UF7.2
Herencia - Principios

Universidad Europea
LAUREATE INTERNATIONAL UNIVERSITIES

Madrid Valencia Canarias

CONTENIDOS

1. Introducción
    1. Definición, justificación.
    2. Ascendentes y descendientes
2. Tipos
3. Implementación (extends)
4. Clase “object”
5. Inicialización (super)
6. Restricción (final)
7. Tipo de (instanceof)

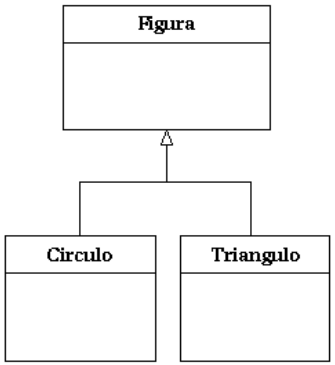
Herencia

La herencia es un mecanismo de la OOP que permite construir una clase incorporando de manera implícita todas las características de una clase previamente existente.

Justificación

Las razones que justifican su necesidad son variadas:

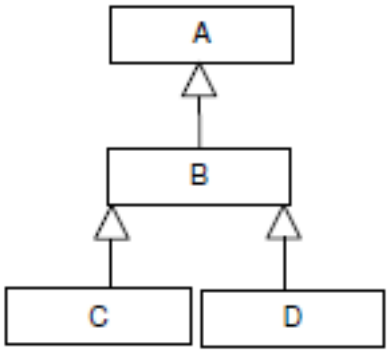
- Modelado de la realidad. Son frecuentes las relaciones de especialización/generalización entre las entidades del mundo real, por tanto es lógico que dispongamos de un mecanismo similar entre las clases de objetos
- Evitar redundancias. Toda la funcionalidad que aporta una clase de objetos es adoptada de manera inmediata por la clase que hereda, por tanto evitamos la repetición de código entre clases semejantes
- Facilitar la reutilización. Una clase no tiene por qué limitarse a recibir una serie de características de otra clase por herencia de forma pasiva. También disponen de cierto margen de adaptación de estas características
- Soporte al polimorfismo



Ascendentes y Descendentes

Sea una clase A. Si una segunda clase B hereda de A entonces decimos:

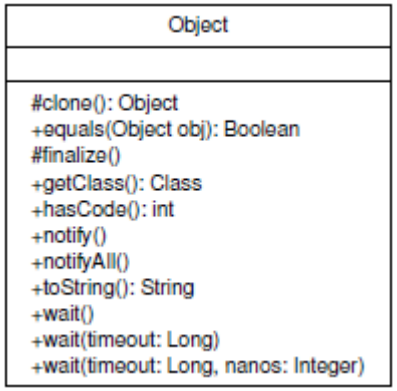
- A es un ascendiente o superclase de B. Si la herencia entre A y B es directa decimos además que A es la clase *padre* de B.
- B es un descendiente o subclase de A. Si la herencia entre A y B es directa decimos además que B es una clase *hija* de A



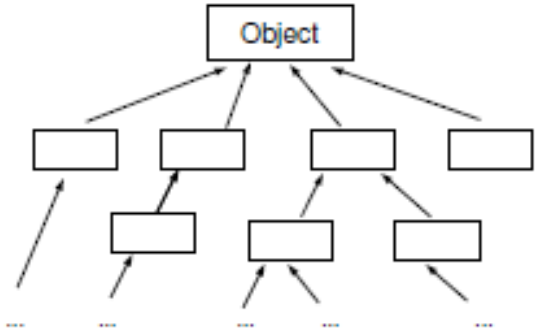
En Java, Eiffel, Smalltalk y otros lenguajes orientados a objetos puros, todas las clases heredan automáticamente de una superclase universal. En Java esta superclase se denomina **Object**

Clase Object

Todos las clases en Java heredan en última instancia de Object. Incluso si creamos una clase independiente, Java la hace heredar implícitamente de Object:



Esto hace que las clases formen una jerarquía con Object como raíz:

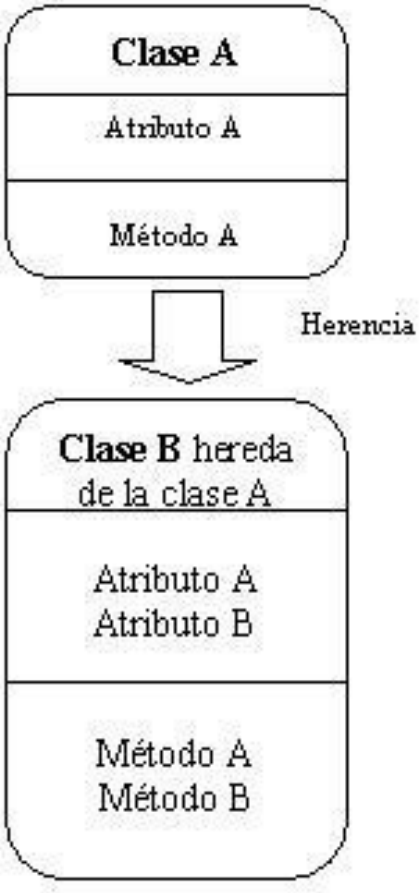


Actividad: Busca atributos y métodos de la clase object.

Tipos de herencia

Existen diferentes situaciones en las que puede aplicarse herencia:

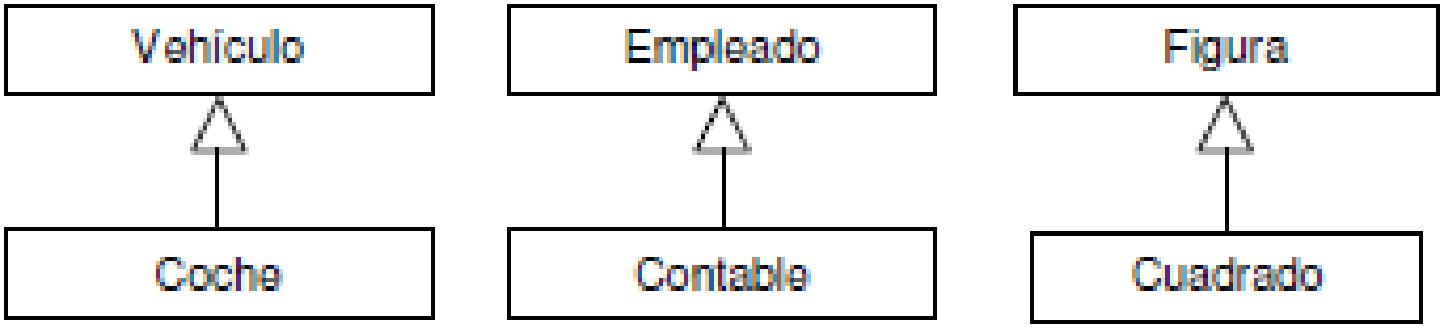
1. Especialización
2. Extensión
3. Especificación
4. Construcción



1.- Especialización

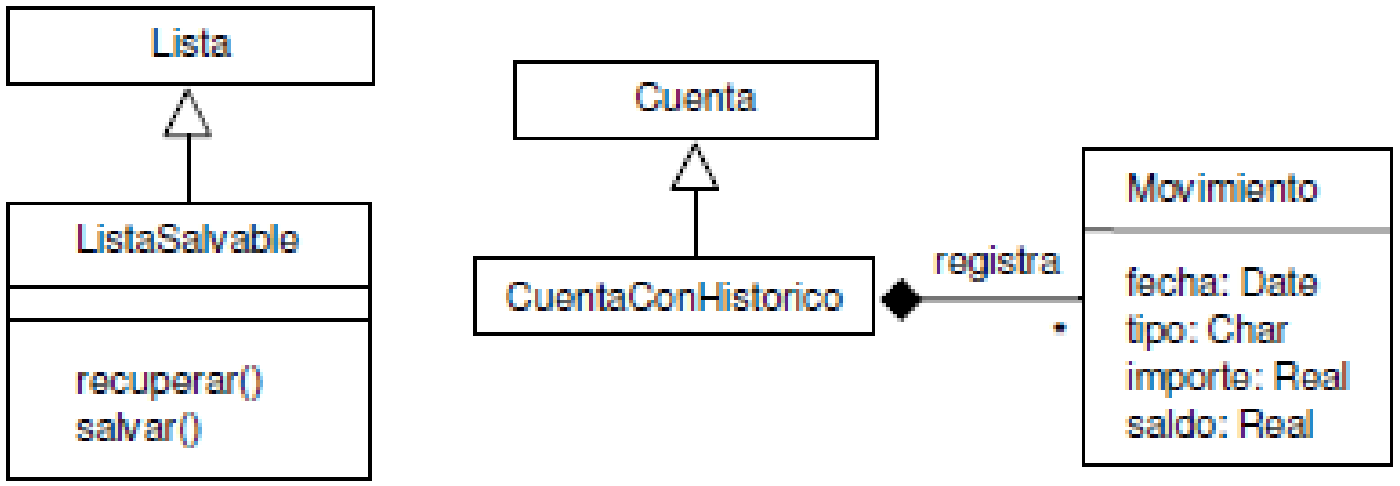
Dado un concepto B y otro concepto A que representa una **especialización** de A, entonces puede establecerse una relación de herencia entre las clases de objetos que representan a A y B.

En estas situaciones, el enunciado "A es un B" suele ser aplicable:



2.- Extensión

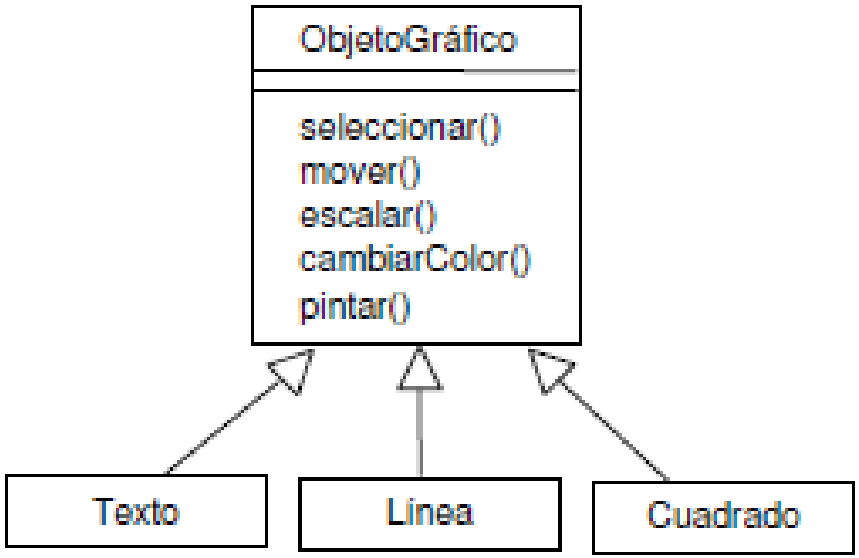
Una clase puede servir para extender la funcionalidad de una superclase sin que represente necesariamente un concepto más específico:



3.- Especificación

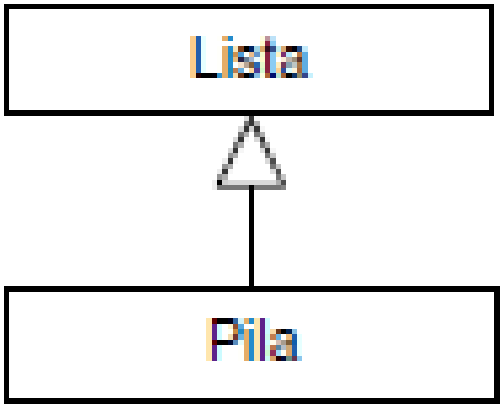
Una superclase puede servir para especificar la funcionalidad mínima común de un conjunto de descendientes.

Existen mecanismos para obligar a la implementación de una serie de operaciones en estos descendientes:



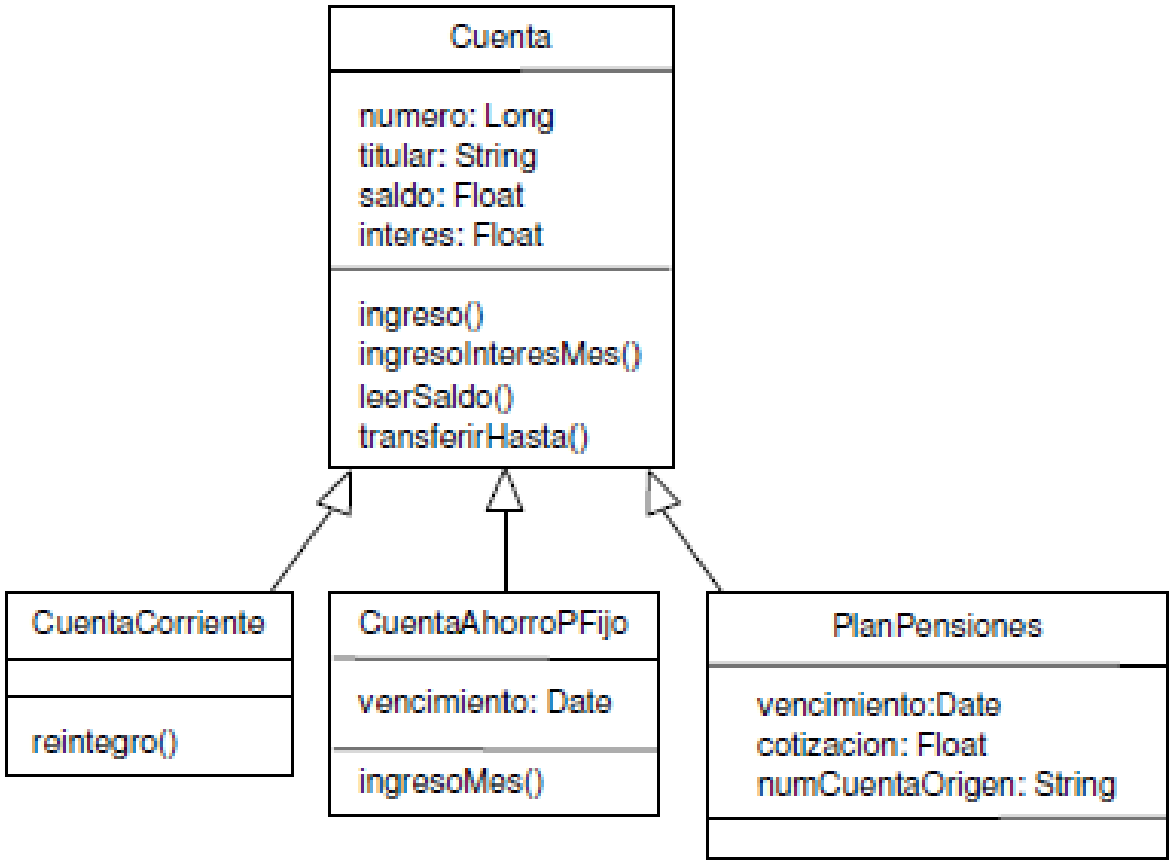
4.- Construcción

Una clase puede construirse a partir de otra, simplemente porque la hija puede aprovechar internamente parte o toda la funcionalidad del padre, aunque representen entidades sin conexión alguna:



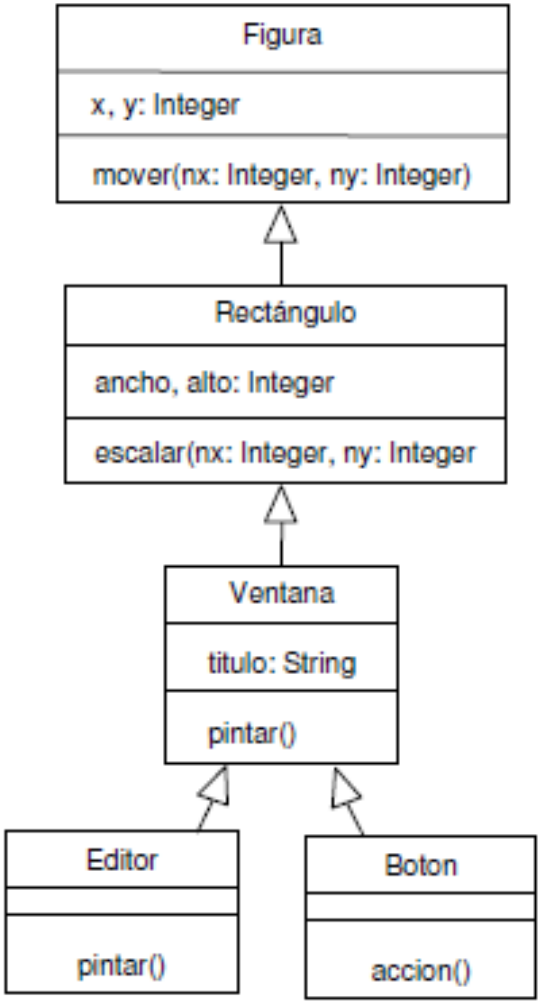
Ejemplo 1

Distintos tipos de cuentas bancarias:



Ejemplo 2

Elementos de una interfaz de usuario:



Ejemplo 3

Estructuras de datos: