Introducción a POO

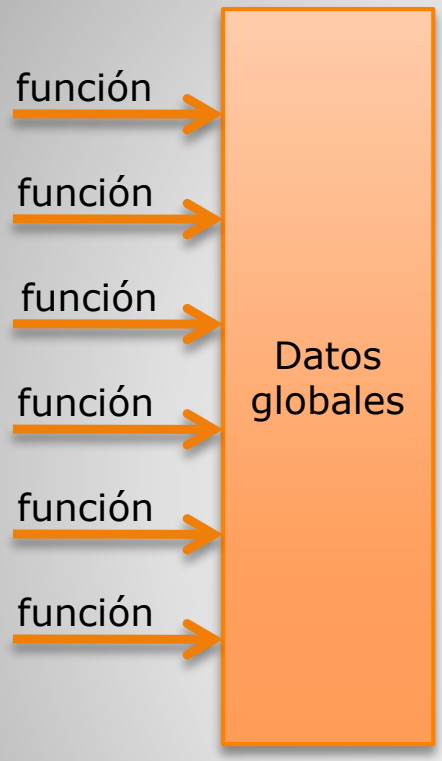
Fundamentos del Paradigma

Paradigma de programación

- Un **Paradigma de Programación** es una propuesta tecnológica que es adoptada por una comunidad de programadores cuyo núcleo central es incuestionable en cuanto a que unívocamente trata de resolver uno o varios problemas claramente delimitados.

- El **Paradigma de Programación Orientada a Objetos** es la implementación de un Paradigma de Programación

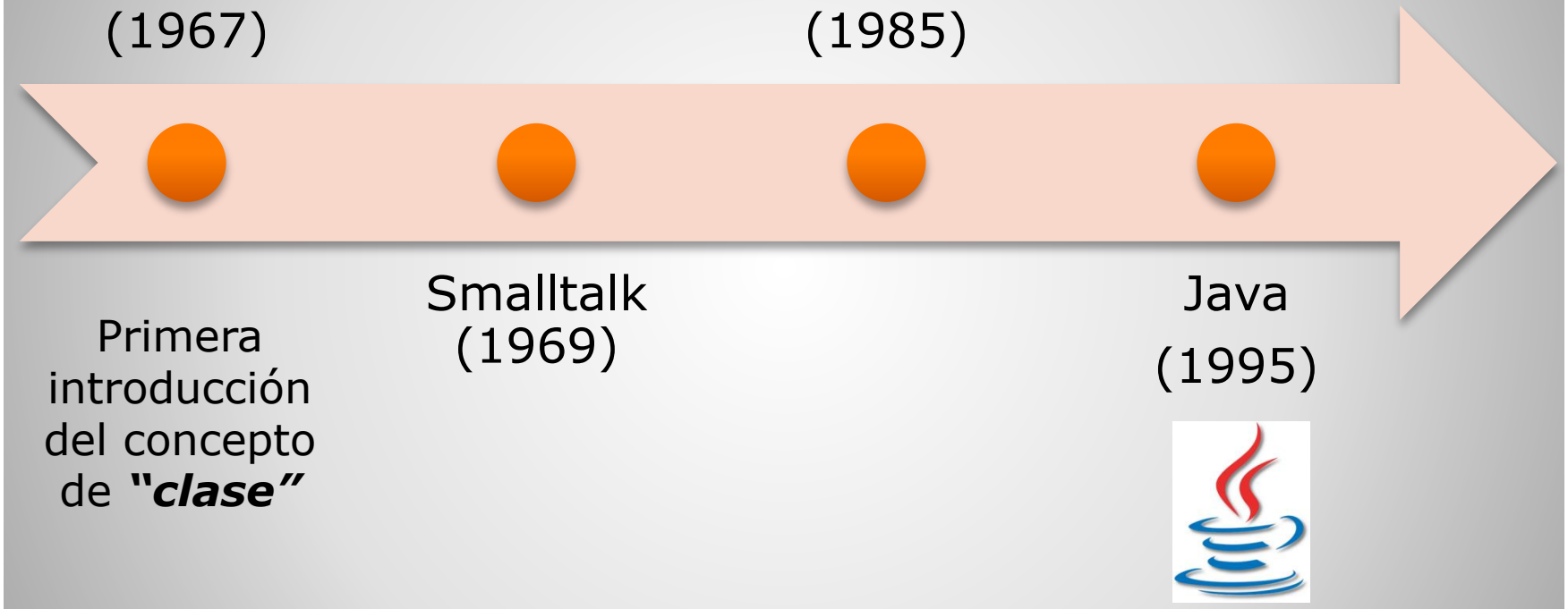
PP vs. POO



De un módulo sólo puedo tener una instancia de una clase infinitas.

Breve historia de POO

La evolución del paradigma

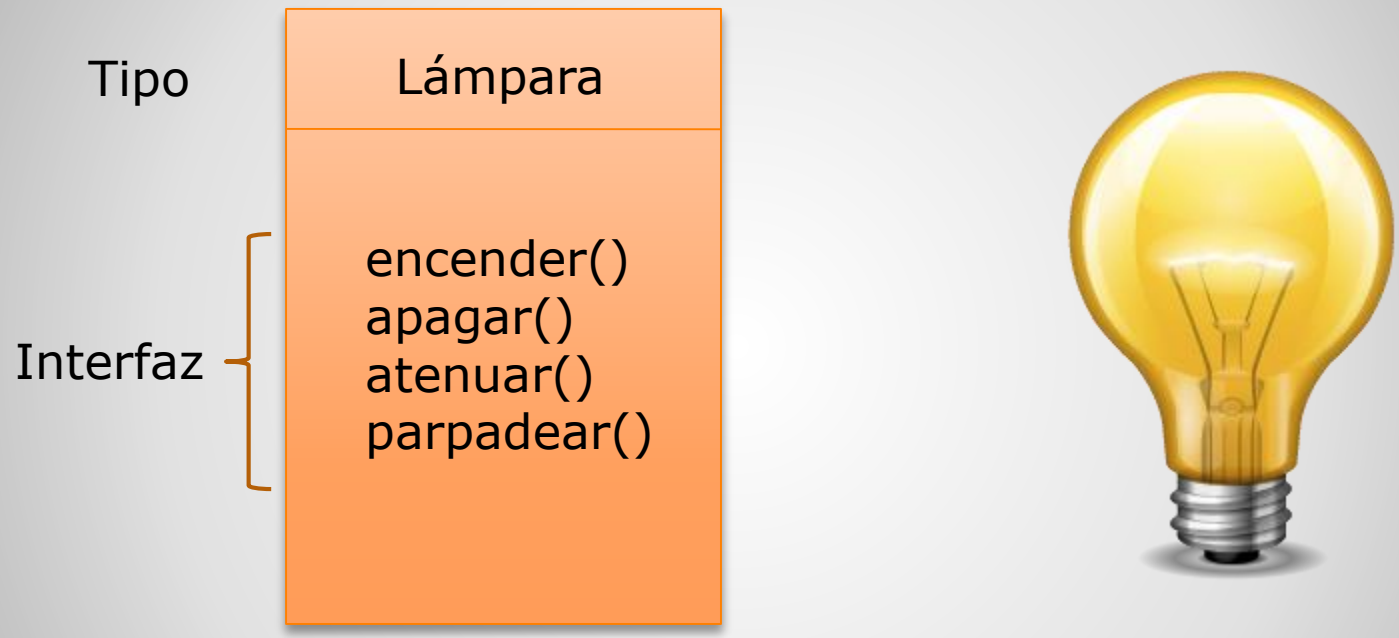


¿ Por qué POO ?

Características de POO

- Abstracción
- Encapsulamiento / principio de ocultación
- Recolección de residuos
- Polimorfismo
- Herencia

Todo es un objeto



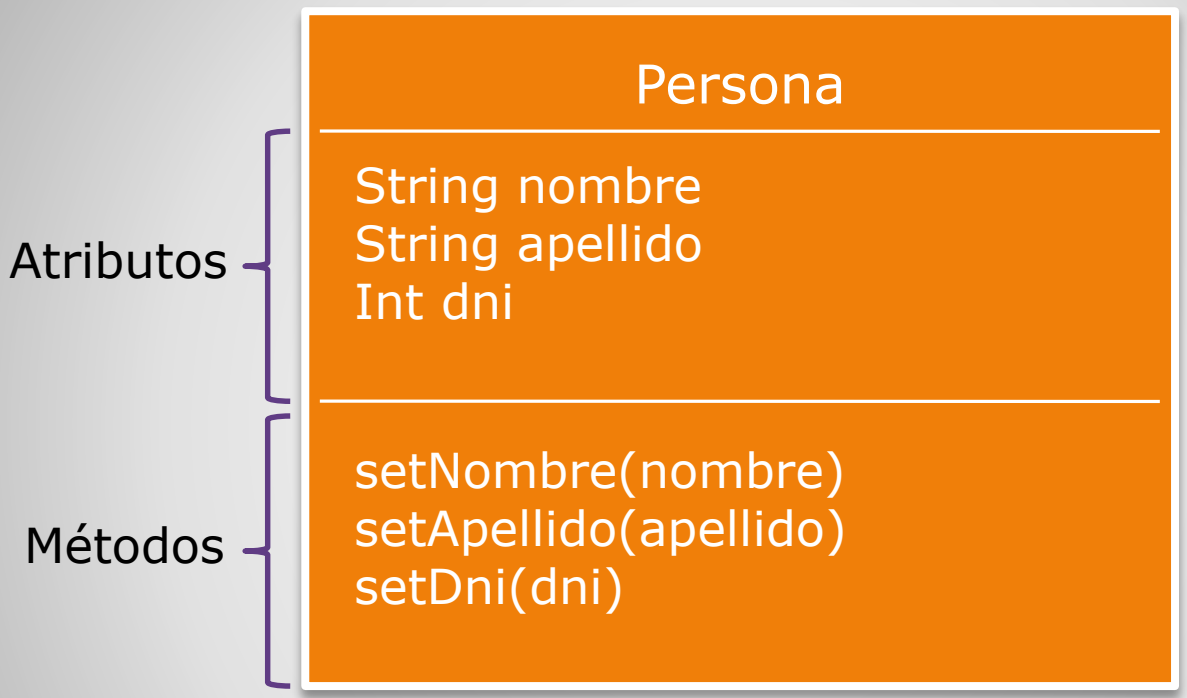
Todo objeto tiene un tipo y una interfaz

Conceptos fundamentales

- Clase
- Objeto
- Método
- Atributos
- Mensaje
- Interfaz

¿ Qué es una clase ?

- La definición de las propiedades (atributos) y comportamiento (métodos) que poseerá un tipo de objeto concreto.
- El **molde** necesario para definir el objeto.



Modelo de clase

¿ Qué es un objeto ?

- Es una **entidad** provista de un conjunto de atributos y de métodos los cuales reaccionan a determinados eventos.
- Es una **instancia** de una clase.
- Las peticiones que pueden hacerse a un objeto se denominan interfaz.

¿ Qué es un método ?

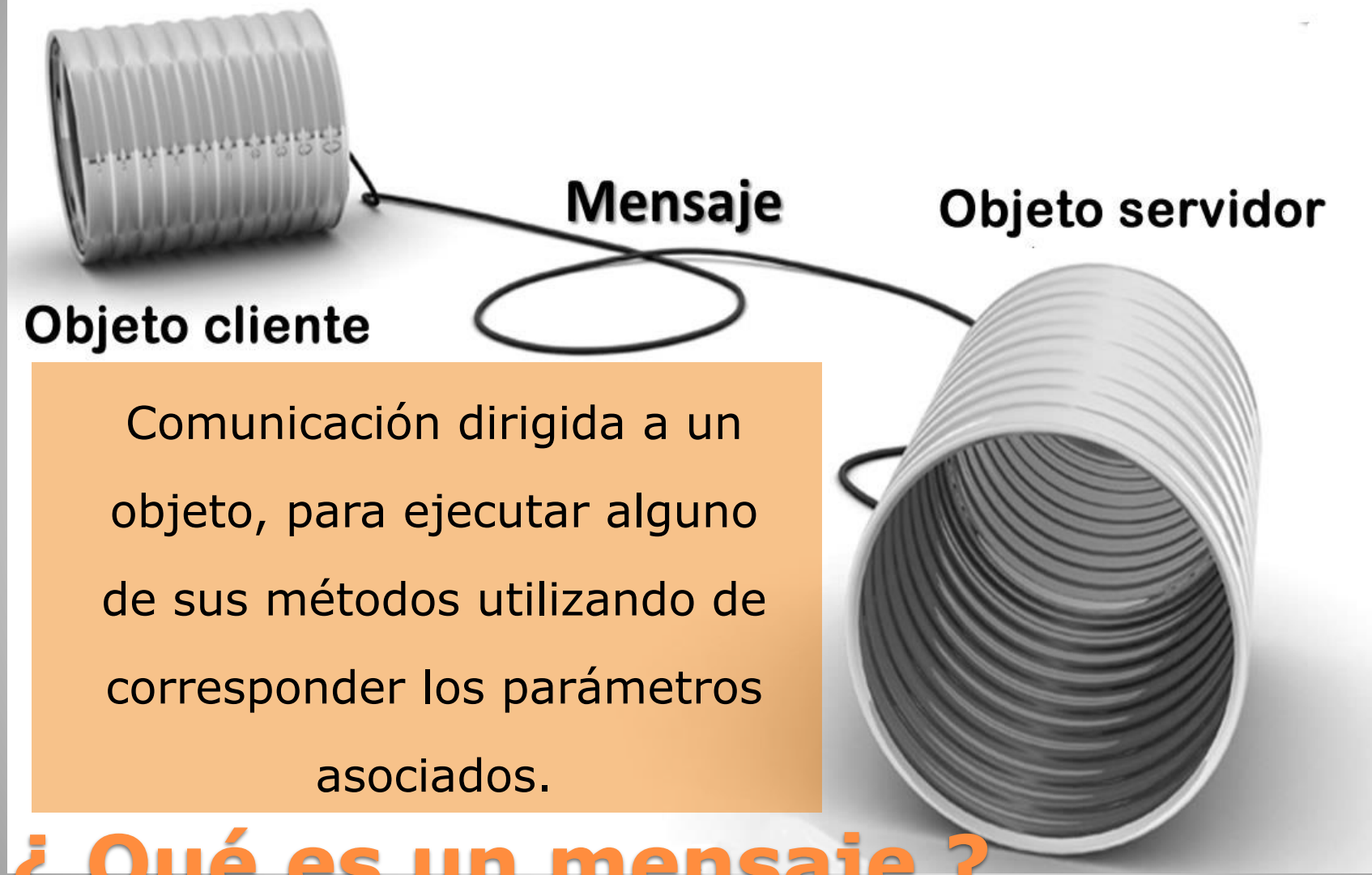
- Un bloque funcional de código asociado a un objeto.
- Su ejecución se determina a partir de la recepción de un mensaje.
- Un método puede modificar los atributos del objeto.

¿ Qué es un atributo ?

- Es un contenedor de un tipo de dato asociado a un objeto.

  Ej. : nombre, código.

- Su valor puede ser alterado a través del uso de métodos.



Comunicación dirigida a un objeto, para ejecutar alguno de sus métodos utilizando de corresponder los parámetros asociados.

¿ Qué es un mensaje ?

Modificadores de acceso

- Determinan quien puede usar las definiciones a las que preceden.
- **Public**, las definiciones están disponibles para todo el mundo.
- **Private**, nadie excepto el creador del tipo puede acceder a las definiciones.
- **Protected**, actua como privado a excepción de las clases heredadas tienen accesos a estos miembros.

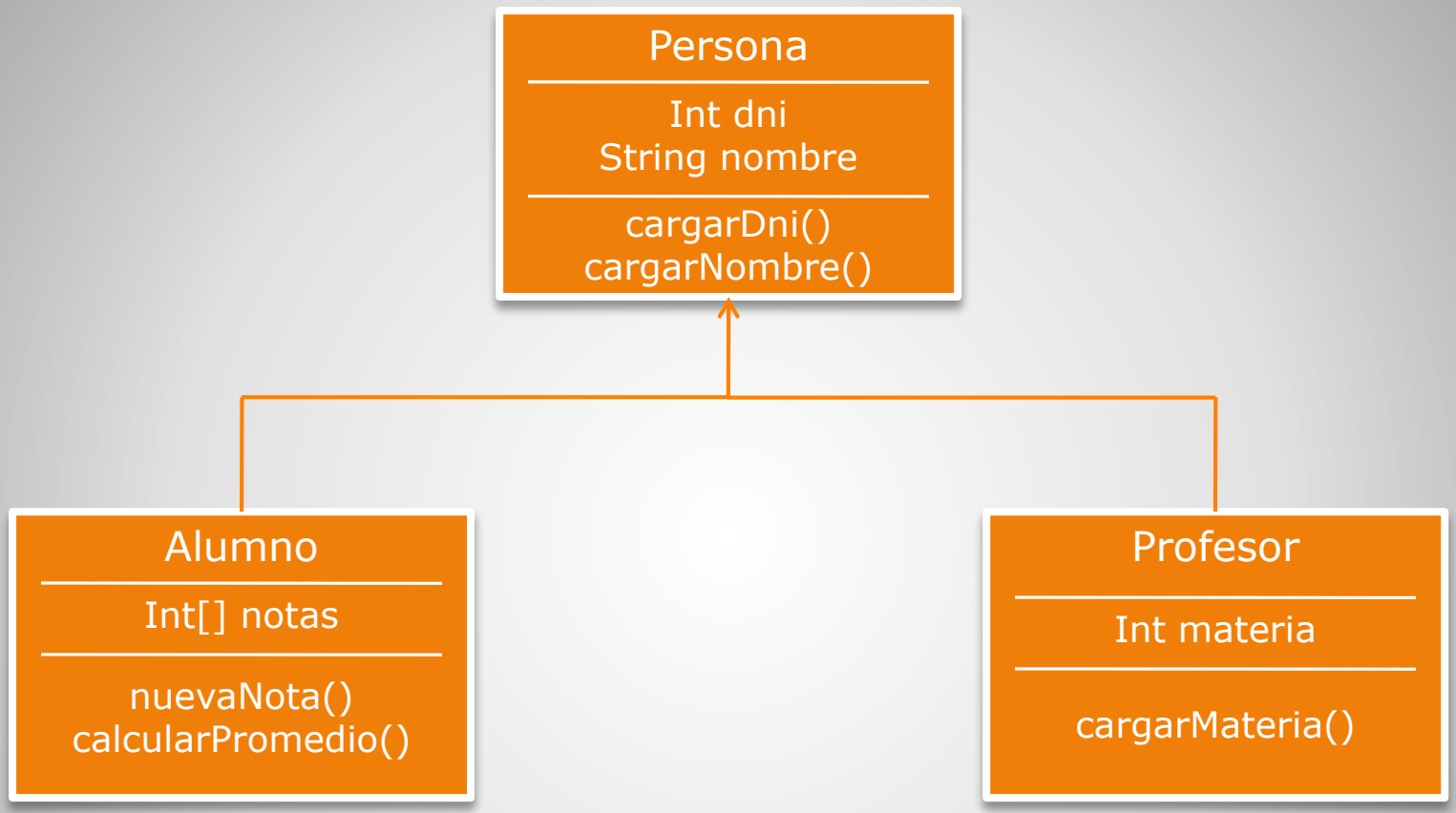
Vida de un objeto

- Creación dinámica de objetos (heap).
- Operador new()
- Recolector de basura, determina cuándo se ha dejado de usar el objeto y lo destruye.

Herencia

¿ Qué es la Herencia ?

Es la capacidad de una clase hija de heredar métodos y atributos, **como si hubieran sido definidos por ella misma**. (es una)

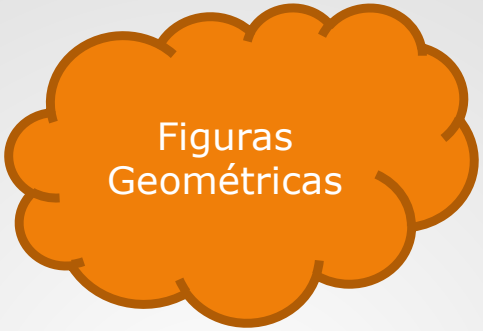


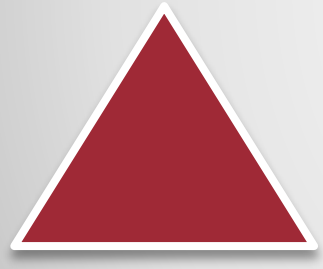
Estructura de Herencia

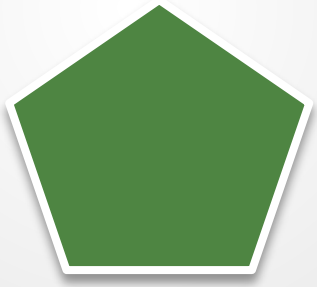
Polimorfismo

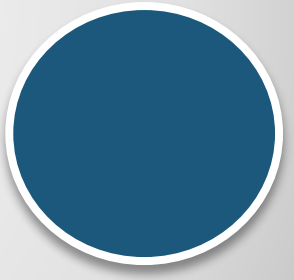
Polimorfismo

- Un mismo identificador puede tener distintas formas (distintos cuerpos de función, distintos comportamientos) dependiendo del contexto en el que se halle.
- El polimorfismo se puede establecer mediante sobrecarga, sobre-escritura.









Ejemplo de Polimorfismo

Sobrecarga

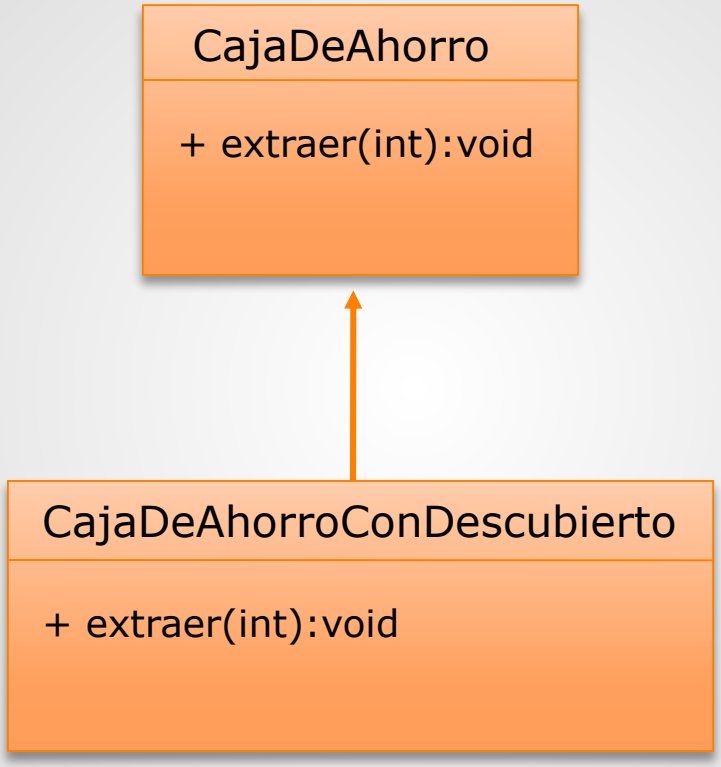
- Se refiere al uso del mismo identificador u operador en distintos contextos y con distintos significados.

  Ejemplo:

  - setPrecio(costo,moneda)
  - setPrecio(costo,porcentajeGanancia)
  - setPrecio(precio,porcentajeImpuestos)

Sobre-escritura

- La sobre-escritura se aplica a los métodos y está directamente relacionada a la herencia.
- Se refiere a la redefinición de los métodos de la clase base en las subclases.



Ejemplo de sobre-escritura

Clase abstracta

Una clase que **no será instanciada**.

Sólo presenta una interfaz para las clases derivadas.

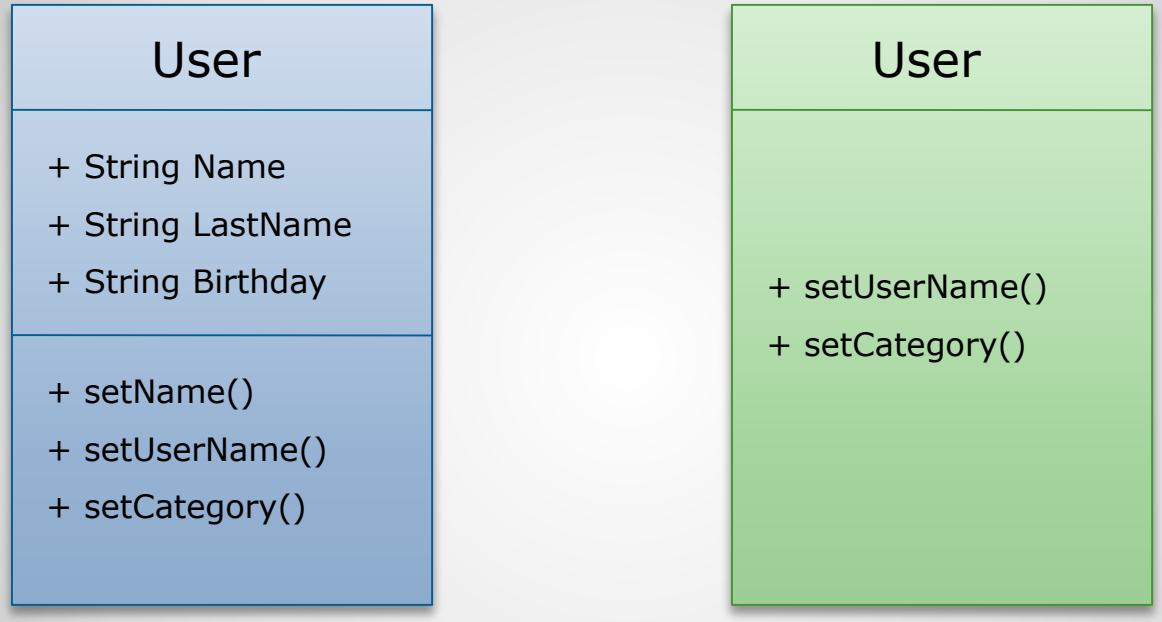
Se desea evitar generar objetos de la clase base.

(abstract)

Se crea solo dentro de una clase abstracta.

Se debe implementar por las clases que heredan a la clase base.

Método abstracto



Interface vs. clase abstracta

Jerarquía de raíz unica

Todas las clases deben ser heredadas de una única clase base. **Object**

Permite llevar a cabo ciertas operaciones básicas sobre todos los objetos del sistema.

Simplifica la implementación del garbage collector.

Tipos parametrizados

Uno de los principales cambios en Java SE5, es la incorporación de tipos de datos parametrizados (genéricos).

```
ArrayList<Forma> formas = new ArrayList<Forma>();
```

Especificación de tipo genérico

Manejo de excepciones

Una excepción es un objeto “lanzado”, desde el punto donde se produce el error y que puede ser “capturado” por el gestor de excepciones apropiado, diseñado para manejar ese tipo de error concreto.

(try-catch)

Multithreading

Se denominan hilos a los fragmentos de código que pueden ser ejecutados por separado.

Un ejemplo clásico es la interfaz de usuario.

La dificultad radica en los recursos compartidos.

servidor

middleware

cliente

CGI

transacciones

plug-ins

Arquitectura cliente/servidor

Applets

Bloques de código en Java que reemplazan la función de los scripts en el cliente.

La mayor ventaja frente a los scripts JavaScript es que son código compilado y por lo tanto no transparente para el usuario.

Los Ventajas de POO

- Escalabilidad
- Mantenimiento
- Seguridad
- Reutilización
- Simplicidad