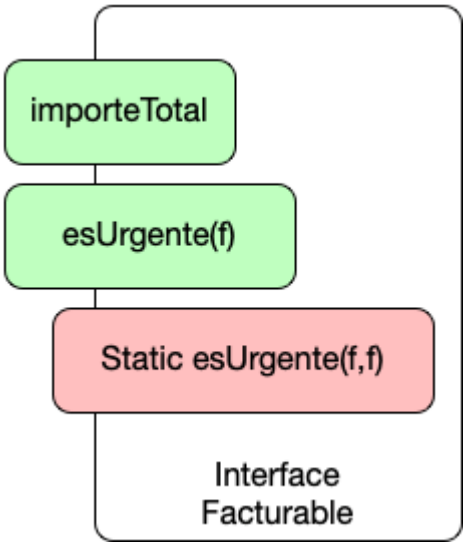
¿Java Interface Metodos privados? . Esta es una pregunta que nos puede parecer absurda ya que un interface esta diseñado por métodos eminentemente públicos. Si a un método de un interface en Java no se le asigna un ámbito de visibilidad es publico por defecto . La pregunta que uno se puede hacer es ¿Puede un interface de Java soportar métodos privados? La respuesta parece trivial y debiera ser que No. Lamentablemente las cosas no son tan sencillas como parece y un interface a partir de Java 8 puede disponer de métodos con código y dentro de ellos podrían existir métodos privados .



Java Interface Metodos privados

Esos métodos con código pueden ser de dos tipos . Default Methods que hacen referencia a métodos que afectan a las instancias de la clase o Static Methods que son métodos estáticos como su nombre indica y pertenecen a la clase en sí. Estos métodos podrían ser privados a nivel de una interface. Vamos a poner un ejemplo , imaginémonos que disponemos de un interface que se denomina Facturable. Ese interface tiene un método importe() que nos indica el importe Facturable de un Objeto.

```
public interface Facturable {
        double importeTotal();
}
```

Puedo por lo tanto crear una clase que implemente este interface :

```
public class Caja implements Facturable {

	private double importe;

	public Caja(double importe) {
		super();
		this.importe = importe;
	}

	@Override
	public double importeTotal() {
		// TODO Auto-generated method stub
		return importe;
	}

}
```

Con lo cual nos encontramos en una situación clásica una Caja puede ser Facturable y la implementación del interface se puede usar:

```
public class PrincipalFacturable {

	public static void main(String[] args) {
		Caja c=new Caja(100);
		Caja c2= new Caja(200);
		System.out.println(c.importeTotal());
	}
}
```

El resultado será :

```
100.0
```

Ahora bien cuando uno tiene elementos facturables puede hacer unos que sean mas urgentes de cobrar que otros. Podríamos diseñar el interface con un método estático privado y un método default (de instancia).

```
public interface Facturable {
	double importeTotal();
	private static Facturable esMasUrgente(Facturable a ,
Facturable b) {
		return a.importeTotal()>b.importeTotal()?a:b;
	}
	default Facturable esMasUrgente(Facturable b) {
		return esMasUrgente(this,b);
	}

}
```

De esta manera tenemos un Java Interface Private Methods . El método privado define que elemento Facturable es más urgente de cobrar y luego usa un default método publico para hacerle visible para la clase. Es momento de usarle.

```
public class PrincipalFacturable {

	public static void main(String[] args) {
		Caja c=new Caja(100);
		Caja c2= new Caja(200);
		System.out.println(c.esMasUrgente(c2).importeTotal());
	}
}
```

Esto nos imprimirá 200 que es el importe más urgente de cobrar utilizando el Default

Method esMasUrgente que delega en el Java Interface Private Method.

Otros artículos relacionados

- Java Comparable Interface y Ordenaciones
- Clases Abstractas vs Interfaces
- Java 8 Functional Interfaces y sus tipos
- Java Package , Diseño e Interfaces
- Java Interface