Clase Abstracta

Una clase Abstracta, es una clase de la que **no podrá haber instancias**.

No podrá haber Objetos del tipo de esta clase

Por ejemplo, si la Clase CuentaBancaria fuese abstracta, no podrá haber Objetos del tipo CuentaBancaria.

...¿Entonces para qué sirve?

Clase Abstracta

Si diseñamos una herencia, y decimos que **CajaDeAhorro** y **CuentaCorriente**, heredan de **CuentaBancaria**, entonces sí podremos tener instancias. Pero de las hijas.

Podremos tener objetos de tipo **CajaDeAhorro**, y podremos tener objetos de tipo **CuentaCorriente**.

Las Clases Abstractas se utilizarán en conjunto con Herencia.

Clase Abstracta

Veamos un ejemplo en donde **no** nos sirve hacer una Clase Abstracta: Supongamos que **CapitanPirata**, hereda de **Pirata**.

Un **CapitanPirata**, es un **Pirata** como los demás, que además puede **darOrdenes()**.

En este caso, claramente me interesa poder instanciar **CapitanPirata**, y también poder instanciar **Pirata**, para construir la tripulación.

Ninguna de estas clases será Abstracta.

Clase Abstracta

Ahora pensemos el caso en que **Manzana hereda de Fruta.**

Está claro que me interesa poder instanciar **Manzanas**.
Pero... ¿Me interesa poder instanciar **Fruta**? ¿Tiene sentido?

Si a alguien le dijera “dame **Fruta**”. ¿Cómo reaccionaría?.

Seguramente me de una **manzana**, o una **mandarina**. Que son hijas de **Fruta**. Nunca me dará “**Fruta**”.
“**Fruta**” no existe. “**Manzana**”, “**Pera**”, “**Naranja**” si existen.

Entonces en este caso, a la clase “Fruta” la declaro como Abstracta.

Clase Abstracta

La clase **Fruta** será una clase con nombre, atributos y métodos, pero que no podrá ser instanciada.

Los métodos podrían tener alguna implementación, y sus “hijos” decidirán si la utilizan, la descartan y reemplazan por su propia implementación (Override), o la complementan, utilizándola y agregándole más código.

Clase Abstracta

Otro ejemplo: si tuviera la clase **AnimalDomestico**, con uno de sus métodos llamado **emitirSonido()**, que devolviera el string **“Estoy emitiendo sonido”**.

La clase **Perro**, que **hereda de AnimalDomestico**, tendrá **tres** opciones con respecto a este método.

1. **No implementar** emitirSonido() → entonces unPerro->emitirSonido() **devolvería: “Estoy emitiendo sonido”** (utilizando la implementación que hereda de su padre)

Clase Abstracta

2. **Implementar** emitirSonido() reescribiendolo completamente devolviendo “Guau” → entonces unPerro->emitirSonido(), **devolvería:** “Guau”.

3. **Usar la implementación pero complementándola** Agregándole un “Guauuuu!” por ejemplo → entonces unPerro->emitirSonido(), **devolvería:** “Estoy emitiendo sonido. Guauuuu!”

UML - Clase Abstracta

Para diagramar una clase Abstracta con el diagrama de clases UML, la diagramamos como una clase común, y le agregamos los símbolos **<<** y **>>** antes y después de su nombre.
Así:
<<Mascota>>, **<<CuentaBancaria>>**, **<<Fruta>>**

UML - Herencia

un Perro

hereda de (es)

un AnimalDomestico

Esta es una relación de herencia, en donde la clase padre es abstracta.
Se diagrama con una **línea sólida** y punta de flecha **vacía** desde el hijo hacia el padre, igual que en el caso de herencia.