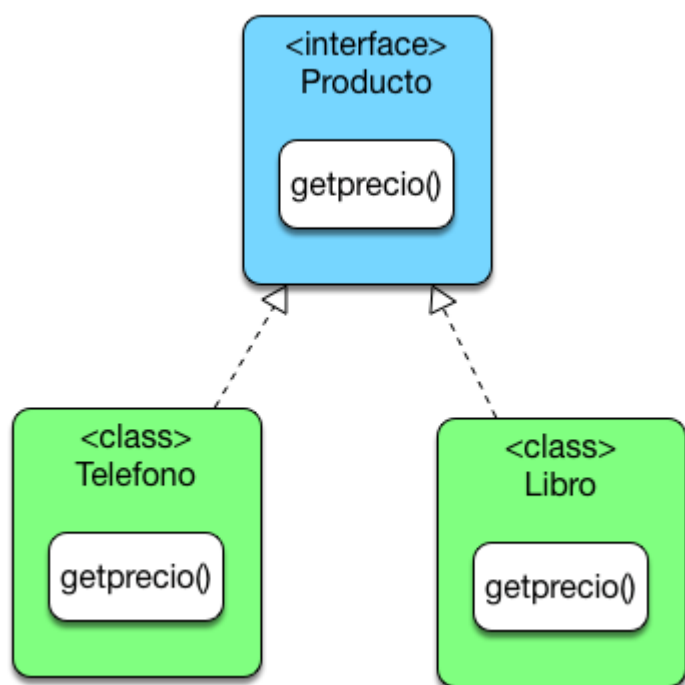


El uso de Java 8 interface static methods genera muchas dudas entre los desarrolladores. ¿Para qué podemos necesitar usar métodos estáticos a nivel de interfaces?. La respuesta tiene que ver con el concepto de reutilización de código y agrupamiento de responsabilidades . Vamos a construir como siempre un ejemplo que nos ayude a clarificar las dudas. Para ello vamos a partir de dos clases independientes (Teléfono y Libro) que implementan el mismo interface Producto . Este nos permite acceder al precio de dos clases diferentes.



Vamos a ver su código:

```
package com.arquitecturajava.ejemplo2;

public class Libro implements Producto {

    private String titulo;
    private int precio;
```

```
public Libro(String titulo, int precio) {  
    super();  
    this.titulo = titulo;  
    this.precio = precio;  
}  
public String getTitulo() {  
    return titulo;  
}  
public void setTitulo(String titulo) {  
    this.titulo = titulo;  
}  
public int getPrecio() {  
    return precio;  
}  
public void setPrecio(int precio) {  
    this.precio = precio;  
}  
}
```

```
package com.arquitecturajava.ejemplo2;
```

```
public class Telefono implements Producto{
```

```
    private String modelo;  
    private String marca;  
    private int precio;  
    public String getModelo() {  
        return modelo;  
    }
```

```

    }
    public void setModelo(String modelo) {
        this.modelo = modelo;
    }
    public String getMarca() {
        return marca;
    }
    public void setMarca(String marca) {
        this.marca = marca;
    }
    public int getPrecio() {
        return precio;
    }
    public void setPrecio(int precio) {
        this.precio = precio;
    }
    public Telefono(String modelo, String marca, int precio) {
        super();
        this.modelo = modelo;
        this.marca = marca;
        this.precio = precio;
    }
}

```

```
package com.arquitecturajava.ejemplo2;
```

```
import java.util.List;
```

```
public interface Producto {  
  
    public int getPrecio();  
  
}
```

Java 8 interface static methods

Es momento de construir varios objetos (Libro , Teléfono) añadirlos a una lista y calcular el precio total de todos los objetos. Para ello podemos usar un programa main y operar con un bucle for.

```
package com.arquitecturajava.ejemplo2;  
  
import java.util.Arrays;  
import java.util.List;  
  
public class Principal {  
  
    public static void main(String[] args) {  
  
        Libro l1= new Libro("Dune", 10);  
        Libro l2= new Libro("Java",20);  
        Telefono t1= new Telefono("iphone","apple",600);  
  
        List<Producto> listaProductos=Arrays.asList(l1,l2,t1);  
  
        int total=0;
```

```

        for(Producto p:listaProductos) {

            total+=p.getPrecio();

        }
        System.out.println(total);

    }
}

```

Esto nos imprimirá:



De la misma forma podríamos realizar la operación con un Stream :

```

System.out.println(listaProductos.stream().mapToInt((p) -
&gt;p.getPrecio()).sum());

```

El resultado será idéntico:



¿Es esta la solución idónea? . La realidad es que NO , se trata de una operación muy habitual y nos gustaría poder reutilizarla pero no sabemos donde ubicarla. A partir de Java

8 podemos crear métodos estáticos en los interfaces .



Por lo tanto parece claro que nos vendría bien un método para sumar todos los productos en el interface Producto ya que es el compartido por muchas clases.

```
package com.arquitecturajava.ejemplo2;
```

```
import java.util.List;
```

```
public interface Producto {
```

```
    public int getPrecio();
```

```
    public static int importeTotal(List<Producto>
listaProductos) {
```

```
        return listaProductos.stream().mapToInt((p) -
&gt;p.getPrecio()).sum();
    }
```

```
}
```

De esta manera será mucho más sencillo reutilizar el código. Esta es una de las utilidades fundamentales de los Java 8 interface static methods.

Otros artículos relacionados:

1. [Java 8 Lambda Expressions \(I\)](#)
2. [Java Lambda reduce y wrappers](#)
3. [Programación Funcional, Java 8 Streams](#)