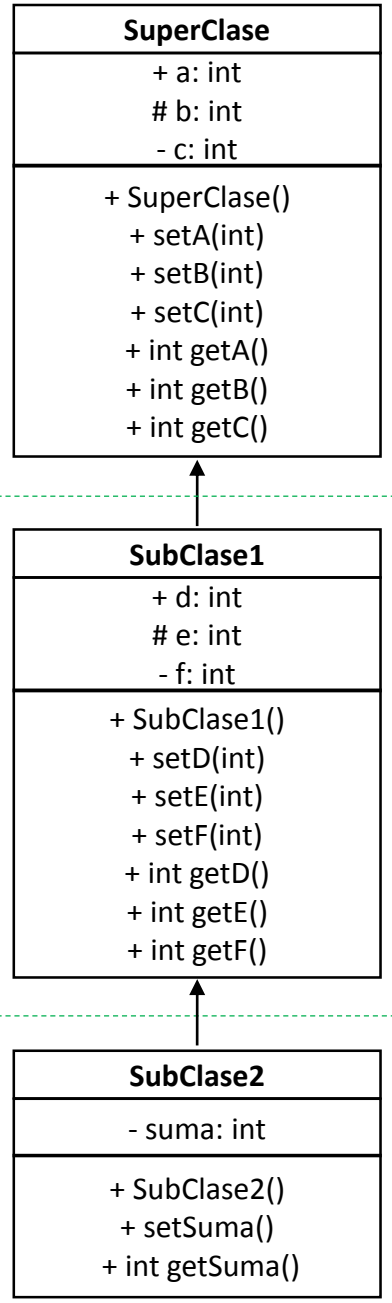
Diagrama de Clases

Jerarquía de Clases de 3 Niveles

Nivel 1

Nivel 2

Nivel 3



Ejemplo (3)

Declaración de las Clases (.h)

SuperClase

```
class SuperClase
{
    public:
        int a;

        SuperClase();
        virtual ~SuperClase();
        void setA(int);
        void setB(int);
        void setC(int);
        int getA();
        int getB();
        int getC();

    protected:
        int b;

    private:
        int c;
};
```

SubClases

```
class SubClase1: public SuperClase
{
    public:
        int d;

        SubClase1();
        virtual ~SubClase1();
        void setD(int);
        void setE(int);
        void setF(int);
        int getD();
        int getE();
        int getF();

    protected:
        int e;

    private:
        int f;
};
```

```
class SubClase2: public SubClase1
{
    public:
        SubClase2();
        virtual ~SubClase2();
        void setSuma();
        int getSuma();

    protected:

    private:
        int suma;
};
```

Ejemplo completo: ModificadoresAcceso.zip