

## 7.D. Clases abstractas.

### 1. Clases abstractas.

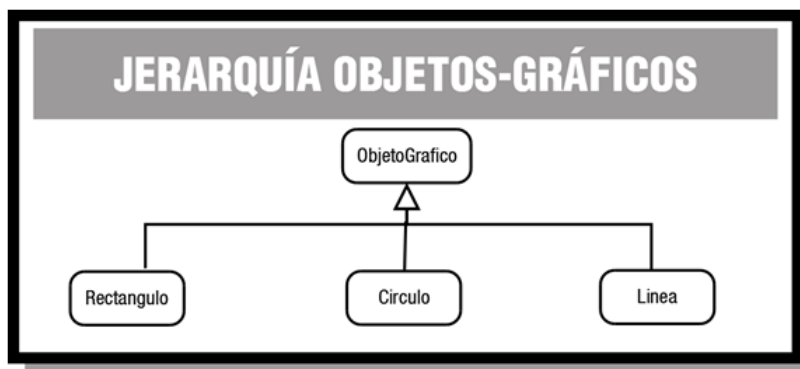
En determinadas ocasiones, es posible que necesites definir una clase que represente un concepto lo suficientemente abstracto como para que nunca vayan a existir instancias de ella (objetos). ¿Tendría eso sentido? ¿Qué utilidad podría tener?

Imagina una aplicación para un **centro educativo** que utilice las clases de ejemplo **Alumno** y **Profesor**, ambas subclases de **Persona**. Es más que probable que esa aplicación nunca llegue a necesitar objetos de la clase **Persona**, pues serían demasiado genéricos como para poder ser utilizados (no contendrían suficiente información específica). Podrías llegar entonces a la conclusión de que la clase **Persona** ha resultado de utilidad como **clase base** para construir otras clases que hereden de ella, pero **no** como **una clase instanciable** de la cual vayan a existir objetos. **A este tipo de clases se les llama clases abstractas.**

**En algunos casos puede resultar útil disponer de clases que nunca serán instanciadas, sino que proporcionan un marco o modelo a seguir por sus clases derivadas dentro de una jerarquía de herencia. Son las clases abstractas.**

La posibilidad de declarar **clases abstractas** es una de las características más útiles de los **lenguajes orientados a objetos**, pues permiten dar unas líneas generales de cómo es una clase sin tener que implementar todos sus métodos o implementando solamente algunos de ellos. Esto resulta especialmente útil cuando las distintas **clases derivadas** deban proporcionar los mismos métodos indicados en la clase **base abstracta**, pero su **implementación sea específica** para cada **subclase**.

Imagina que estás trabajando en un entorno de **manipulación de objetos gráficos** y necesitas trabajar con **líneas, círculos, rectángulos**, etc. Estos objetos tendrán en común algunos atributos que representen su estado (**ubicación, color del contorno, color de relleno**, etc.) y algunos métodos que modelen su comportamiento (**dibujar, rellenar con un color, escalar, desplazar, rotar**, etc.). Algunos de ellos serán comunes para todos ellos (por ejemplo la **ubicación** o el **desplazamiento**) y sin embargo otros (como por ejemplo **dibujar**) necesitarán una implementación específica dependiendo del tipo de objeto. Pero, en cualquier caso, todos ellos necesitan esos métodos (tanto un **círculo** como un **rectángulo** necesitan el método **dibujar**, aunque se lleven a cabo de manera diferente). En este caso resultaría muy útil disponer de una **clase abstracta objeto gráfico** donde se definirían las **líneas generales** (algunos atributos concretos comunes, algunos métodos concretos comunes implementados y algunos métodos genéricos comunes sin implementar) de un objeto gráfico y más adelante, según se vayan definiendo **clases especializadas** (**líneas, círculos, rectángulos**), se irán concretando en cada **subclase** aquellos métodos que se dejaron sin implementar en la **clase abstracta**.



#### Autoevaluación

Una clase abstracta no podrá ser nunca instanciada. ¿Verdadero o Falso?

- ☒ Verdadero  
☐ Falso