

5.C. Métodos.

1. Métodos.

1.3. Modificadores en la declaración de un método.

En la declaración de un método también pueden aparecer modificadores (como en la declaración de la clase o de los atributos). Un método puede tener los siguientes tipos de modificadores:

- **Modificadores de acceso.** Son los mismos que en el caso de los atributos (por omisión o de paquete, `public`, `private` y `protected`) y tienen el mismo cometido (acceso al método sólo por parte de clases del mismo paquete, o por cualquier parte del programa, o sólo para la propia clase, o también para las subclases).
- **Modificadores de contenido.** Son también los mismos que en el caso de los atributos (`static` y `final`) junto con, aunque su significado no es el mismo.
- **Otros modificadores** (no son aplicables a los atributos, sólo a los métodos): `abstract`, `native`, `synchronized`.

Un método `static` es un método cuya implementación es igual para todos los objetos de la clase y sólo tendrá acceso a los atributos estáticos de la clase (dado que se trata de un método de clase y no de objeto, sólo podrá acceder a la información de clase y no la de un objeto en particular). Este tipo de métodos pueden ser llamados sin necesidad de tener un objeto de la clase instanciado.

En Java un ejemplo típico de métodos estáticos se encuentra en la clase `Math`, cuyos métodos son todos estáticos (`Math.abs`, `Math.sin`, `Math.cos`, etc.). Como habrás podido comprobar en este ejemplo, la llamada a métodos estáticos se hace normalmente usando el nombre de la propia clase y no el de una instancia (objeto), pues se trata realmente de un método de clase. En cualquier caso, los objetos también admiten la invocación de los métodos estáticos de su clase y funcionaría correctamente.

Un método `final` es un método que no permite ser sobrescrito por las clases descendientes de la clase a la que pertenece el método. Volverás a ver este modificador cuando estudies en detalle la herencia.

El modificador `native` es utilizado para señalar que un método ha sido implementado en código nativo (en un lenguaje que ha sido compilado a lenguaje máquina, como por ejemplo `C` o `C++`). En estos casos simplemente se indica la cabecera del método, pues no tiene cuerpo escrito en Java.

Un método `abstract` (**método abstracto**) es un método que no tiene implementación (el cuerpo está vacío). La implementación será realizada en las clases descendientes. Un método sólo puede ser declarado como `abstract` si se encuentra dentro de una clase `abstract`. También volverás a este modificador en unidades posteriores cuando trabajes con la herencia.

Por último, si un método ha sido declarado como `synchronized`, el entorno de ejecución obligará a que cuando un proceso esté ejecutando ese método, el resto de procesos que tengan que llamar a ese mismo método deberán esperar a que el otro proceso termine. Puede resultar útil si sabes que un determinado método va a poder ser llamado concurrentemente por varios procesos a la vez. Por ahora no lo vas a necesitar.

Dada la cantidad de modificadores que has visto hasta el momento y su posible aplicación en la declaración de clases, atributos o métodos, veamos un resumen de todos los que has visto y en qué casos pueden aplicarse:

Cuadro de aplicabilidad de los modificadores.			
	Clase	Atributo	Método
Sin modificador (paquete)	Sí	Sí	Sí
<code>public</code>	Sí	Sí	Sí
<code>private</code>		Sí	Sí
<code>protected</code>	Sí	Sí	Sí
<code>static</code>		Sí	Sí

Cuadro de aplicabilidad de los modificadores.			
	Clase	Atributo	Método
final	Sí	Sí	Sí
synchronized			Sí
native			Sí
abstract	Sí		Sí