Static en Java

¿Qué es Static en Java?

En una aplicación Java, cualquier clase, atributo, método que lleve en su declaración el modificador (de no acceso) static entra a formar parte del contexto static de la aplicación:

En una clase anidada

```
private static class miClaseStatic{…}
```

En Java las clases static tienen que estar obligatoriamente declaradas dentro de una clase no static.

En un atributo

```
private static int miVariableDeClase
```

En un método

```
public static void miMetodoStatic(){…}
```

Atributos Static en Java

1 - Es una variable que pertenece a la clase y no al objeto (instancia).

2 - Las variables static se inicializan solo una vez, al inicio de la ejecución.

3 - Las variables static se inicializarán primero, antes que cualquier variable de instancia (no static).

4 - Una sola copia para ser compartida por todas las instancias de la clase

5 - Una variable static es accedida por el nombre de clase (LaClase.variableStatic).

Métodos Static en Java

1 - Es un método que pertenece a la clase y no al objeto (instancia).

2 - Un método static solo puede acceder a datos static. No puede acceder a datos no static (variables de instancia).

3 - Un método static solo puede llamar a otros métodos static y no puede invocar un método no static dentro de su implementación.

4 - Un método static es accedido directamente por el nombre de la clase (LaClase.elMetodoStatic()).

5 - Un método static no puede hacer referencia a “this” o “super” palabras pertenecientes al contexto “no static”.

Clases (Anidadas) Static en Java

En Java, el concepto de clases anidadas hace referencia al hecho de que una clase, de la misma manera que con un atributo o un método, puede “albergar” otra clase como miembro.

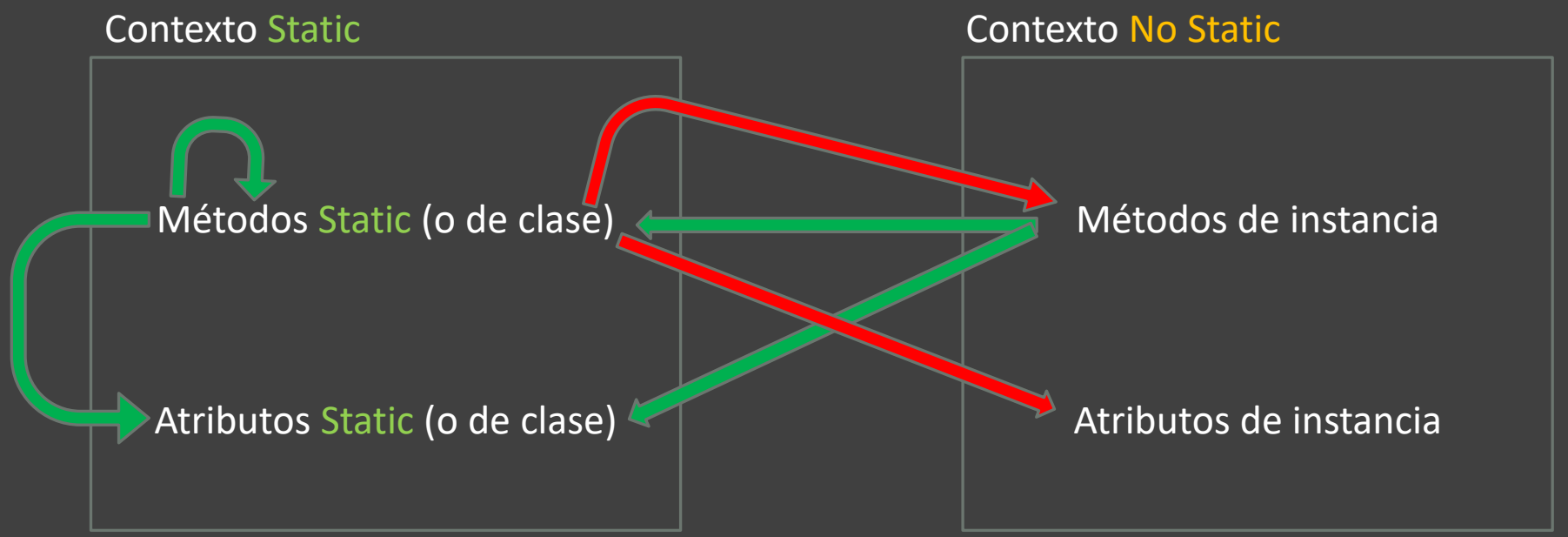
Tenemos dos tipos de clases anidadas en Java:

1 - Clases anidadas (inner classes)
2 – Clases anidadas static (static nested classes)

En Java, no podemos crear una clase externa como static, pero podemos crear una clase static interna anidada dentro de otra clase no static.

En realidad la clase static no es static en si misma. En Java ninguna clase es static.
La palabra static, en este caso, hace alusión a que es un miembro static de la clase exterior

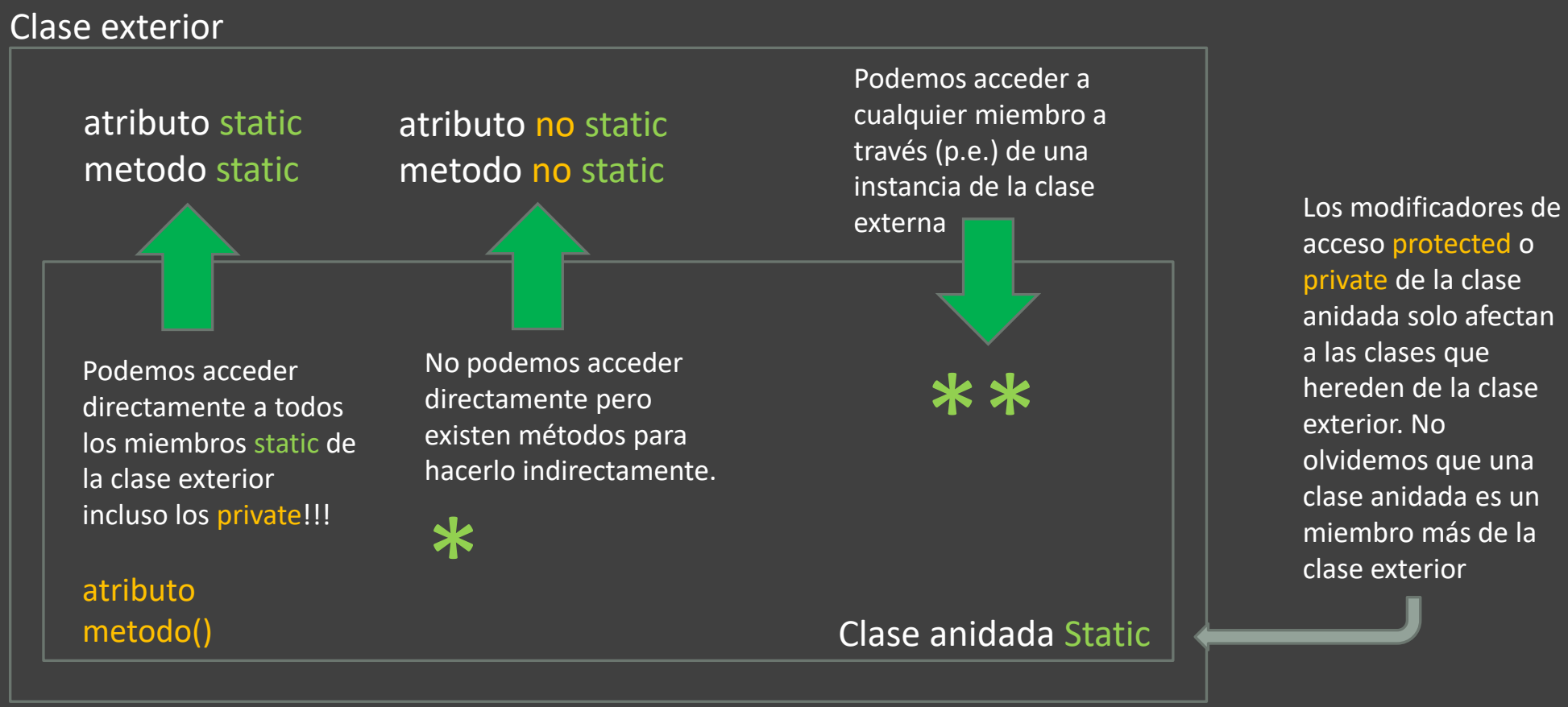
Contexto Static en Java (Métodos)



Un método static sólo puede acceder a los miembros de datos static y a los métodos static de otra clase o de la misma clase, pero no puede acceder a los métodos y variables no static,

Los métodos no static pueden acceder a cualquier método static y variable static también, sin utilizar una instancia de la clase.

Contexto Static en Java (Clases anidadas)



*

```
public class Exterior {
   static int a = 10;
   int b = 20;
   static class Interior {

            public static void display(Exterior c) {
                  System.out.println(“Miembre estático exterior = “ + a);
                  System.out.println(“Miembro no estático exterior = “ + exterior.b);
            }
   }

   public static void main(String[] argv) {
      Exterior exterior = new Exterior();
      Interior.display(exterior);
   }
```

**

```
public class Exterior {
        static class Interior {
                private int x = 10;
        }

         void print() {
                Interior interior = new Interior();
                System.out.println ("Miembro interior = " + interior.x);
         }
}

  public static void main(String[] args) {
    Exterior exterior = new Exterior();
    exterior.print();
  }
```

Instanciación de clases con miembros Static en Java

Cuando instanciamos un objeto de una clase, el objeto no incluye sus miembros static:

Solo la clase tiene acceso al atributo: Perro.nPerros

Solo la clase tiene acceso al método: Perro.método()

Perro

```
int static nPerros
String raza
String color
int edad
String nombre

Perro()
correr()
jugar()
static metodo()
```

```
raza: terrier
color: blanco
edad: 5
nombre: Toby

correr()
jugar()
```

```
raza: terrier
color: crema
edad: 9
nombre: Fifi

correr()
jugar()
```

Las instancias comparten el atributo nPerros

¿Puede un método Static ser abstracto en Java?

Un método abstracto DEBE ser sobrescrito en una subclase. Debe ser implementado.

En Java, un miembro static (método o atributo) no puede ser sobrescrito por subclases. Un miembro estático puede ser ocultado desde una subclase, pero no sobreescrito: un método que no se puede implementar no tendría ningún sentido.

La respuesta sería NO

Un método static no puede ser sobreescrito (anulado) por subclases y como un método abstracto necesita ser sobreescrito para implementado llegamos a una contradicción lógica: static y abstract no pueden convivir. Un método no puede ser static y abstract al mismo tiempo.

Por otro lado un método static abstract pertenecería a una clase abstracta y no podría jamás ser utilizado por dicha clase abstracta ya que estaría sin implementar.

Herencia de métodos y atributos static en Java I

Todos los métodos y atributos accesibles (public y protected) son heredados por las subclases. La herencia de miembros está estrechamente ligada a su accesibilidad declarada (una subclase no hereda los miembros privados de su superclase!: una subclase no puede acceder o modificar los miembros privados de su superclase) … pero un objeto que es una instancia de la subclase ciertamente contiene los campos privados.

La única diferencia entre los métodos heredados static (de clase) y los métodos heredados no static (de instancia) es que cuando intentamos sobreescribir un método static de la superclase en la subclase, el método static en la superclase simplemente se oculta, no se sobrescribe ya que esto no está permitido en Java (en Java no podemos sobreescribir un método static). Los métodos static no son polimórficos!!!.

Herencia de métodos y atributos static en Java II

OCULTACIÓN: si un método static que esté presente en la subclase tiene la misma firma que un método static de su clase padre, el método de la clase hija oculta el método de la clase padre (el método de la clase padre no se hereda). Hay que tener en cuenta que en Java no se pueden sobreescribir métodos static.

SOBREESCRITURA: por el contrario, si un método no static que esté presente en la subclase tiene la misma firma que un método no static de su clase padre, el método de la clase hija sobreescribe el método de la clase padre (el método de la clase padre se hereda).

La distinción entre ocultar y anular (sobreescribir) tiene importantes implicaciones:

1) La versión del método sobreescrito que se invoca es la de la subclase.
2) La versión del método oculto que se invoca depende de si se invoca desde la superclase o desde la subclase.

El método main (Static) en Java

El método main() es el punto de entrada de la aplicación, es decir, es el punto en el que comienza la ejecución de esta. Es por ello que ha de ser public y static.

public: Un método público es accesible desde fuera de la clase.
static: Un método estático es aquel que se puede ejecutar sin una instancia de la clase.

Al ser el punto de entrada, ha de ser accesible desde fuera de la clase en la que se encuentra. Además, al ser lo primero que se ejecuta, ha de ser posible su ejecución antes de instanciar un objeto.

Como consecuencia directa de ser la primera línea de código que se ejecuta, no tiene sentido que tenga un tipo de devolución distinto de void, ya que no hay un código anterior que pueda hacer algo con ese valor. El método main() en Java siempre tiene un tipo de devolución void.