

## 7.E. Interfaces.

### 1. Interfaces.

#### 1.1. Concepto de interfaz.

Una **interfaz** en Java consiste esencialmente en una lista de declaraciones de métodos sin implementar, junto con un conjunto de constantes.

Estos **métodos sin implementar** indican un **comportamiento**, un tipo de conducta, aunque no especifican cómo será ese **comportamiento** (**implementación**), pues eso dependerá de las características específicas de cada clase que decida implementar esa **interfaz**. Podría decirse que una **interfaz** se encarga de establecer qué **comportamientos** hay que tener (qué **métodos**), pero no dice nada de cómo deben llevarse a cabo esos **comportamientos** (**implementación**). Se indica sólo la **forma**, no la **implementación**.

En cierto modo podrías imaginar el concepto de **interfaz** como un **guión que dice**: "éste es el protocolo de comunicación que deben presentar todas las clases que implementen esta interfaz". Se proporciona una lista de **métodos públicos** y, si quieres dotar a tu clase de esa **interfaz**, tendrás que definir todos y cada uno de esos **métodos públicos**.

En conclusión: una **interfaz** se encarga de establecer unas líneas generales sobre los **comportamientos (métodos)** que deberían tener los objetos de toda clase que implemente esa interfaz, es decir, que no indican lo que el objeto es (de eso se encarga la clase y sus superclases), sino acciones (**capacidades**) que el objeto debería ser capaz de realizar. Es por esto que el nombre de muchas interfaces en Java termina con sufijos del tipo "-able", "-or", "-ente" y cosas del estilo, que significan algo así como **capacidad o habilidad** para hacer o ser receptores de algo (**configurable**, **serializable**, **modificable**, **clonable**, **ejecutable**, **administrador**, **servidor**, **buscador**, etc.), dando así la idea de que se tiene la capacidad de llevar a cabo el conjunto de acciones especificadas en la **interfaz**.

Imagínate por ejemplo la clase **Coche**, subclase de **Vehículo**. Los coches son **vehículos a motor**, lo cual implica una serie de acciones como, por ejemplo, **arrancar el motor** o **detener el motor**. Esa acción no la puedes heredar de **Vehículo**, pues no todos los vehículos tienen porqué ser a motor (piensa por ejemplo en una clase **Bicicleta**), y no puedes heredar de otra clase pues ya heredas de **Vehículo**. Una solución podría ser crear una **interfaz Arrancable**, que proporcione los métodos típicos de un **objeto a motor** (no necesariamente vehículos). De este modo la clase **Coche** sigue siendo subclase de **Vehículo**, pero también implementaría los comportamientos de la interfaz **Arrancable**, los cuales podrían ser también implementados por otras clases, hereden o no de **Vehículo** (por ejemplo una clase **Motocicleta** o bien una clase **Motosierra**). La clase **Coche** implementará su método **arrancar** de una manera, la clase **Motocicleta** lo hará de otra (aunque bastante parecida) y la clase **Motosierra** de otra forma probablemente muy diferente, pero todos tendrán su propia versión del método **arrancar** como parte de la interfaz **Arrancable**.

Según esta concepción, podrías hacerte la siguiente pregunta: **¿podrá una clase implementar varias interfaces?** La respuesta en este caso sí es **afirmativa**.

Una clase puede adoptar distintos modelos de comportamiento establecidos en diferentes interfaces. **Es decir una clase puede implementar varias interfaces.**

#### Autoevaluación

Una interfaz en Java no puede contener la implementación de un método mientras que una clase abstracta sí. ¿Verdadero o Falso?

- ☒ Verdadero  
☐ Falso