INTERFACES

Concepto de interface

Una ***interface*** es un conjunto de ***declaraciones de métodos*** (sin ***definición***). También puede definir ***constantes***, que son implícitamente ***public***, ***static*** y ***final***, y deben siempre inicializarse en la declaración. Estos métodos definen un *tipo de conducta*. Todas las clases que implementan una determinada ***interface*** están obligadas a proporcionar una definición de los métodos de la ***interface***, y en ese sentido adquieren una ***conducta*** o ***modo de funcionamiento***.

Una ***clase*** puede ***implementar*** una o varias ***interfaces***. Para indicar que una clase implementa una o más interfaces se ponen los nombres de las ***interfaces***, separados por comas, detrás de la palabra ***implements***, que a su vez va siempre a la derecha del nombre de la clase o del nombre de la super-clase en el caso de herencia. Por ejemplo,

```
public class CirculoGrafico extends Circulo implements Dibujable, Cloneable {

...

}
```

¿Qué diferencia hay entre una ***interface*** y una ***clase abstract***? Ambas tienen en común que pueden contener varias declaraciones de métodos (la clase ***abstract*** puede además definirlos). A pesar de esta semejanza, que hace que en algunas ocasiones se pueda sustituir una por otra, existen también algunas ***diferencias importantes***:

1. Una clase no puede heredar de dos clases ***abstract***, pero sí puede heredar de una clase ***abstract*** e implementar una ***interface***, o bien implementar dos o más ***interfaces***.
2. Una clase no puede heredar métodos -definidos- de una ***interface***, aunque sí ***constantes***.
3. Las ***interfaces*** permiten mucha más flexibilidad para conseguir que dos clases tengan el mismo comportamiento, inpendientemente de su situación en la jerarquía de clases de ***Java***.
4. Las ***interfaces*** permiten "publicar" el comportamiento de una clase desvelando un mínimo de información.
5. Las ***interfaces*** tienen una ***jerarquía*** propia, independiente y más flexible que la de las clases, ya que tienen permitida la ***herencia múltiple***.
6. De cara al ***polimorfismo***, las ***referencias*** de un tipo ***interface*** se pueden utilizar de modo similar a las clases ***abstract***.

Definición de interfaces

Una ***interface*** se define de un modo muy similar a las clases. A modo de ejemplo se da la definición de la interface ***Dibujable:***

```
// fichero Dibujable.java
import java.awt.Graphics;
public interface Dibujable {
public void setPosicion(double x, double y);
public void dibujar(Graphics dw);
}
```

Cada ***interface public*** debe ser definida en un fichero ****.java*** con el mismo nombre de la ***interface***. Los ***nombres de las interfaces*** suelen comenzar también con ***mayúscula***.

Las ***interfaces*** no admiten más que los modificadores de acceso ***public*** y ***package***. Si la ***interface*** no es ***public*** no será accesible desde fuera del ***package*** (tendrá la accesibilidad por defecto, que es ***package***). Los métodos declarados en una ***interface*** son siempre ***public*** y ***abstract***, de modo implícito.

Herencia en interfaces

Entre las ***interfaces*** existe una ***jerarquía*** (independiente de la de las clases) que permite ***herencia simple y múltiple***. Cuando una ***interface*** deriva de otra, incluye todas sus constantes y declaraciones de métodos.

Una ***interface*** puede derivar de varias ***interfaces***. Para la herencia de ***interfaces*** se utiliza asimismo la palabra ***extends***, seguida por el nombre de las ***interfaces*** de las que deriva, separadas por comas.

Una ***interface*** puede ocultar una constante definida en una ***super-interface*** definiendo otra constante con el mismo nombre. De la misma forma puede ocultar, re-declarándolo de nuevo, la declaración de un método heredado de una ***super-interface***.

Las ***interfaces*** no deberían ser modificadas más que en caso de extrema necesidad. Si se modifican, por ejemplo añadiendo alguna nueva declaración de un método, las clases que hayan implementado dicha ***interface*** dejarán de funcionar, a menos que implementen el nuevo método.

Utilización de interfaces

Las ***constantes*** definidas en una ***interface*** se pueden utilizar en cualquier clase (aunque no implemente la ***interface***) precediéndolas del nombre de la ***interface***, como por ejemplo (suponiendo que PI hubiera sido definida en ***Dibujable***):

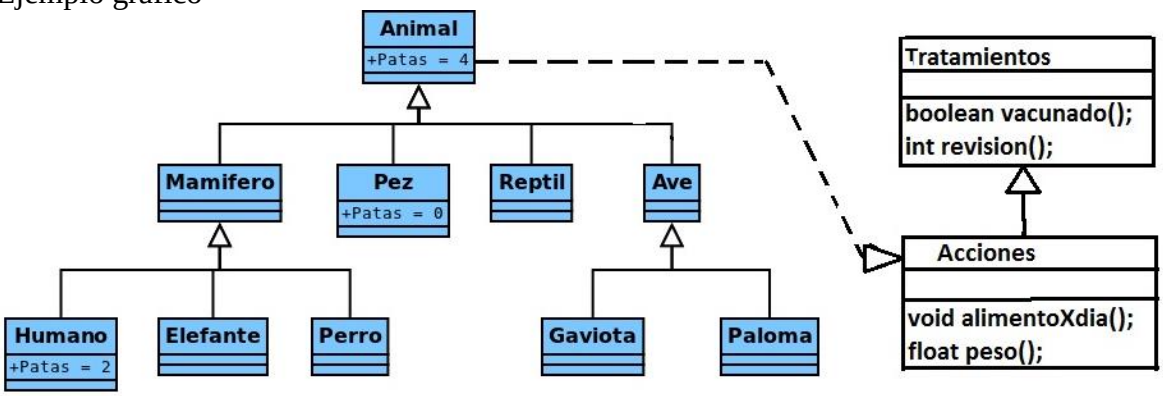
```
area = 2.0*Dibujable.PI*r;
```

Sin embargo, en las clases que ***implementan*** la ***interface*** las constantes se pueden utilizar directamente, como si fueran constantes de la clase. A veces se crean interfaces para agrupar constantes simbólicas relacionadas (en este sentido pueden en parte suplir las variables ***enum*** de C/C++).

De cara al ***polimorfismo***, el nombre de una ***interface*** se puede utilizar como un ***nuevo tipo de referencia***. En este sentido, el nombre de una interface puede ser utilizado en lugar del nombre de cualquier clase que la implemente, aunque su uso estará restringido a los métodos de la ***interface***.

Un objeto de ese tipo puede también ser utilizado como valor de retorno o como argumento de un método.

Ejemplo gráfico



➔ public abstract class Animal implements Tratamientos, Acciones { …