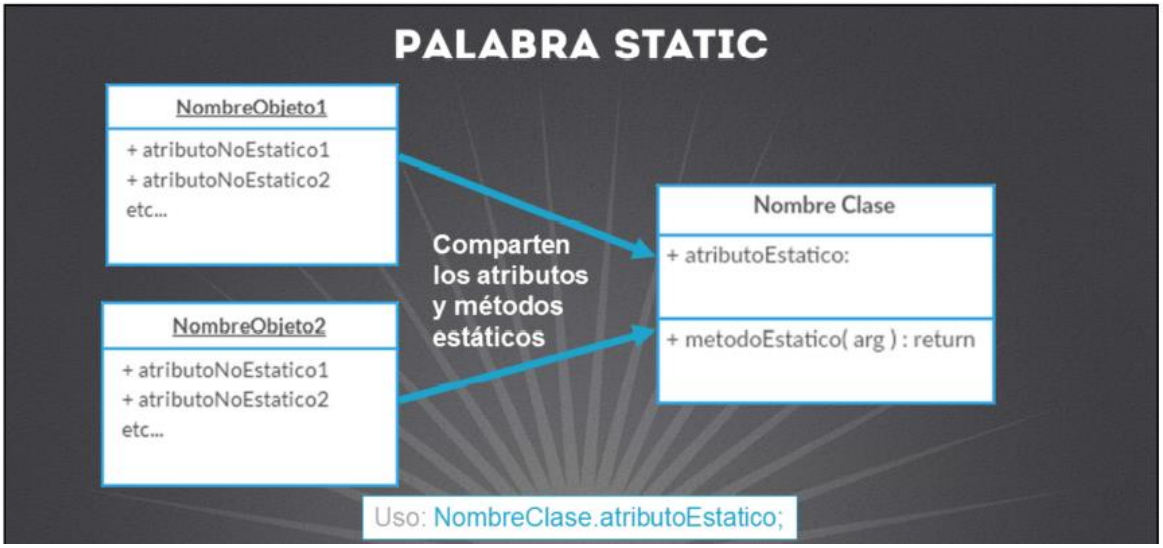
OPERADOR STATIC



Podemos utilizar la palabra **static** para interactuar con el contexto estático. Por ejemplo, si definimos un atributo o un método como estático, lo que estamos indicamos es que el atributo o método pertenecen a la clase y no al objeto.

Por ejemplo, si creamos un atributo sin usar la palabra static, que es como normalmente los definimos, cada que creamos un objeto se creará también una variable asociada al objeto que se crea, pero si definimos el atributo como estático, estamos indicando que el atributo solo se crea una vez, sin importar cuantos objetos se creen, sólo habrá una variable la cual se asocia a la clase y no al objeto. Y si un objeto accede al valor de la variable estática leerá el mismo valor que los demás objetos, y si un objeto modifica el valor estático, todos los demás objetos accederán al mismo valor ya que este valor está almacenado en la clase y no en los objetos de dicha clase.

Utilizando la palabra static es posible interactuar con la clase sin necesidad de haber creado un objeto de la misma. Por ejemplo, podemos acceder a atributos y/o métodos sin necesidad de haber instanciado ningún objeto de la clase que deseamos utilizar. Con el uso de la palabra ***static*** ya sea en el atributo o en el método que deseamos acceder podremos usar directamente el atributo o método sin necesidad de generar un objeto de la clase, solo colocando el nombre de la clase, el operador punto, y finalmente el nombre del atributo o método estático definido en la clase. Normalmente los atributos o métodos estáticos si deseamos que sean accedidos desde otras clases deberán contener el modificador de acceso ***public*** para que no tengan ningún problema en ser accedidos.

Por lo tanto, los atributos o métodos marcados con la palabra static se les conoce como miembros de clase o métodos de clase, ya que pertenecen a la clase y no a los objetos que se crean de dicha clase.

STATIC Y THIS

Uso de this y el contexto estático:

```
public class EjemploThisStatic {


    public static void main(String[] args) {
       //this no puede usarse dentro de un contexto estático
       //ya que solo tiene sentido cuando se ha creado un objeto
       this.imprimir("Hola"); //Marca error

    }

    //Metodo NO estatico
    public void imprimir(String s){
        System.out.println("Valor recibido:" + s);
    }
}
```

Una vez que hemos mencionado el contexto estático y el dinámico, podemos aclarar lo siguiente.

El operador this se asocia con el contexto dinámico, y debido a que el contexto estático carga primero que el contexto dinámico, el operador this no funciona en el contexto estático.

Es por esta razón que en el método main o cualquier otro método estático, no es posible utilizar el operador this.

En pocas palabras, un método estático puede acceder a otro método estático, pero no a un método NO estático, ya que el contexto dinámico aún no se crea.

Pero, un método dinámico (no estático) SI puede acceder a un método estático, ya que este contexto ya ha sido creado.

Ahora, podemos acceder a un método dinámico desde un método estático siempre y cuando creemos un nuevo objeto. De esta manera es si podemos acceder a métodos dinámicos, pero esto se debe a que una vez que creamos un objeto, por ejemplo dentro del método main, lo que estamos haciendo es creando el contexto dinámico, y por lo tanto ya es posible acceder a los métodos del objeto a través del mismo. De esta manera es que un objeto puede acceder tanto a los métodos no estáticos o estáticos. Sin embargo un método estático aunque pueda ser accedido por medio de un objeto, lo correcto es que utilice el nombre de la clase y no el nombre del objeto.

EJEMPLO CÓDIGO ESTÁTICO

```
1  public class EjemploStatic {
2
3      public static void main(String[] args) {
4          Persona p1 = new Persona("Juan");
5          System.out.println("Persona1: " + p1);
6
7          //Imprimimos el contadorPersonas
8          System.out.println("No. Personas:" + Persona.getContadorPersonas());
9      }
10 }
11
12 class Persona {
13
14     private String nombre;
15     private int idPersona;
16     private static int contadorPersonas;
17
18     public Persona(String nombre){
19         //Cada que creamos un objeto persona incrementamos el contador para obtener un nuevo idPersona
20         contadorPersonas++;
21         //asignamos el nuevo valor al idPersona
22         idPersona = contadorPersonas;
23         //Asignamos el nombre recibido
24         this.nombre = nombre;
25     }
26
27     public static int getContadorPersonas(){
28         return contadorPersonas;
29     }
30 }
```

En el ejemplo mostrado podemos observar un ejemplo del uso de la palabra static.

Por un lado declaramos una variable de tipo static llamada contadorPersonas. Al declarar esta variable como estática, lo que buscamos es que cada vez que creamos un objeto de tipo Persona, vamos a incrementar el valor de esta variable. Esto lo podemos lograr utilizando el constructor de la clase, ya que cada que creamos un objeto de tipo Persona, se mandará llamar el constructor de la clase, y allí podemos incrementar esta variable con cada llamada realizada al constructor (línea 20).

Una vez que se ha incrementado esta variable estática, este valor aplica para todos los objetos, por lo tanto cada que incrementamos esta variable puede ser consultada por todos los objetos que creemos y de esta manera es que esta variable no comienza desde cero cada que creamos un nuevo objeto, sino que se queda en el último valor y se incrementa una vez más, esto debido a que esta variable pertenece a la clase y no a los objetos.

Una vez que ya tenemos el nuevo valor, podemos entonces asignar el nuevo valor al atributo idPersona (línea 22), y el valor de idPersona al no ser estático será único para cada objeto, de allí que podemos estar seguros que el valor será único para cada objeto creado de tipo Persona.

Otra sección importante de este código es la línea 27, en la cual declaramos el método getContadorPersonas( ), con este método el cual es estático, podemos recuperar el valor de contadorPersonas, para saber cuantos objetos de tipo Persona se han creado.

Ahora, en el método main, el cual es un método estático, vamos a hacer uso de la clase Persona. En primer lugar crearemos un objeto de tipo Persona, y asignamos el valor de Juan al atributo nombre con ayuda del constructor de la clase.

Desde el momento que estamos utilizando la palabra new, quiere decir que ya estamos trabajando con el contexto dinámico, esto permite trabajar ya con el contexto dinámico (objetos) y el contexto estático (clases).

Por lo tanto al crear el objeto persona, el constructor de la clase se encarga de asignar el idPersona con un valor único al incrementar la variable contadorPersonas, la cual al ser estática permite que el conteo de objetos Persona sea correcto.

En este código estamos omitiendo la implementación del método toString para la clase Persona, el cual se debe agregar para observar el estado del objeto persona impreso en la línea 5.

Finalmente, una vez que hemos terminado de crear objetos de tipo persona, podemos mandar a imprimir el número total de objetos Persona que se han creado, y aquí es donde podemos observar la notación que hemos platicado, para mandar a llamar un método estático público de una clase basta con colocar el nombre de la clase, el operador punto y el nombre del método estático, es decir que no debemos utilizar el objeto, de hecho no es buena práctica y si insistimos en usar una variable de tipo object para mandar a llamar un método estático nos mostrará un warning, avisándonos que es una mala práctica.