Herencia

La herencia define una relación entre clases en la cual una clase posee características (métodos y propiedades) que proceden de otra.

Esto permite estructura de forma muy atractiva los programas y reutilizar código de forma más eficiente. Es decir genera relaciones entre clases del tipo **es u n / como** ...

A la clase que posee las características a heredar se la llama **superclase** y la clase que las hereda se llama **subclase**. Una subclase puede incluso ser superclase en otra relación de herencia.

También se emplea habitualmente los términos **madre** para referirnos a una superclase e **hija** para una subclase (e incluso **abuela** y **nieta**).

Si diseñáramos una clase que represente animales en general, podría tener como métodos: ***respirar***, ***reproducirse*** o ***crecer*** por ejemplo. Si después quisiéramos una clase para representar leopardos, tendrían métodos como ***correr*** o ***rugir***. Por ello lo que hay que hacer no es definir de nuevo esos métodos, sino simplemente indicar que el leopardo **es un** animal.

Sin herencia la programación orientada o objetos no sería tal, incumpliría uno de sus pilares básicos.

La herencia facilita enormemente el trabajo del programador o programadora porque permite crear clases estándar y a partir de ellas crear nuestras propias clases personales. Esto es más cómodo que tener que crear todas las clases desde cero. Además simplifica la reutilización del código y la creación de métodos y elementos más capaces e independientes.

Para que una clase herede las características de otra hay que utilizar la palabra clave **extends** tras el nombre de la clase. A esta palabra le sigue el nombre de la clase cuyas características se heredarán. **En Java sólo se puede tener herencia de una clase.**