

# **DIAGRAMAS DE CLASES**

**SABELA SOBRINO SAN MARTÍN**

# INTRODUCCIÓN



# PROPÓSITO Y FUNCIÓN

- Componentes estáticos de un sistemas (clases, módulos, atributos, etc.)
- Expresan unidades de código y cómo se organizan.
- Documentación del proyecto.

# ELEMENTOS DE DIAGRAMAS DE CLASES












# MÉTODOS Y ATRIBUTOS DE UNA CLASE

## Atributos

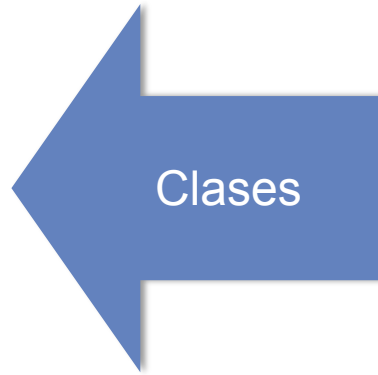
- Nombre
- Tipo
- Visibilidad

## Métodos

- Nombre
- Parámetros
- Tipo devuelto
- Visibilidad
- (Descripción)

 NombreClase	
	-Object atributo_privado
	+Object Atributo_publico
	#Object Atributo_protegido
	~Object Atributo_paquete
<hr/>	
	-void Metodo_privado()
	+void Metodo_publico()
	#void Metodo_protegido()
	~void Metodo_paquete()

# RECOMENDACIONES



# EXEMPLO

- ***Analista:*** "Adestrador, de que trata o xogo? "
- ***Adestrador:*** "Consiste en botar o balón a través dun aro, coñecido como cesto, e facer unha maior puntuación que o opoñente. Cada equipo consta de cinco xogadores: dous defensas, dous dianteiros e un central. Cada equipo leva o balón ao cesto do equipo opoñente co obxectivo de facer que o balón sexa encestando. "
- ***Analista:*** "Como se fai para levar o balón ao outro cesto? "
- ***Adestrador:*** "Mediante pases e dribles. Pero o equipo terá que encestar antes de que remate o lapso para tirar. "
- ***Analista:*** "O lapso para tirar?"

...

# EXEMPLO

 <b>Balon</b>
<i>Attributes</i> private short diametro private short volumen
<i>Operations</i> public Balon( ) public short getDiametro( ) public void setDiametro( short val ) public short getVolumen( ) public void setVolumen( short val ) public void driblar( ) public void tirar( ) public void pasar( ) public void avanzar( )

 <b>Cesto</b>
<i>Attributes</i>
<i>Operations</i> public Cesto( )



# ASOCIACIÓN

- Representa dos clases que están relacionadas entre sí.
- Además, se indica la navegabilidad de la clase origen a la de destino y la cardinalidad de la clase destino de la asociación.



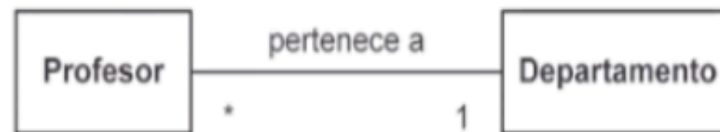
# CARDINALIDAD

1	0..1
N..M	*
0..*	1..*

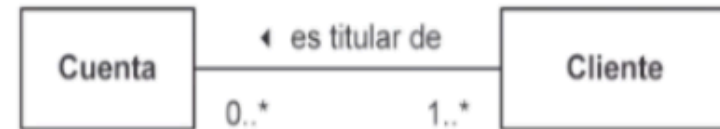
# CARDINALIDAD



Todo departamento tiene un director. Un profesor puede dirigir un departamento.



Todo profesor pertenece a un departamento. A un departamento pueden pertenecer varios profesores.

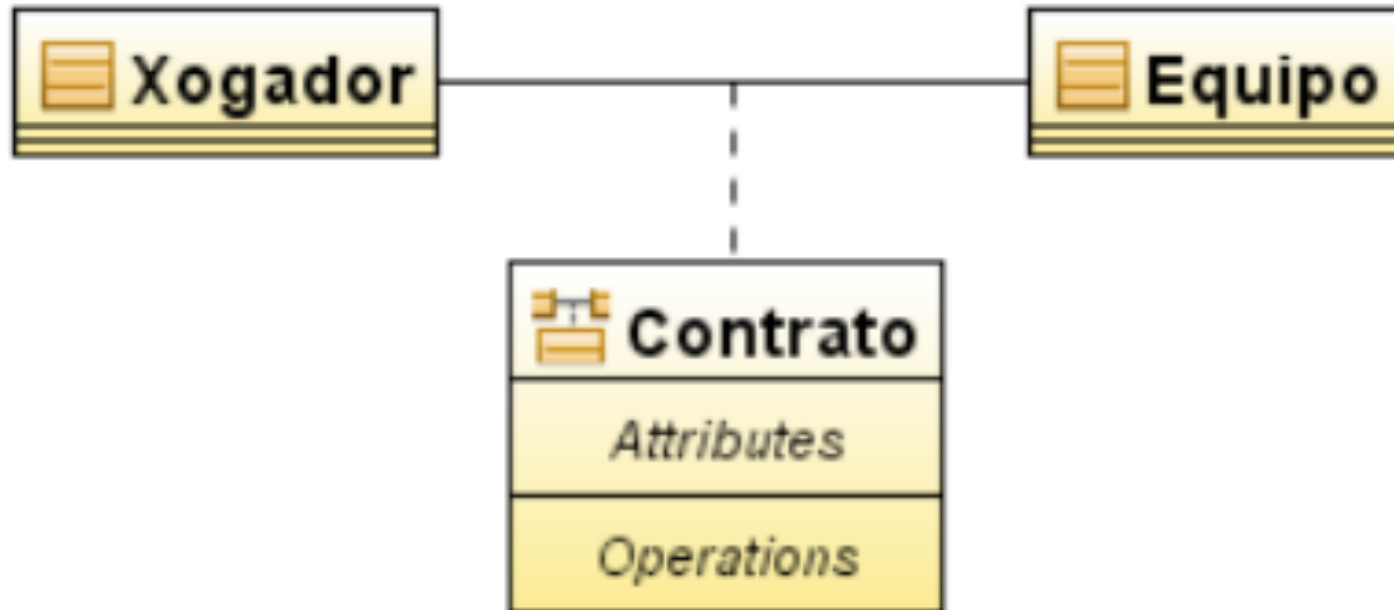


Un cliente puede o no ser titular de una cuenta. Una cuenta debe tener un titular como mínimo



Relación involutiva: La misma clase se encuentra en ambos extremos de una relación.

# RESTRICCIONES



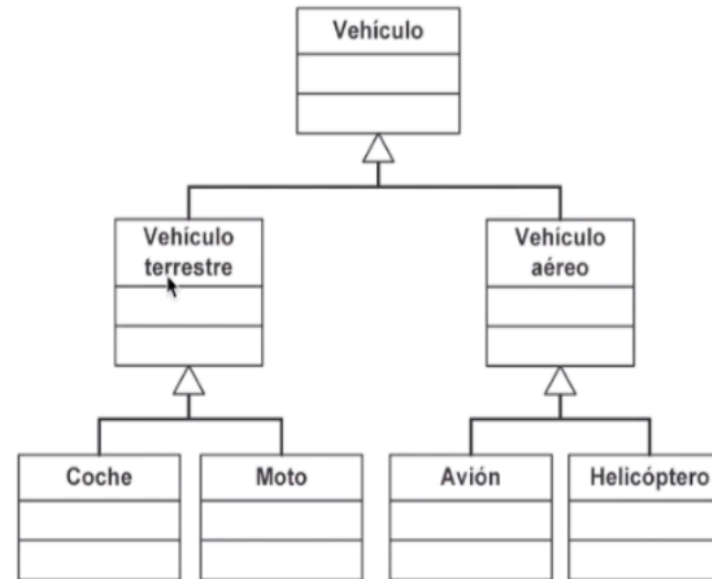
# GENERALIZACIÓN/ HERENCIA

## Simple

- Una clase derivada puede tener solo una clase base heredando sus miembros.

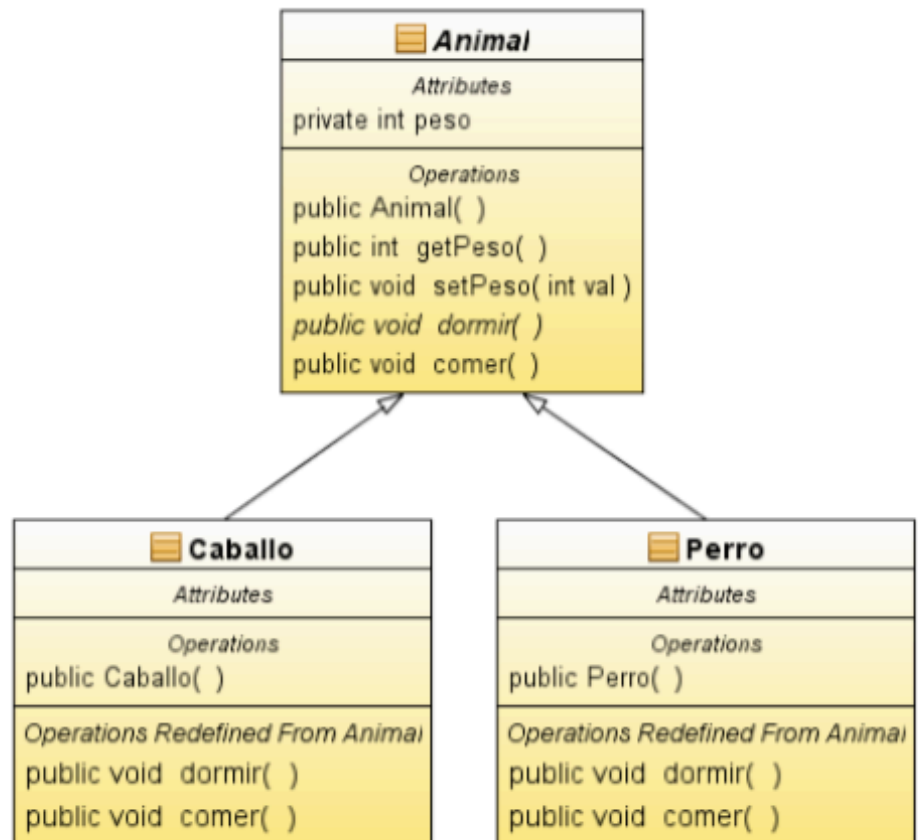
## Múltiple

- Una clase derivada puede tener más de una clase base heredando los miembros de todas.



# CLASES ABSTRACTAS

- No implementa métodos.
- Subclases que defina



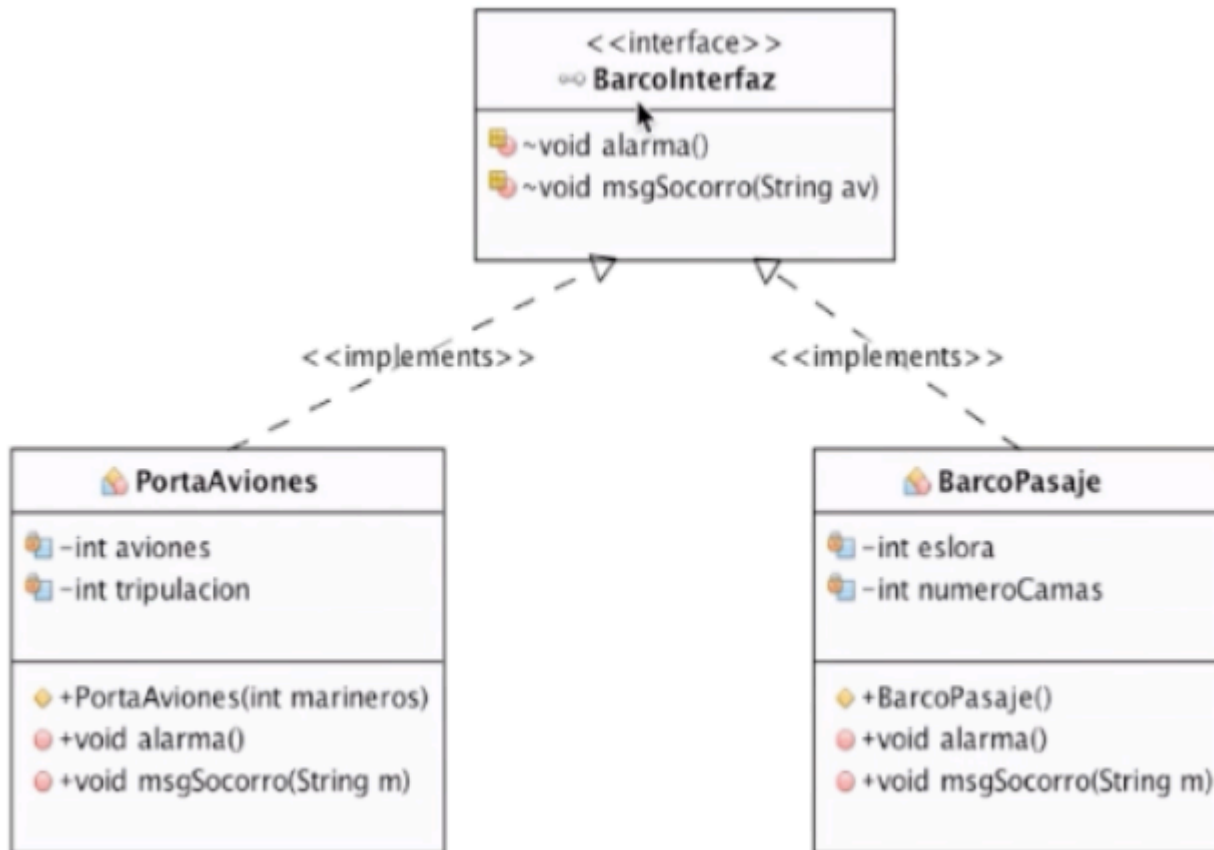
# DEPENDENCIA

- Una clase necesita de otra para su cometido.
- Desde la clase dependiente a la utilizada.



# INTERFAZ

- Clase abstracta pura.
- Declara la forma de una clase





# AGREGACIÓN/ COMPOSICIÓN

## Agregación

- Relación todo-parte (pertenece a, es un, es parte de).
- Los elementos pueden existir por sí mismos sin contenedor.

## Composición

- La parte está relacionada, como máximo, con un todo.
- Si se elimina el todo se eliminan sus partes.

# AGREGACIÓN/ COMPOSICIÓN

