Visibilidad de los miembros de una clase

Se pueden establecer distintos niveles de encapsulación
para los miembros de una clase (atributos y operaciones)
en función de desde dónde queremos que se pueda acceder a ellos:

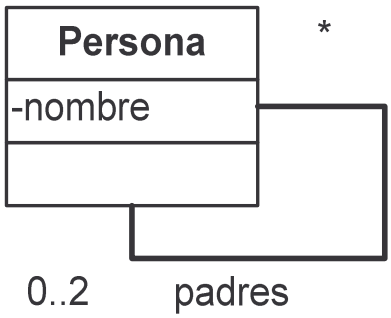
Visibilidad	Significado	Java	UML
Pública	Se puede acceder al miembro de la clase desde cualquier lugar.	`public`	`+`
Protegida	Sólo se puede acceder al miembro de la clase desde la propia clase o desde una clase que herede de ella.	`protected`	`#`
Privada	Sólo se puede acceder al miembro de la clase desde la propia clase.	`private`	`-`

Para encapsular por completo el estado de un objeto,
todos sus atributos se declaran como variables de instancia privadas
(usando el modificador de acceso `private`).

A un objeto siempre se accede a través de sus métodos públicos
(su **interfaz**).

Para usar el objeto no es necesario conocer
qué algoritmos utilizan sus métodos
ni qué tipos de datos se emplean para mantener su estado
(su **implementación**).

Diseño incorrecto




```
public class Persona
{
    public String nombre;
    public Persona padre;
    public Persona madre;
    public ArrayList hijos = new ArrayList();
}
```

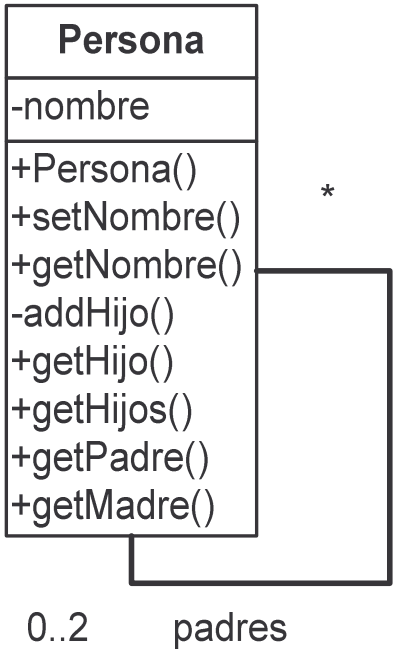
Uso correcto de la clase:

```
Persona juan = new Persona();
Persona carlos = new Persona();
Persona silvia = new Persona();

juan.nombre = “Juan”;
carlos.nombre = “Carlos”;
silvia.nombre = “Silvia”;

juan.padre = carlos;
juan.madre = silvia;
carlos.hijos.add(juan);
silvia.hijos.add(juan);
```

Uso incorrecto de la clase
(pese a ser válido tal como está implementada):

```
juan.padre = carlos;
juan.madre = carlos;
silvia.hijos.add(juan);
juan.hijos.add(juan);
```

Diseño correcto




```
import java.util.ArrayList;

public class Persona
{
  // Variables de instancia privadas
  private String nombre;
  private Persona padre;
  private Persona madre;
  private ArrayList hijos = new ArrayList();

  // Constructores públicos
  public Persona (String nombre)
  {
    this.nombre = nombre;
  }

  public Persona
        (String nombre, Persona padre, Persona madre)
  {
    this.nombre = nombre;
    this.padre = padre;
    this.madre = madre;

    padre.addHijo(this);
    madre.addHijo(this);
  }
```

```
  // Método privado

  private void addHijo (Persona hijo)
  {
    hijos.add(hijo);
  }

  // Métodos públicos
  // p.ej. Acceso a las variables de instancia

  public void setNombre (String nombre)
  {
    this.nombre = nombre;
  }

  public String getNombre ()
  {
    return nombre;
  }
  …
}
```

Con esta implementación, desde el exterior de la clase **no** se pueden modificar las relaciones existentes entre padres e hijos, por lo que estas siempre se mantendrán correctamente si implementamos bien la clase `Persona`.

Ejemplo

```
Persona carlos = new Persona("Carlos");
Persona silvia = new Persona("Silvia");
Persona juan = new Persona("Juan",carlos,silvia);
```

Operación permitida (a través de un método público):

```
juan.setNombre("Antonio");  // Cambio de nombre
```

Operación no permitida (error de compilación):

```
addHijo(Persona) has private access in Persona
juan.addHijo(carlos);
    ^
```