**SECCIÓN 18**

**ENCAPSULAMIENTO EN JAVA**

* **El estado de un objeto esta generalmente oculto.**

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.

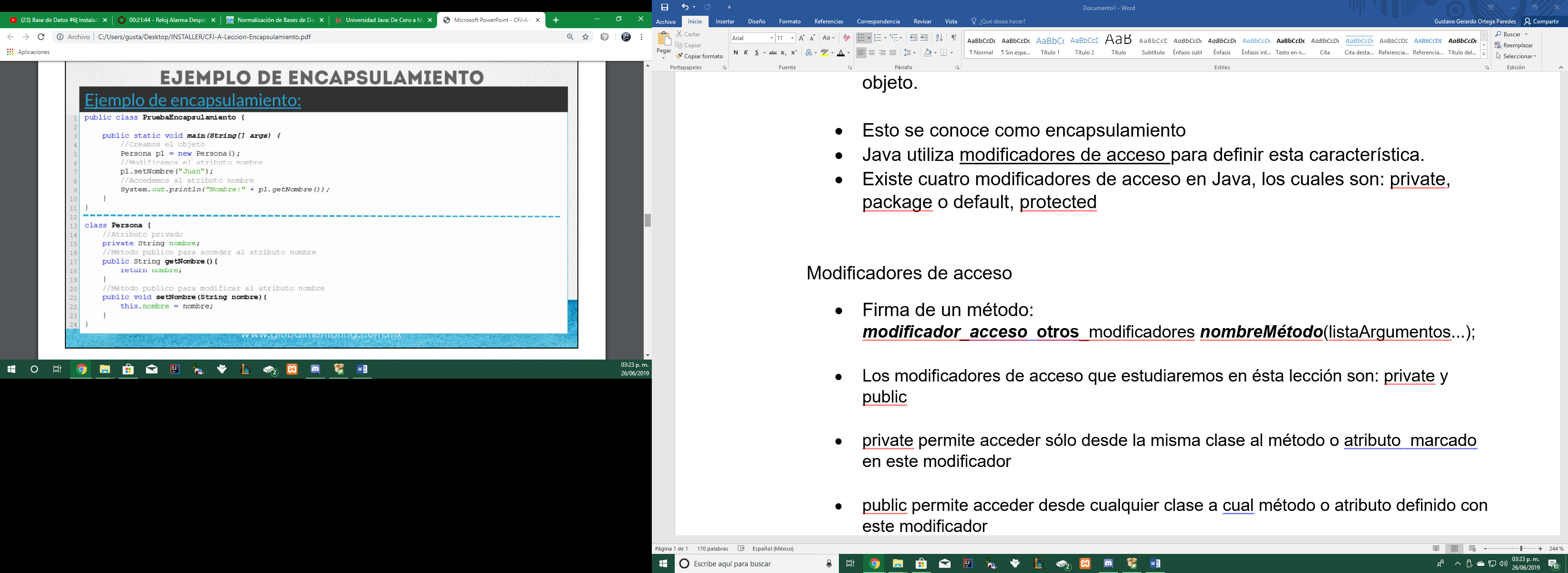
* Esto se conoce como encapsulamiento
* Java utiliza modificadores de acceso para definir esta característica.
* Existe cuatro modificadores de acceso en Java, los cuales son: private, package o default, protected

Modificadores de acceso

* Firma de un método:

***modificador\_acceso* otros**\_modificadores ***nombreMétodo***(listaArgumentos...);

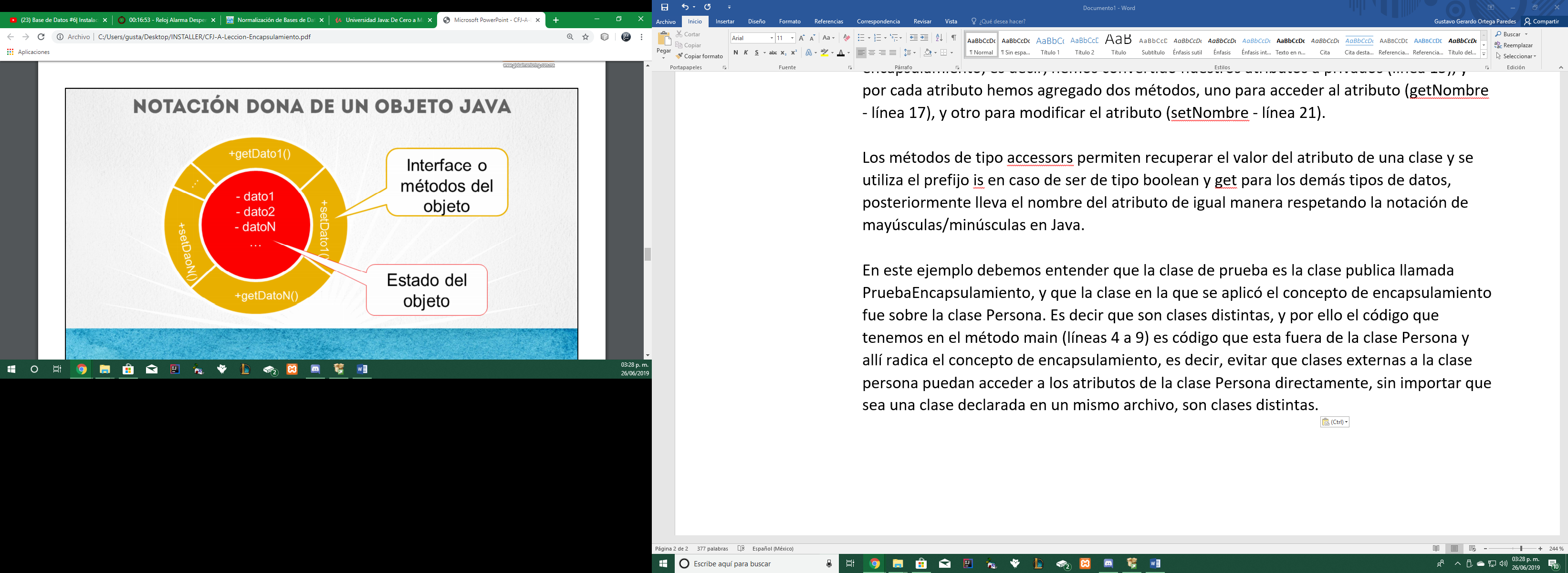
* Los modificadores de acceso que estudiaremos en ésta lección son: private y public
* private permite acceder sólo desde la misma clase al método o atributo marcado en este modificador
* public permite acceder desde cualquier clase a cual método o atributo definido con este modificador



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).

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 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.



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.