Tengo conocimientos con varios lenguajes, esto me lleva a tener problemas al recordar ciertas cosas, puesto que cuando uso uno durante mucho tiempo, el resto tiendo a olvidarlo, pero considero que hay ciertos **conceptos** que son comunes a los distintos lenguajes.

**Función recursiva**: Aquella función que se llama así misma resolverse. El ejemplo que se pone comúnmente es el cálculo del factorial de un número n.

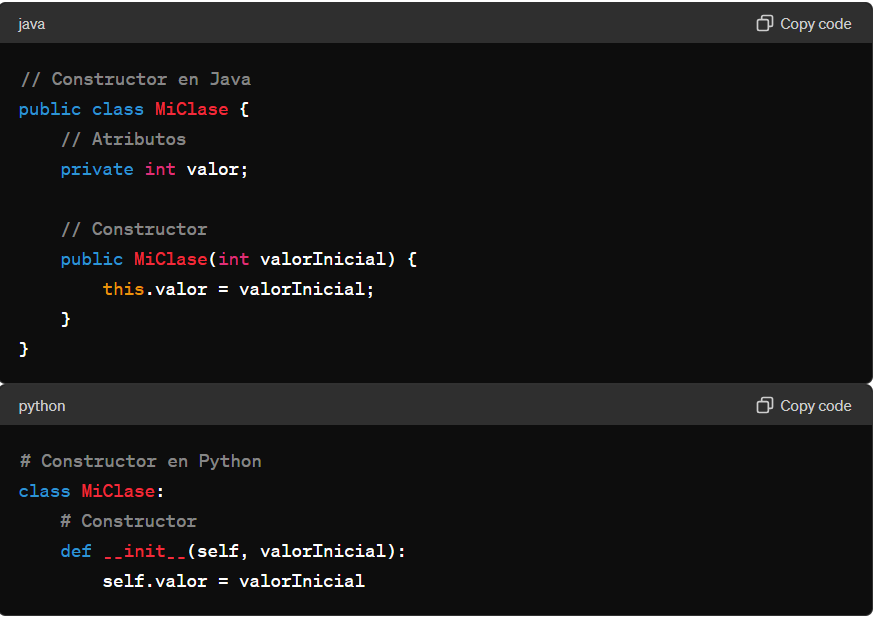
fact(n) = n \* fact(n-1), con n>=2 y fact(1) = 1

Con lo cual, para averiguar el factorial de un número n, tendríamos que multiplicar n por el factorial de n-1, y para el factorial de ese n-1, el mismo proceso hasta llegar al factorial de 1 que siempre es 1.

Con lo cual, haciendo una función recursiva, resolveríamos esto con una única función que va a estar entrando en si misma hasta llegar al Fact(1) = 1. Y una vez llegado ahí, volverá para atrás para resolver el factorial del número que sea.

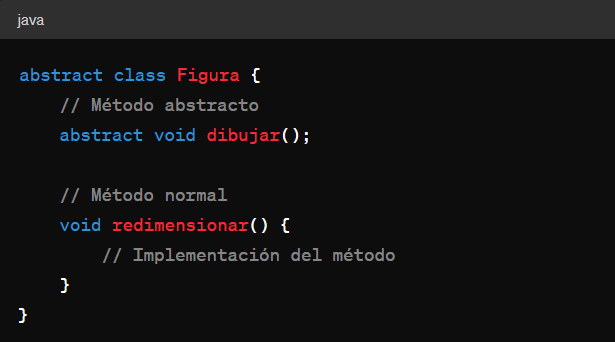
**Constructor**: En programación, tenemos clases, las cuales tienen objetos y los objetos tienen atributos. El objetivo principal del constructor es inicializar los atributos de los objetos cuando se crea una instancia de la clase.

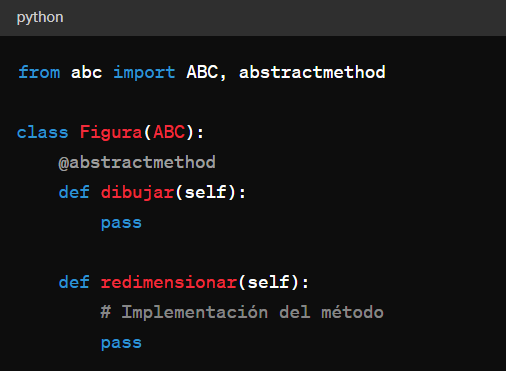
* **Nombre**. Normalmente el nombre del constructor es el mismo que el de la clase a la que pertenece.
* **Invocación automática**. El constructor se crea invoca automáticamente cuando se crea un nuevo objeto de esa clase. No es necesario llamarlo explícitamente.
* **Sintaxis**.



**Clase abstracta**: Es una clase que no se puede instanciar directamente. Sino que se utiliza como una plantilla para otras clases. Intentar crear un objeto de esta dará un error.

* **Métodos abstractos**. Una clase abstracta puede contener métodos abstractos. Un método abstracto, es un método sin implementación.
* **Métodos normales**. También puede tener métodos normales.
* **Herencia**. Las clases derivadas(subclases) de una abstracta deben de implementar todos los métodos abstractos definidos en la clase abstracta, a menos que la subclase también sea abstracta.
* **Uso**. Se utilizan cuando se quiere definir una interfaz común para un conjunto de clases relacionadas.
* **Sintaxis**.





**Interfaz**: Nace para resolver el problema de herencia múltiple (no puedes heredar de más de una clase) la cuál no es posible en java ni en C#. De varias interfaces sí que se puede heredar.

Es parecido a una clase abstracta con la diferencia de que no tiene implementados los métodos, simplemente los tiene definidos. Y las clases que la utilicen tienen que seguir las normas de esta.

Se compone de métodos abstractos y constantes, donde se especifica que se debe de hacer, pero no como. Son las clases hijas las que definen el comportamiento de los métodos de esta interfaz.

**Polimorfismo**: Métodos o atributos iguales que están en varias clases, estas a su vez vienen de un miso padre, y pueden tener distintas funcionalidades. Por ejemplo si una clase padre Personaje con un método atacar(). De esta clase heredan dos clases, guerrero() y mago(). Pues el método heredado atacar() puede ser distinto en estas dos clases.

**Encapsulación**: Es cuando definimos atributos o métodos privados a una clase. Por ejemplo un personaje que tiene su constructor (para instanciar el objeto), turno (), atacar() y moverse() y “encapsulamos” atacar y moverse, para que estos no se puedan modificar.