



POO/OOP

Programación Orientado a Objetos

Object-Oriented Programming



Vocabulario



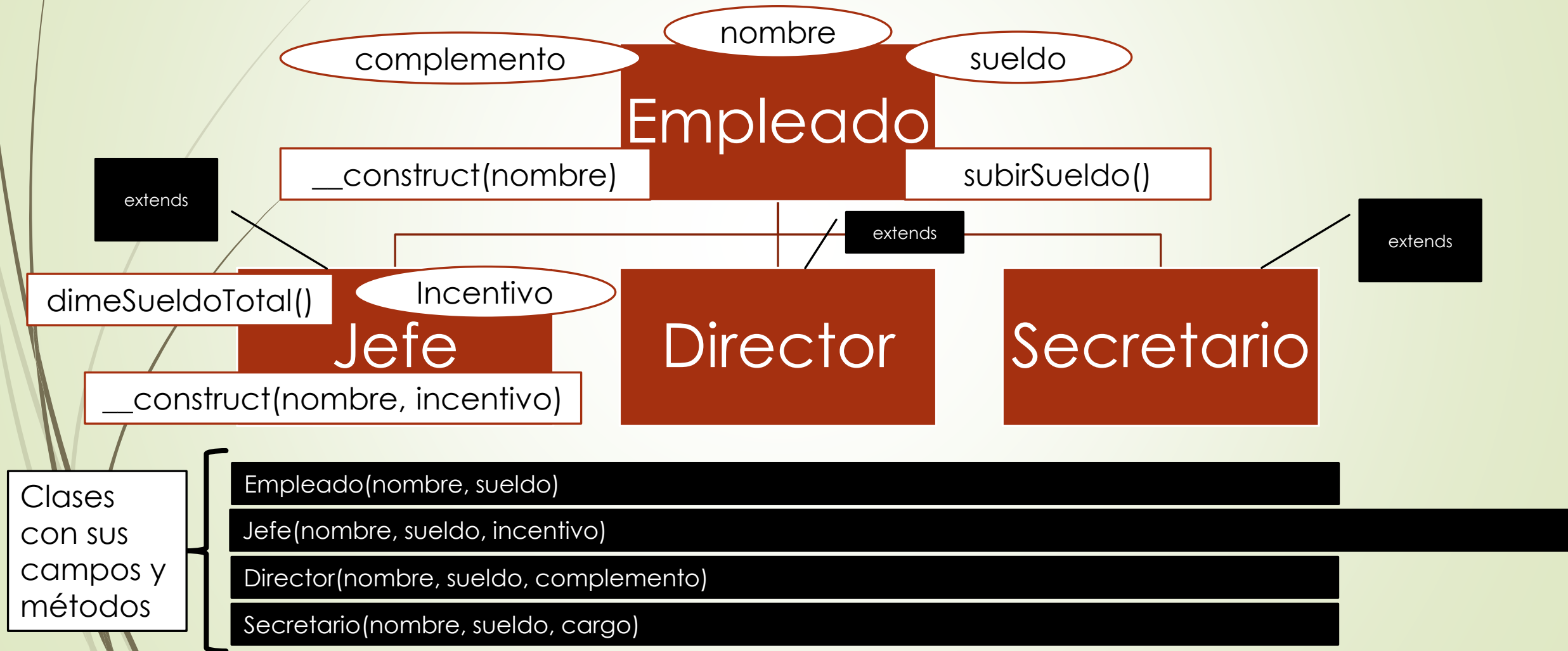
- Clase, objeto
- Ejemplar de clase, instancia de clase, ejemplarizar una clase, instanciar una clase
- Modularización
- Encapsulamiento / encapsulación
- Herencia
 - parent, isinstance
- Polimorfismo

Herencia

- Es posible definir una clase a partir de una clase existente
- La clase que se toma como referencia vendría a ser la clase padre y recibe varios nombres
 - Superclase, **clase base**, clase padre
- La clase que “hereda” estas propiedades y métodos, es decir, la clase “hijo” recibe nombres como
 - Subclase, **clase derivada**
- La nueva clase **posee implícitamente los atributos o métodos de la clase derivada**, aunque **pueden ser sobrescritas**.
- Si la clase derivada no tiene método constructor automáticamente se llama al método constructor del padre
- Para establecer una clase derivada o clase base se utiliza la palabra extends (extender)
- Para invocar al método constructor de la clase base desde la subclase se utiliza
 - `parent::__construct()`

Herencia

En la herencia las **clases derivadas** heredan todos las propiedades y métodos de la **clase base**, **a no ser que la propiedad o método se cree en la clase hija que en tal caso la sobre-escribe.**



Herencia

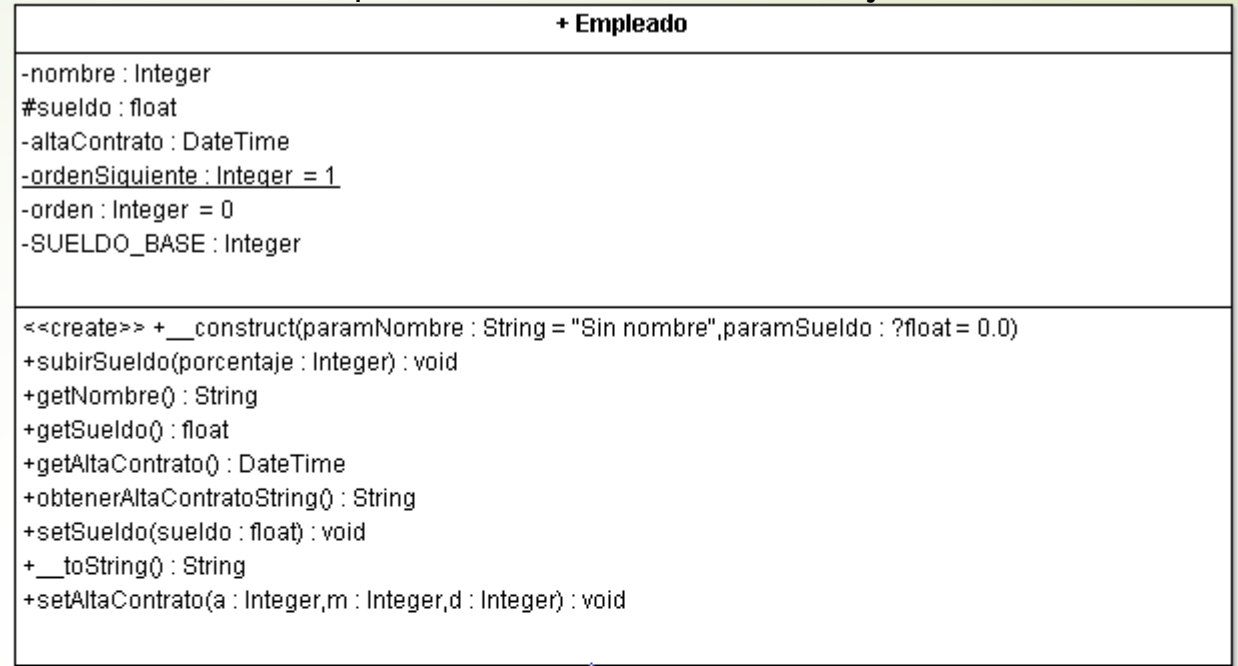
¿Cómo diferenciamos la clase derivada de la clase base?

Regla “es siempre”

Un Jefe “es siempre” un Empleado

Un Empleado “es siempre” un Jefe

- Queremos crear empleados que son jefes
- Incentivo que sólo lo tendrán los jefes

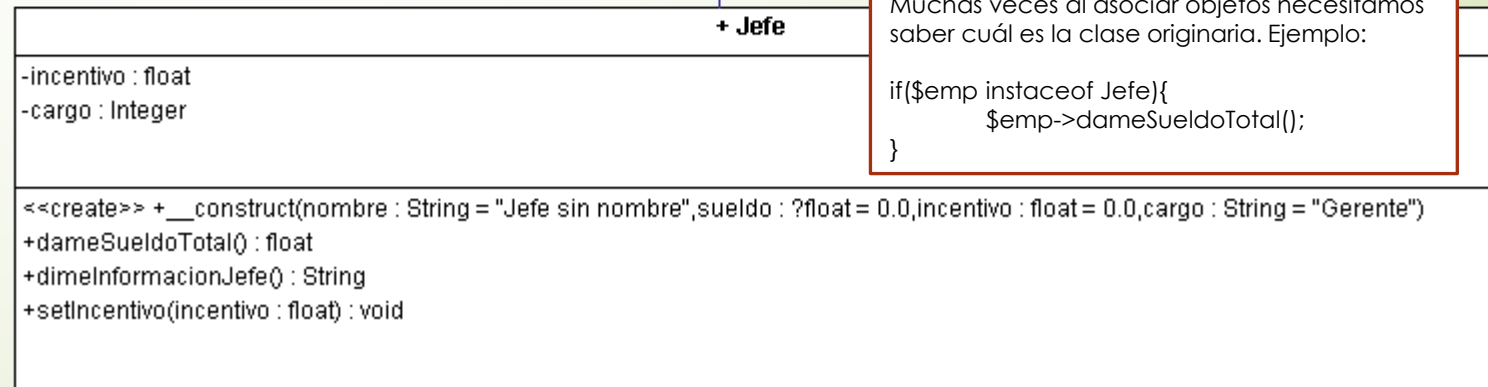


es siempre

Instanceof

Muchas veces al asociar objetos necesitamos saber cuál es la clase originaria. Ejemplo:

```
if($emp instanceof Jefe){
    $emp->dameSueldoTotal();
}
```



Ejercicios1

- De las **clases** **Empleado** y **Jefe** crea un setter que permita modificar la fecha de alta del contrato tanto de los jefes como de los empleados
 - `setAltaContrato()`

Ejercicio 2

- Se desea modelizar en una empresa de reparaciones de coches a los mismos teniendo en cuenta:
 - Número de ruedas (4 por defecto), largo (2000 cm por defecto) y ancho (300 cm por defecto) del vehículo en centímetros
 - Motor (número de caballos). Por defecto 1600
 - Marca
 - Peso de la plataforma del coche (chasis) y peso total del vehículo en Kg. Por defecto 500
 - Si lleva asientos de cuero, si lleva climatizador
 - Se podrán dar un estado inicial a todos estos parámetros y poder obtener la información de las mismas
- Habrà que tener un método que:
 - Crear por lo menos un método constructor que cargue la marca
 - Getter que devuelva el color del coche, marca del coche
 - Métodos que permitan por un lado especificar que el coche tenga asientos de cuero y obtener información de ello con mensajes como:
 - El coche dispone de asientos de cuero
 - El coche dispone de asientos de serie
 - Otros métodos parecidos al caso anterior pero con el climatizador los mensajes a mostrar son
 - El vehículo incorpora climatizador
 - El vehículo lleva aire acondicionado
 - Otro método que de información general del coche. El mensaje resultante sería algo como siguientes:
"La plataforma del coche tiene 4 ruedas. Mide 20 metros con un ancho de 300 centímetros y cuyo peso de plataforma es de 500 Kg."
 - Habrà que también calcular el peso total del vehículo teniendo en cuenta que el climatizador supone un peso extra de 20 kg y si tiene asientos de cuero de 10 kg
- Además de los vehículos ordinarios existirá otra clase de vehículo que serán las furgonetas que además de contener los métodos y propiedades anteriormente definidas también contendrá otros dos campos y métodos:
 - Número de asientos
 - Capacidad extra de carga
 - dimeDatosParticularesFurgoneta()