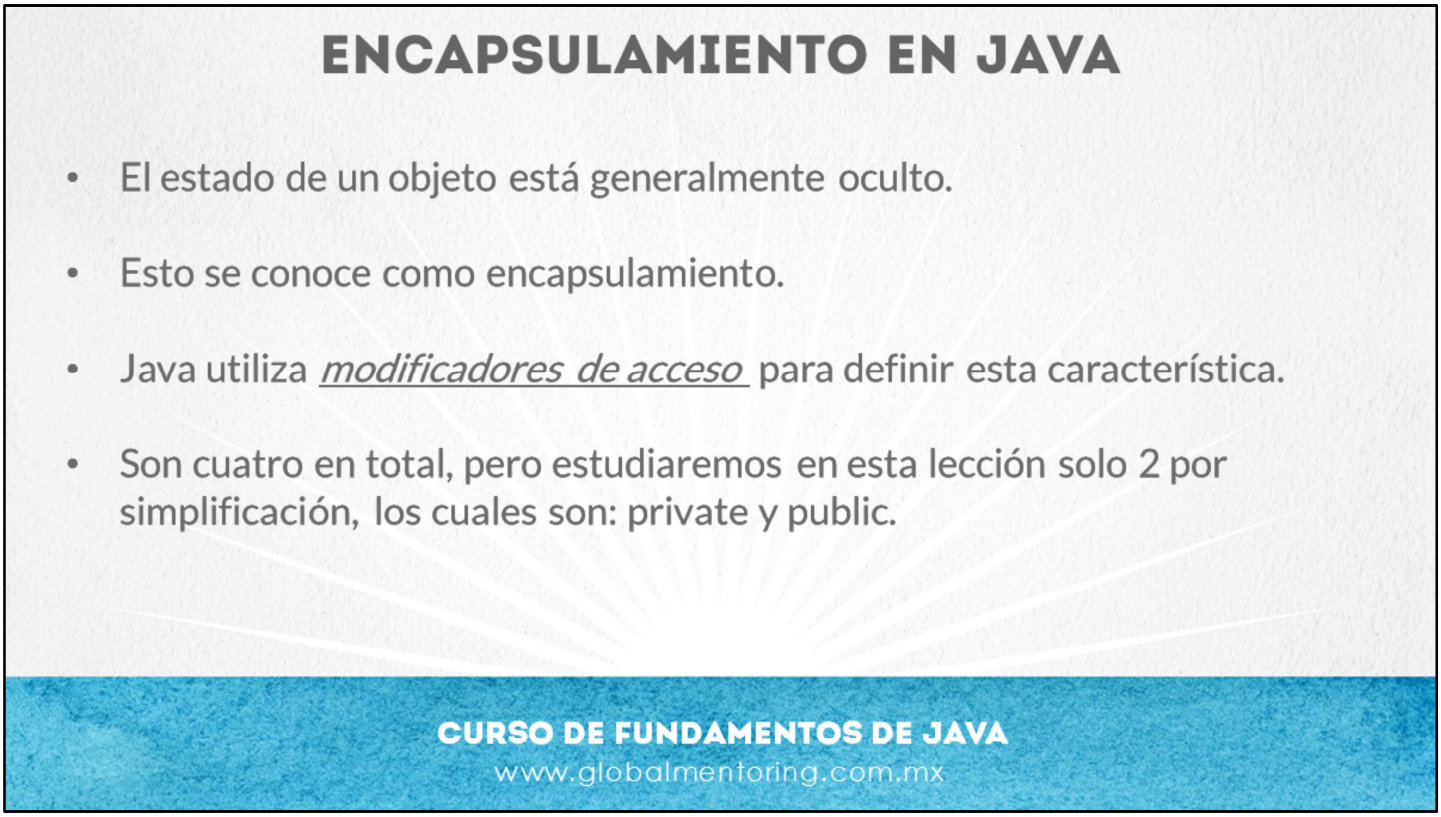
Hola, te saluda Ubaldo Acosta. Bienvenida o bienvenido nuevamente. Espero que estés listo para comenzar con esta lección.

Vamos a estudiar el tema de Encapsulamiento en Java.

¿Estás listo? Ok, ¡Vamos!

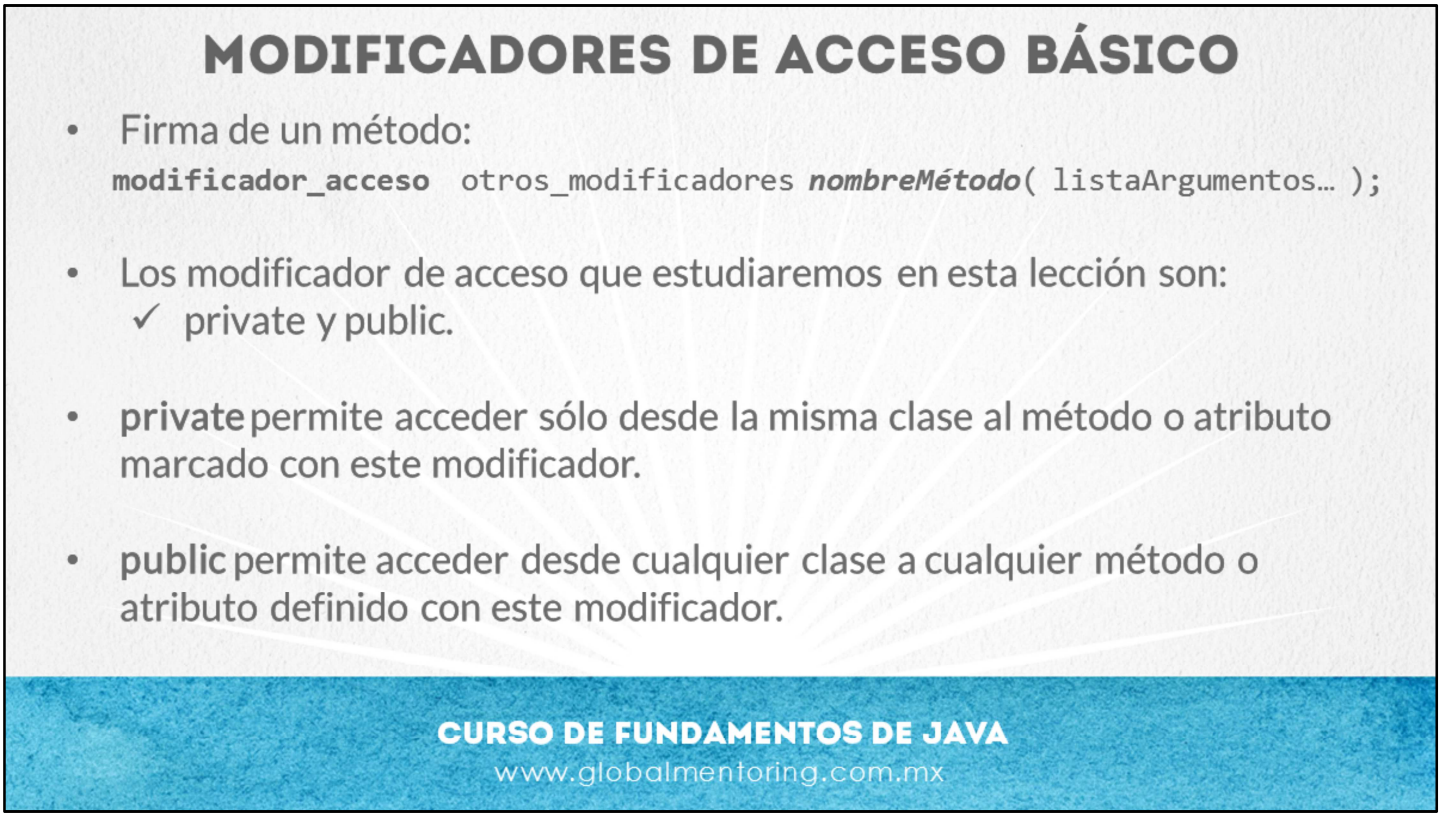


Una de las características más importantes de la programación orientada a objetos es el encapsulamiento.

Esta característica nos permite aislar los datos de nuestros objetos del acceso de otros objetos externos, y de esta manera restringir el acceso directo a los atributos o métodos que no deseemos permitir, ya que el estado de un objeto está generalmente oculto. Podemos entender por estado de un objeto como los valores actuales de cada uno de los atributos del objeto, y cualquier cambio en estos valores cambia el estado interno del objeto.

Al ser uno de los principios más importantes, el encapsulamiento se logra aplicar a través de los llamados modificadores de acceso. Existen cuatro modificadores de acceso en Java, los cuales son: private, package o default, protected y public.

En esta lección estudiaremos sólo los modificadores private y public, debido a que para el nivel de este curso es suficiente para aplicar el concepto de encapsulamiento, y en cursos posteriores veremos a detalle cómo implementar cada uno de estos modificadores de acceso.



Como hemos comentado, comenzaremos con el estudio de los modificadores: private y public.

Estos modificadores aplican en varias partes de una clase, sin embargo en esta lección estudiaremos cómo aplican en dos lugares, en los atributos de una clase, así como en los métodos de una clase.

El modificador private al agregarlo a un atributo o método de una clase básicamente lo que estamos indicando es que sólo será posible acceder al atributo o método marcado desde la misma clase y no desde alguna otra, por ello el nombre de privado.

En cambio el modificador public lo que indica es que es posible acceder a este a este atributo o método desde cualquier otra clase, por ello el nombre de publico.

A continuación veremos un ejemplo de cómo aplicar estos modificadores de acceso a un atributo o método de una clase Java con el objetivo de aplicar el concepto de encapsulamiento.

EJEMPLO DE LOS MODIFICADORES public Y private

Ejemplo uso de public y private para lograr encapsulamiento:

```
1  public class PruebaEncapsulamiento{//Clase1
2
3      public static void main(String[] args) {
4          Persona p1 = new Persona("Juan");
5          //Marca error,
6          //no se puede acceder directamente un atributo privado desde otra clase
7          p1.nombre = "Pedro";
8          //Si es posible acceder a un método o atributo publico desde otra clase
9          p1.obtenerNombre();
10     }
11 }
12
13 class Persona {//Clase2
14
15     private String nombre; //Uso de private en un atributo
16
17     public Persona(String nombre) { //Uso de public en un método
18         this.nombre = nombre;
19     }
20
21     public String obtenerNombre() { //Uso de public en un método
22         return nombre;
23     }
24 }
```


Podemos observar en el código mostrado el uso de private y public. Veamos algunos cambios en la clase Persona.

En la línea 15 el atributo nombre se le ha agregado el modificador de acceso private, de tal manera que cuando creemos un objeto de tipo Persona (línea 4), no podremos acceder directamente a modificar el valor del atributo nombre, ya que nos marcará un error de compilación (línea 7).

Si queremos leer el valor de este atributo, creamos un método para ello, y agregamos el modificador de acceso public con el objetivo de que cualquier otra clase pueda acceder a este valor. Por lo tanto, creamos el método obtenerNombre (línea 21) y agregamos el modificador de acceso public, para posteriormente en la clase de prueba acceder a este método por medio de la variable de tipo Persona llamada p1 (linea 9).

De esta manera lo que estamos haciendo es modificar nuestra clase Persona para que ahora el atributo no sea accesible ni para lectura ni para escritura de manera directa, y tengamos la necesidad de accederlo por medio de métodos públicos que creemos para cada atributo de nuestra clase.

Sin embargo, esta es una manera informal de aplicar el encapsulamiento en Java, veamos cómo aplicarlo de manera más formal en la siguiente lámina.

EJEMPLO DE ENCAPSULAMIENTO

Ejemplo de encapsulamiento:

```
1  public class PruebaEncapsulamiento {
2
3      public static void main(String[] args) {
4          //Creamos el objeto
5          Persona p1 = new Persona();
6          //Modificamos el atributo nombre
7          p1.setNombre("Juan");
8          //Accedemos al atributo nombre
9          System.out.println("Nombre:" + p1.getNombre());
10     }
11 }
12
13 class Persona {
14     //Atributo privado
15     private String nombre;
16     //Método publico para acceder al atributo nombre
17     public String getNombre(){
18         return nombre;
19     }
20     //Método publico para modificar al atributo nombre
21     public void setNombre(String nombre){
22         this.nombre = nombre;
23     }
24 }
```


En la lámina podemos observar un código más formal para el concepto de encapsulamiento, es decir, hemos convertido nuestros atributos a privados (línea 15), y por cada atributo hemos agregado dos métodos, uno para acceder al atributo (getNombre - línea 17), y otro para modificar el atributo (setNombre - línea 21).

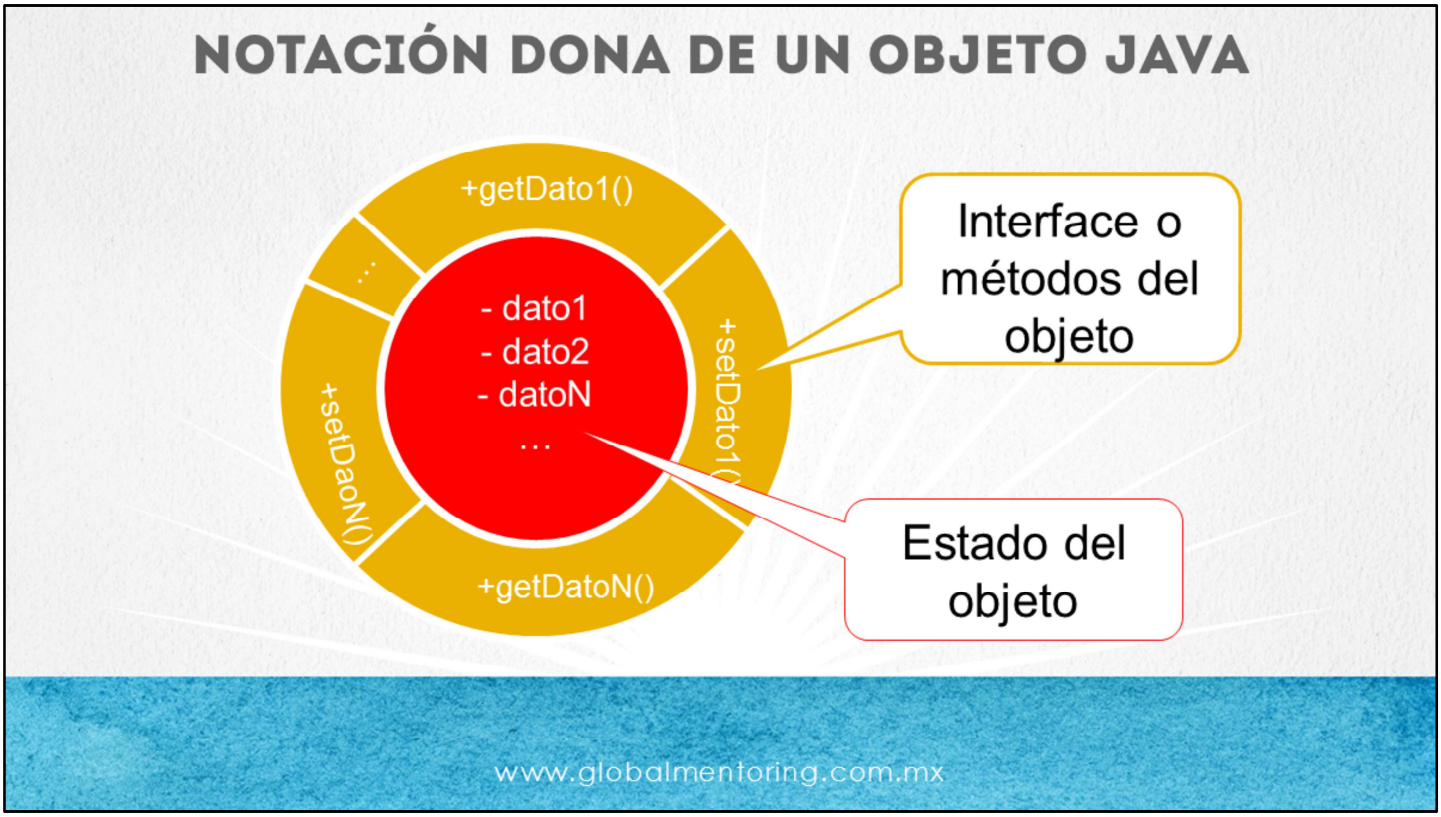
Estos métodos reciben un nombre especial según la función que realizan. Los métodos de tipo mutator modifican el valor del atributo de una clase y generalmente se utiliza el prefijo set y posteriormente el nombre del atributo de la clase respetando la notación de minúsculas/mayúsculas.

Los métodos de tipo accessors permiten recuperar el valor del atributo de una clase y se utiliza el prefijo is en caso de ser de tipo boolean y get para los demás tipos de datos, posteriormente lleva el nombre del atributo de igual manera respetando la notación de mayúsculas/minúsculas en Java.

En el código mostrado en el ejemplo podemos observar el uso de estos mutators y accessors. Como hemos comentado el objetivo es generar una interface que permita encapsular el estado del objeto Java y que no sea posible acceder directamente al estado del objeto y modificarlo, sino que sea a través de los mutators y si queremos conocer el estado del objeto sea a través de los accessors.

En este ejemplo debemos entender que la clase de prueba es la clase publica llamada PruebaEncapsulamiento, y que la clase en la que se aplicó el concepto de encapsulamiento fue sobre la clase Persona. Es decir que son clases distintas, y por ello el código que tenemos en el método main (líneas 4 a 9) es código que esta fuera de la clase Persona y allí radica el concepto de encapsulamiento, es decir, evitar que clases externas a la clase persona puedan acceder a los atributos de la clase Persona directamente, sin importar que sea una clase declarada en un mismo archivo, son clases distintas.

A continuación veremos otra forma de conceptualizar el encapsulamiento.



Como podemos observar en la lámina, la idea de encapsulamiento de la programación orientada a objetos es evitar el acceso y manipulación directa de otras clases a los datos de un objeto, por dato nos referimos a cualquier atributo de la clase en cuestión, esto es, el estado del objeto.

Para evitar el acceso directo de otras clases a los datos de la clase que deseamos encapsular lo primero que debemos hacer, según hemos comentado, es agregar el modificador de acceso private a nuestros atributos de clase, de esta manera únicamente los métodos de la misma clase son los que podrán acceder (leer/get) y manipular (modificar/set) los datos de la clase que estamos creando.

Por otro lado, ¿Cómo podrán acceder una clase externa a los datos de nuestros objetos encapsulados? La respuesta es creando ***métodos públicos por cada atributo privado***, los cuales puedan tanto modificar como recuperar el valor de nuestros datos o atributos de clase. A esto se le conoce como la ***interface*** del objeto, debido a que es a través de estos métodos que nos estaremos comunicando con el objeto creado y así poder leer el estado de cada dato y/o modificar también sus datos. Normalmente se crean dos métodos públicos por cada atributo privado, un método set y otro get.

Los métodos que modifican los datos se conocen como ***mutators*** y llevan el prefijo ***set*** (colocar) seguido del nombre de la variable en notación de mayúsculas/minúsculas. Por otro lado, los métodos que leen la información de los datos se conocen como ***accessors*** y llevan el prefijo de ***get*** (obtener) seguido del nombre de la variable, excepto en los casos en que la variable sea de tipo boolean, entonces el prefijo en lugar de get será ***is*** (es).

Ahora, ¿Por qué evitar la modificación directa del estado de un objeto en Java, y en general en la programación orientada a objetos? Lo que buscamos es que tengamos un control sobre la información y estado de nuestros objetos, de tal manera que si queremos por ejemplo modificar el dato1, podamos aplicar una validación sobre la información que queremos colocar en el dato1, y este sea un valor adecuado, como puede ser la restricción de recibir sólo números, entonces, antes de hacer la actualización del valor del dato1 podamos verificar que efectivamente el valor que se va a colocar sea un valor numérico, esto sólo por poner un ejemplo sencillo, pero la idea es que otros objetos no puedan directamente modificar el estado de nuestros objetos si no lo deseamos, y de esta manera el concepto de encapsulamiento en Java nos permite lograr esto, ocultando cualquier validación pero aplicándola en todo momento, respetando en todo momento las reglas que apliquen al estado del objeto.

En la lámina podemos observar que los atributos tienen un símbolo de – (menos), y que los métodos tienen un símbolo de + (mas), esto es parte de la notación de este diagrama de dona o de objetos, y lo que nos indica es que el signo de – es el modificador de acceso private, y el signo de + es el modificador de acceso public, de esta manera podemos observar gráficamente los elementos privados y públicos en nuestro objeto Java.

Esto es sólo un resumen de lo que hemos visto anteriormente, pero es otra forma en que podemos entender el concepto de encapsulamiento. Pondremos en práctica este concepto en el siguiente ejercicio.

En Global Mentoring promovemos la Pasión por la Tecnología Java. Te invitamos a visitar nuestro sitio Web donde encontrarás cursos Java Online desde Niveles Básicos, Intermedios y Avanzados, y así te conviertas en un experto programador Java.

Además agregamos nuevos cursos para que continúes con tu preparación como programador Java profesional. A continuación te presentamos nuestro listado de cursos:

- Lógica de Programación
- Fundamentos de Java
- Programación con Java
- Java con JDBC
- HTML, CSS y JavaScript
- Servlets y JSP's
- Struts Framework
- Hibernate Framework
- Spring Framework
- JavaServer Faces
- Java EE (EJB, JPA y Web Services)
- JBoss Administration
- Android con Java
- HTML5 y CSS3

Datos de Contacto:

Sitio Web: www.globalmentoring.com.mx
Email: informes@globalmentoring.com.mx