Interfaces y clases abstractas

Ya hemos hablado con anterioridad de las **clases abstractas** (capítulo polimorfismo), en este capítulo profundizaremos un poco mas sobre ellas y además hablaremos por primera vez de las **interfaces**.

Interfaces

Cuando alguien te pregunta que es una interface suele decirse que es un contrato, una forma de asegurarse que todo el mundo que implemente una interface va a hacer algo. No nos importa como vayan a hacerlo pero estamos seguros que si una clase implemente una interface entonces va a tener que cumplir el contrato e implementar lo acordado.

Vamos a ver la sintaxis

```
public interface NombreInterface{
    sentencias;
}
```

Y cuando una clase implementa una interface la sintaxis es

```
class NombreClase implements NombreInterface{
    sentencias;
}
```

Una clase puede implementar 0 o varias interfaces al mismo tiempo. En caso de implementar varias interfaces separaremos sus nombres por comas. El hecho de poder implementar varias interfaces al mismo tiempo nos sirve para esquivar la limitación de que una clase solo pueda heredar de una única clase. En caso de que una clase extendise a otra clase y a la vez implementase 1 o varias interfaces la sintaxis sería la siguiente:

```
class NombreClase extends SuperClase implements NombreInterface1, nombreInterface2,...{
    sentencias;
}
```

Las interfaces pueden ser públicas (accesibles desde cualquier lugar) o sin modificador de acceso (accesibles desde el paquete de la interface).

Una interface no puede ser instanciada (si se podría pero es demasiado avanzado para el curso).

Una interface puede contener métodos abstractos y métodos estáticos, ambos se considerarán públicos (aunque no se indique). También pueden añadir métodos por defecto (con el modificador default) pero no los vamos a ver en este curso.

Una interface puede contener atributos que a todos los efectos será una constante, estaremos obligados a darle valor. Los atributos que creemos se considerarán públicos, estáticos y finales por lo que podemos omitir los modificadores.

Una interface puede extender otras inferfaces, lo hará con la siguiente sintaxis

```
public interface NombreInterface extends NombreOtraInterface1, NombreOtraInterface2,...{
    sentencias;
}
```

Si, es correcto extends, no me he equivocado.

Clases abstractas

La sintaxis necesaria para crear una clase abstracta es la siguiente

```
abstract class NombreClase{
    sentencias;
}
```

Las clases abstractas pueden incluir o no **métodos abstractos**. Las clases abstractas no pueden ser instanciadas pero si pueden ser usadas como subclases.

Un método abstracto es un método que está declarado pero no está implementado. En su sintaxis se suprimen las llaves ({}) y tras cerrar el paántesis que contiene los parámentros se pone un punto y coma.

```
public abstract void montar();
```

Si una clase contiene un método abstracto tiene que ser obligatoriamente una clase abstracta.

Cuando una clase hereda de una clase abstracta y la superclase contiene un método abstracto estamos obligados a implementarlo.

Diferencias entre clases abstractas e interfaces

Las clases abstractas y las interfaces son similares. No puedes instanciarlas y pueden contener métodos implementados o no.

En las clases abstractas puedes disponer de atributos sin que estos sean constantes (como ocurre en las interfaces) y además puedes crear métodos públicos, protegidos o privados (en las interfaces todos eran públicos).

Solo puedes extender una clase (abstracta o no) mientas que puedes implementar cualquier número de interfaces.

Entonces, ¿cuándo elijo una u otra?

- Clases abstractas cuando se cunpla alguna de las condiciones siguientes:
  - Quieres compartir código entre muchas clases relacionadas
  - Esperas que las clases que extiendan tu clase abstracta tengan en común métodos o campos
  - Quieres disponer de atributos no estáticos o no finales. Esto te habilita para definir métodos que puedan acceder y modificar el estado del objeto al que pertenecen
- Interfaces cuando se cumpla alguna de las condiciones posteriores:
  - Esperas que clases sin relación entre si implementen tu interface.
  - Quieres un comportamiento específico sin importante la implementación
  - Quieres disponer de herencia múltiple.

Si deseas ampliar la información, en la documentación oficial de Java existe un tutorial específico sobre interfaces y herencia (en inglés).

En el apartado "código utilizado en los ejemplos" dejo un proyecto en el cual hago uso de interfaces y clases abstractas.

Revision #1
Created 2022-02-01 11:11:21 CET by Equipo CATEDU
Updated 2022-02-01 11:11:21 CET by Equipo CATEDU