APRENDERAPROGRAMAR.COM

# TIPOS DE MODIFICADORES DE ACCESO EN JAVA: CONCEPTOS PUBLIC, PRIVATE Y PROTECTED O PROTEGIDO. VISIBILIDAD. (CU00693B)

Sección: Cursos
Categoría: Curso “Aprender programación Java desde cero”

Fecha revisión: 2029

**Resumen:** Entrega nº93 curso Aprender programación Java desde cero.

Autor: Alex Rodríguez

## MODIFICADORES DE ACCESO JAVA: PUBLIC, PRIVATE, PROTECTED.

Hasta ahora habíamos dicho que una subclase no tiene acceso a los campos de una superclase de acuerdo con el principio de **ocultación de la información.** Sin embargo, esto podría considerarse como demasiado restrictivo.


Decimos que podría considerarse demasiado restrictivo porque limita el acceso a una subclase como si se tratara de una clase cualquiera, cuando en realidad la relación de una superclase con una subclase es más estrecha que con una clase externa. Por ello en diferentes lenguajes, Java entre ellos, se usa un nivel de acceso intermedio que no es ni *public* ni *private*, sino algo intermedio que se denomina como "acceso protegido", expresado con la palabra clave **protected**, que significa que las subclases sí pueden tener acceso al campo o método.

El modificador de acceso protected puede aplicarse a todos los miembros de una clase, es decir, tanto a campos como a métodos o constructores. En el caso de métodos o constructores protegidos, estos serán visibles/utilizables por las subclases y otras clases del mismo package. El acceso protegido suele aplicarse a métodos o constructores, pero preferiblemente no a campos, para evitar debilitar el encapsulamiento. En ocasiones puntuales sí resulta de interés declarar campos con acceso protegido.

La sintaxis para emplear esta palabra clave es análoga a la que usamos con las palabras *public* y *private*, con la salvedad de que *protected* suele usarse cuando se trabaja con herencia. Desde un objeto de una subclase podremos acceder o invocar un campo o método declarado como *protected*, pero no podemos acceder o invocar a campos o métodos privados de una superclase. Declara un campo de una clase como *protected* y en un test crea un objeto de la subclase y trata de acceder a ese campo con una invocación directa del tipo interino43.IdProfesor = "54-DY-87".

Java admite una variante más en cuanto a modificadores de acceso: la omisión del mismo (no declarar ninguno de los modificadores *public*, *private* o *protected*). En la siguiente tabla puedes comparar los efectos de usar uno u otro tipo de declaración en cuanto a visibilidad de los campos o métodos:

| MODIFICADOR | CLASE | PACKAGE | SUBCLASE | TODOS |
|---|---|---|---|---|
| **public** | Sí | Sí | Sí | Sí |
| **protected** | Sí | Sí | Sí | No |
| **No especificado** | Sí | Sí | No | No |
| **private** | Sí | No | No | No |

**EJERCICIO**

Considera que estás desarrollando un programa Java donde trabajas con la superclase Profesor y la subclase ProfesorEmerito. Crea el código para estas clases que cumpla los requisitos que indicamos.

Como atributos de la superclase tendremos nombre (String), edad (int) y añosConsolidados (int) declarados como **protected**.

En la subclase se trabajará con el campo adicional añosEmerito declarado como private.

Un método de la subclase será double obtenerSalarioBase () que obtendrá el salario base como (925 + añosConsolidados * 33.25 + 47.80 * añosEmerito).

Intenta acceder directamente al campo añosConsolidados desde la subclase (como si fuera un campo más de la subclase) para implementar este método. ¿Es posible sin utilizar una invocación a super ni un método get? ¿Qué ocurre si el atributo en la superclase lo declaras private?

Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU00694B

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:

http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=68&Itemid=188