Interfaces en Java

Joselu Cantero

¿Qué es una Interface?

Una interface es conceptualmente similar a una clase abstracta, en cuanto a que podemos especificar uno o más métodos (o ninguno en el caso de las interfaces de marcado) que no tienen cuerpo: es decir son métodos que no están implementados.

Las interfaces especifican un protocolo de comportamiento para las clases que las utilizan, lo que significa que sus métodos deben ser obligatoriamente implementados por una clase para que se definan sus acciones o bien declararse como clase abstracta.

Por lo tanto, una interface especifica qué se debe hacer, pero no cómo hacerlo (esto lo hacen las clases).

Una vez que se define una interface cualquier clase puede implementarla.

Una interface puede emplearse también para declarar constantes que luego puedan ser utilizadas por otras clases.

Una clase puede implementar un número ilimitado de interfaces. De esta forma Java consigue la capacidad de imitar la herencia múltiple.

Diferencias entre Interface y clase abstracta

Una interface puede parecer similar a una clase abstracta, pero existen una serie de notables diferencias:

1 - Todos los métodos de una interfaz se declaran implícitamente como abstract y public (se pueden omitir).

2 - Una clase abstracta no puede implementar los métodos declarados como abstractos, una interface no puede implementar ningún método (ya que todos son abstractos).

3 - Una interface no declara variables de instancia. Esto es porque por defecto todas las variables declaradas dentro de una interface son public, static y final (se pueden omitir). El hecho de ser static elimina toda posibilidad. Una variable static es una variable de clase, por el contrario una variable de instancia es toda variable que pertenece a las instancias de una clase.

4 - Una clase abstracta puede implementar varias interfaces, pero sólo puede tener una clase ascendiente directa.

5 - Una clase abstracta pertenece a una jerarquía de clases mientras que una interface no pertenece a una jerarquía de clases. En consecuencia, clases sin relación de herencia pueden implementar la misma interface.

Interface tips 

1 - Una interface puede heredar (extends) de una o varias interfaces.

2 - Una clase, tanto abstracta como concreta, puede implementar (implements) una o varias interfaces.

3 - Una clase abstracta no está obligada a implementar los métodos de una interface que dicha clase implemente.

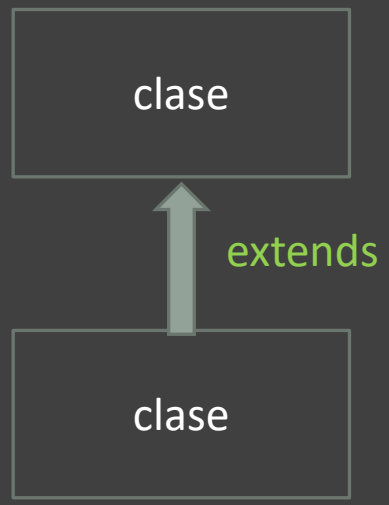
4 - Una clase concreta está obligada a implementar los métodos de una interface que dicha clase implemente.

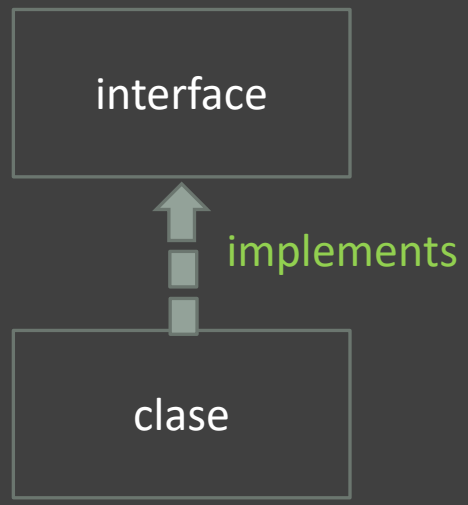
5 - Una clase concreta que herede de una clase abstracta que implementa una o varias interfaces está obligada a implementar los métodos de la interface si no los implementa su clase abstracta padre.

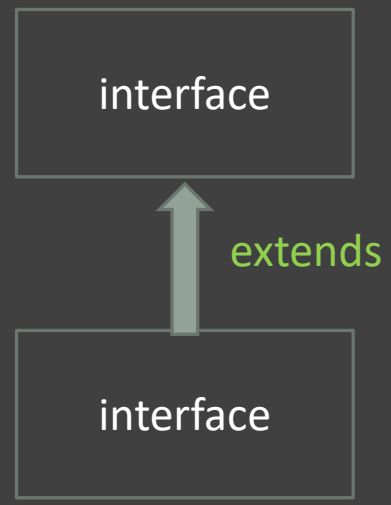
6 - Las variables declaradas en una interface no son variables de instancia. Esto es porque son implícitamente public, final y static, y deben inicializarse. Por lo tanto, son esencialmente constantes.

7 - El cuerpo de una interface puede tener métodos default, abstract y static. Estos métodos son por defecto public, por lo que no es necesario indicar estos modificadores de acceso al declararlos.

Interface tips







Una interface nunca implementa!!!

Implementando una Interface

Para implementar una interface, una clase debe proporcionar cuerpos (implementaciones) para los métodos descritos por la interface. Cada clase es libre de determinar los detalles de su propia implementación. Dos clases pueden implementar la misma interfaz de diferentes maneras:

IDestruible

```
String explosionar();
```

Las clases Castillo y la clase Barco adquieren la obligación de implementar el método explosionar()

Castillo implements IDestruible

```
public String explosionar(){

return "boom!! el castillo ha volado por los aires!!!";
}
```

Barco implements IDestruible

```
public String explosionar(){

return "boom!! el barco ha ido a pique!!!";
}
```

La clase Castillo y la clase Barco han implementado el método de forma diferente

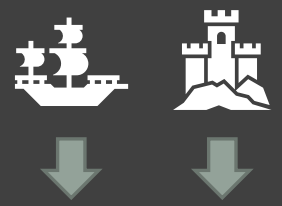
Polimorfismo de Interface

La clase Castillo y la clase Barco no tienen ninguna relación en la estructura de clases. Sin embargo las instancias de ambas clases pueden actuar polimórficamente como IDestruible:

```
IDestruible perlaNegra = new Barco();

IDestruible castilloDelLago = new Castillo();
```

Y pasar sus instancias a métodos que requieran objetos IDestruible:



```
public void hacerAlgo(IDestruible elDestruible){…código sumamente complejo…}
```

Jerarquía entre Interfaces

La jerarquía entre interfaces permite la herencia simple y múltiple. Es decir, tanto la declaración de una clase, como la de una interface pueden incluir la implementación de otras interfaces. Los identificadores de las interfaces se separan por comas:

IUna extends IDos, ITres

IDos

ITres

ICuatro

Castillo implements IUna

Las clases que implementan la interfaz IUna también lo hacen con IDos y ITres

Barco implements ITres

Solo implementa ITres

Caballo extends Animal implements IDos, ICuatro

Hereda de la clase Animal e implementa IDos e ICuatro

Utilización de una Interface como un tipo de dato

Al declarar una interface, se declara un nuevo tipo de referencia.

Pueden emplearse identificadores de interface en cualquier lugar donde se pueda utilizar el identificador de un tipo de dato (o de una clase). El objetivo es garantizar la sustituibilidad por cualquier instancia de una clase que la implemente.

Por ejemplo es totalmente válido declarar una interface vacía:

```
public interface IInventariable {}
```

…para (por ejemplo) poder almacenar objetos de varias clases dentro de (por ejemplo) un ArrayList:

```
private final List<IInventariable> inventarioMercaderias = new ArrayList<>();
```

Estas interfaces son conocidas como Interfaces de Marcado (Marker Interfaces).

Una interface no es una clase pero se considera un tipo en Java y puede ser utilizado como tal.

Interface de marcado

Manzana implements IInventariable

Espada implements IInventariable

Caballo implements IInventariable

IInventariable

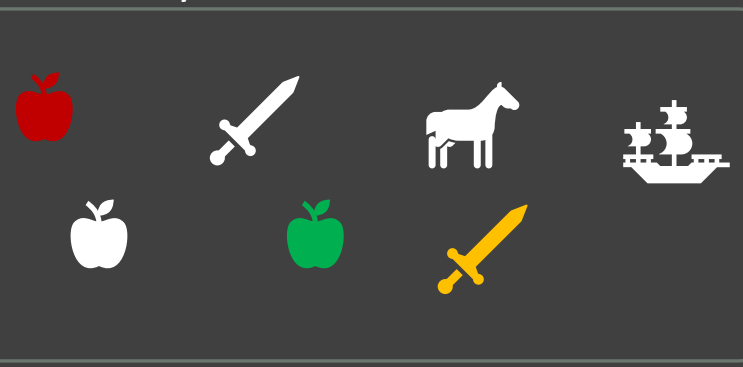
(Vacía)

Interface de marcado

Barco implements IInventariable

Mediante la Interface podemos almacenar objetos, que no tienen que ver nada unos con otros, dentro de un ArrayList

ArrayList<IInventariable>



Variables en Interfaces

Podemos declarar variables en una interface y éstas son implícitamente públicas, estáticas y finales (public, static, y final). A primera vista, podríamos pensar que tendríamos un uso muy limitado para tales variables, pero ocurre lo contrario.

Muchos programas hacen uso de varios valores constantes que describen cosas como el tamaño de un array, diversos límites, valores especiales etc. Estos datos constantes han de ser utilizados por diferentes clases que, a menudo, no comparten líneas de herencia etc. En Java, las variables de interface ofrecen una solución.

```
public interface Constantes{
 //Definiendo 3 constantes
   int MIN=0;
   int MAX=10;
   String MSJERROR="LIMITE ERROR";
}
```

```
public class MiClase
implements Constantes{
}
```

```
public class Principal {
   public static void main(String[] args) {
        MiClase miInstancia = new MiClase();
        System.out.println(miInstancia.MIN);
       System.out.println(miInstancia.MAX);
       System.out.println(miInstancia.MSJERROR);
   }
}
```