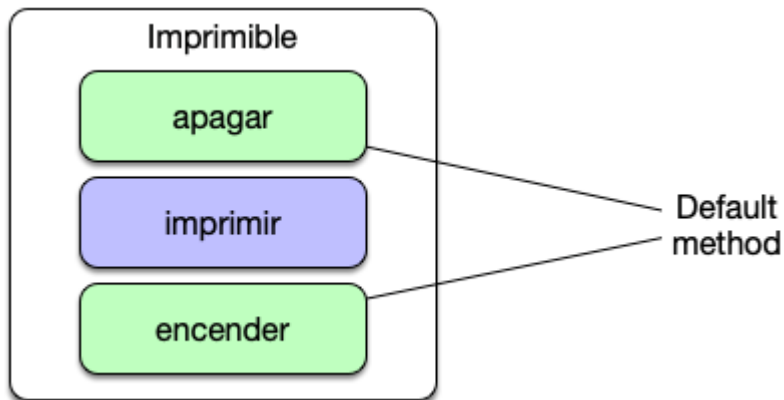


El concepto de Java interface private method es relativamente nuevo y llega con Java 9 . ¿Cómo funciona este concepto y para que se usa? . Para ello tenemos que recordar que es Java 8 la primera versión de Java que soporta implementar métodos a nivel de interface .Tanto **métodos estáticos** como **default methods**. Vamos a ver un ejemplo sencillo de qué es lo que pueden aportar los métodos privados a nivel de un interface ,algo que de entrada suena raro. Para ello nos vamos a construir el interface Imprimible que dispondrá de dos default methods y un método clásico.



Veamos su código:

```
package com.arquitecturajava;

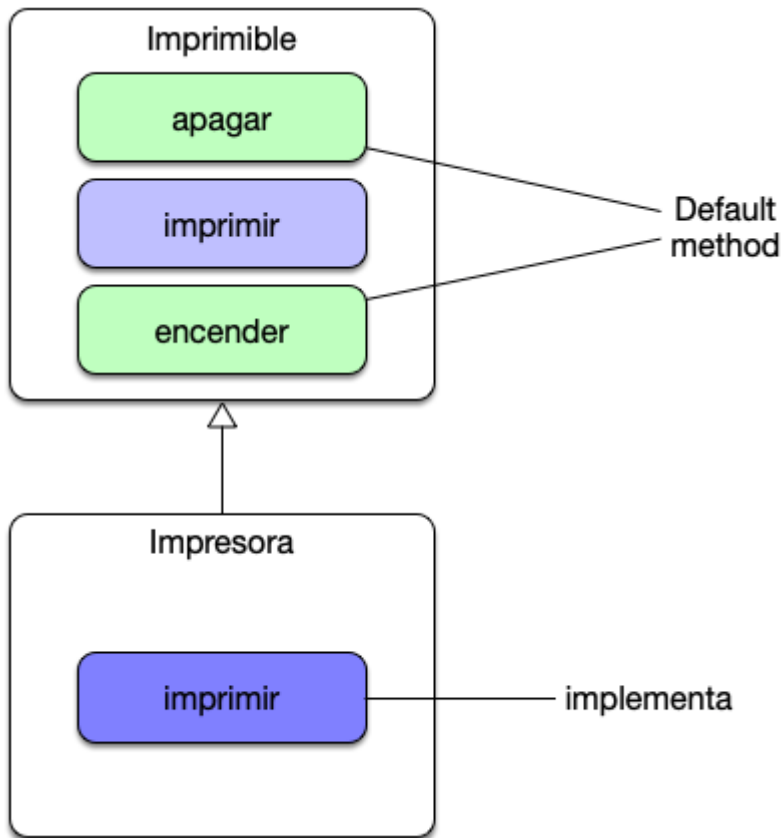
public interface Imprimible {

    public default void encender() {

        System.out.println("encendiendo el dispositivo");
    }
    public default void apagar() {
```

```
        System.out.println("apagando el dispositivo");  
    }  
    public void imprimir();  
}
```

Ahora necesitaremos una clase que implemente el interface y que de forma automática heredará los métodos encender y apagar.



El código es el siguiente:

```
package com.arquitecturajava;

public class Impresora implements Imprimible {

    @Override
    public void imprimir() {
        System.out.println("imprimiendo con la impresora");
    }

}
```

Ya únicamente nos queda por construir el programa main que ejecuta el código:

```
package com.arquitecturajava;

public class Principal {

    public static void main(String[] args) {
        Imprimible i= new Impresora();
        i.encender();
        i.imprimir();
        i.apagar();
    }

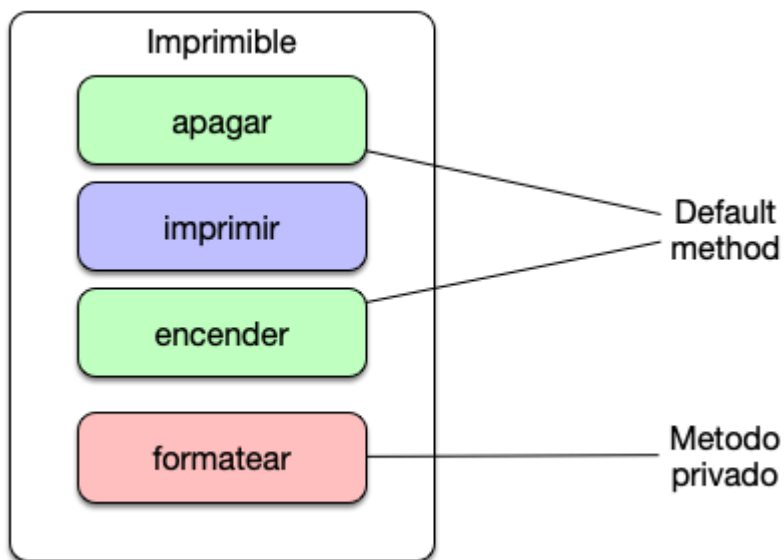
}
```

El resultado lo podemos ver impreso en la consola:

```
<terminated> Principal (2) [Java Applica  
encendiendo el dispositivo  
imprimiendo con la impresora  
apagando el dispositivo
```

Java private interface Method

Ahora bien hay situaciones en las cuales quizás necesitemos formatear de alguna forma los mensajes de la impresora . Para ello necesitaremos añadir un método adicional que reciba un texto como parámetro y le aplique un formato. ¿Cómo podemos hacerlo con Java 9? . Muy sencillo podemos declarar un Java interface private method y definir el formateador a ese nivel .



Vamos a ver cómo se construye esta funcionalidad.

```
package com.arquitecturajava.ejemplo2;

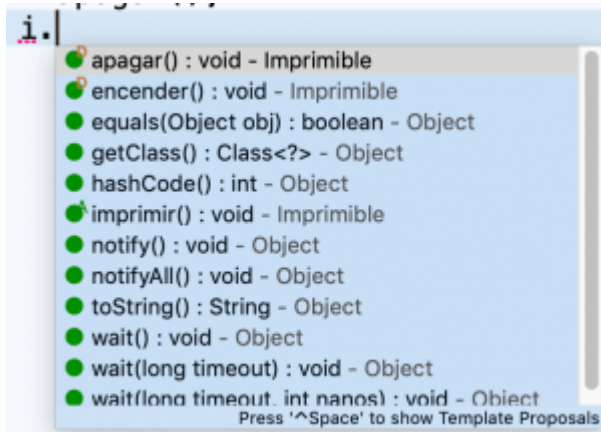
public interface Imprimible {

    public default void encender() {

        formatear("encendiendo el dispositivo");
    }
    public default void apagar() {

        formatear("apagando el dispositivo");
    }
    private static void formatear( String mensaje) {
        System.out.println("*****");
        System.out.println(mensaje);
        System.out.println("*****");
    }
    public void imprimir();
}
```

Cómo podemos ver hemos declarado el método de formatear como un Java private interface method. Si intentamos utilizar a nivel de programa main ni siquiera estará disponible. Esto es debido a que el método es privado a nivel de la interface.



Ahora bien aporta la funcionalidad adicional que necesitamos. Ya que si ejecutamos el programa main todos los métodos aparecerán formateados.

```
*****
encendiendo el dispositivo
*****
imprimiendo con la impresora
*****
apagando el dispositivo
*****
```

Como siempre Java sigue evolucionando.

Otros artículos relacionados:

1. [Java 8 interface static methods y reutilizacion](#)
2. [Java 8 Factory Pattern y su implementación](#)
3. [Java 8 Functional Interfaces y sus tipos](#)
4. [Java 9](#)