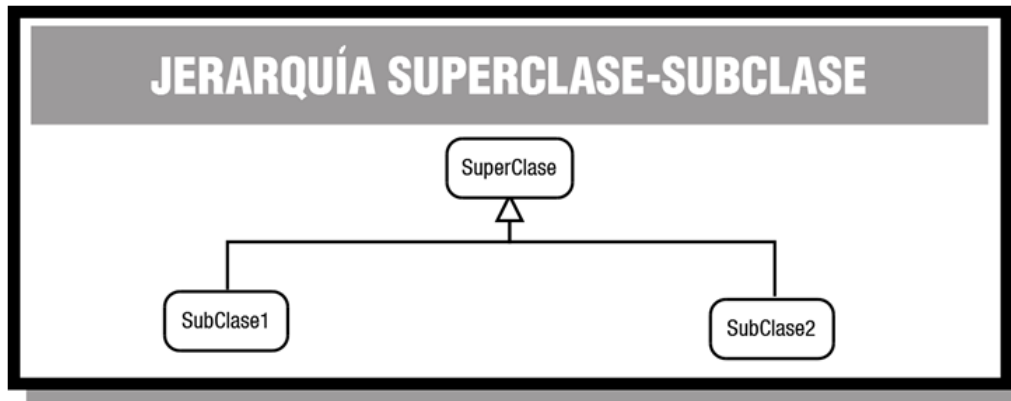


7.C. Herencia.

1. Herencia.

La **herencia** es el mecanismo que permite definir una nueva clase a partir de otra, pudiendo añadir nuevas características, sin tener que volver a escribir todo el código de la clase base.



La clase de la que se hereda suele ser llamada **clase base**, **clase padre** o **superclase**. A la clase que hereda se le suele llamar **clase hija**, **clase derivada** o **subclase**.

Una clase derivada puede ser a su vez **clase padre** de otra que herede de ella y así sucesivamente dando lugar a una **jerarquía de clases**, excepto aquellas que estén en la parte de arriba de la jerarquía (sólo serán **clases padre**) o en la parte de abajo (sólo serán **clases hijas**).

Una **clase hija** no tiene acceso a los miembros **privados** de su **clase padre**, tan solo a los **públicos** (como cualquier parte del código tendría) y los **protegidos** (a los que sólo tienen acceso las **clases derivadas** y las del mismo **paquete**). Aquellos miembros que sean privados en la clase base también habrán sido heredados, pero el acceso a ellos estará restringido al propio funcionamiento de la **superclase** y sólo se podrá acceder a ellos si la **superclase** ha dejado algún medio indirecto para hacerlo (por ejemplo a través de algún método).

Todos los miembros de la **superclase**, tanto atributos como métodos, son heredados por la subclase. Algunos de estos miembros heredados podrán ser **redefinidos** o **sobrescritos** (**override**) y también podrán añadirse nuevos miembros. De alguna manera podría decirse que estás “ampliando” la **clase base** con características adicionales o modificando algunas de ellas (proceso de **especialización**).

Una clase derivada extiende la funcionalidad de la clase base sin tener que volver a escribir el código de la clase base.

Autoevaluación

Una clase derivada hereda todos los miembros de su clase base, pudiendo acceder a cualquiera de ellos en cualquier momento. ¿Verdadero o Falso?

- ☐ Verdadero
☐ Falso