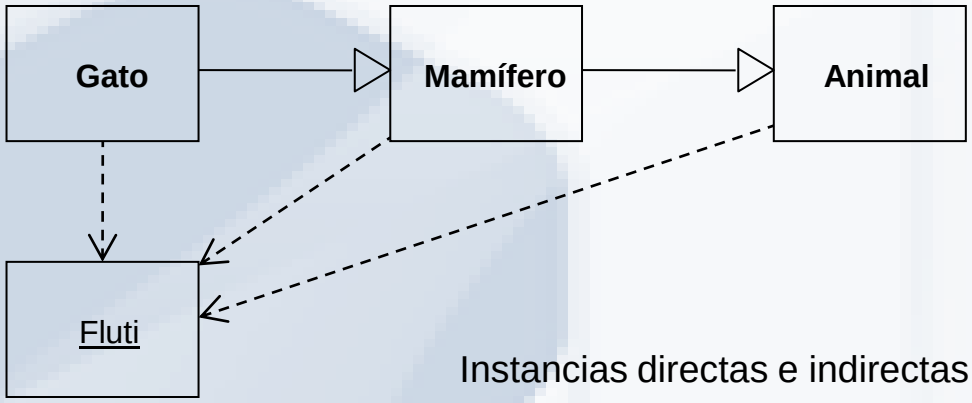
Modelado Estático Avanzado (Generalizaciones)

Generalización y Clasificación

- Principio de sustitución:
  - **Extensión**: todos los objetos de la subclase son también de la superclase.
  - **Intensión**: la definición de la superclase es aplicable a la subclase.
- Generalización: clase-clase.
  - Gato **es un tipo de** Mamífero, Mamífero **es un tipo de** Animal.
- Clasificación: objeto-clase.
  - Fluti **es un** Gato, Fluti **es un** Mamífero, Fluti **es un** Animal.

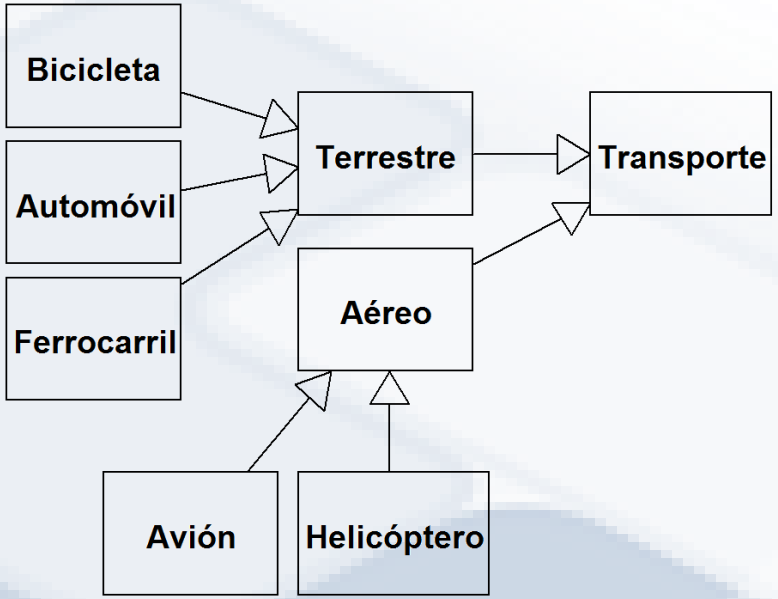


Instancias directas e indirectas

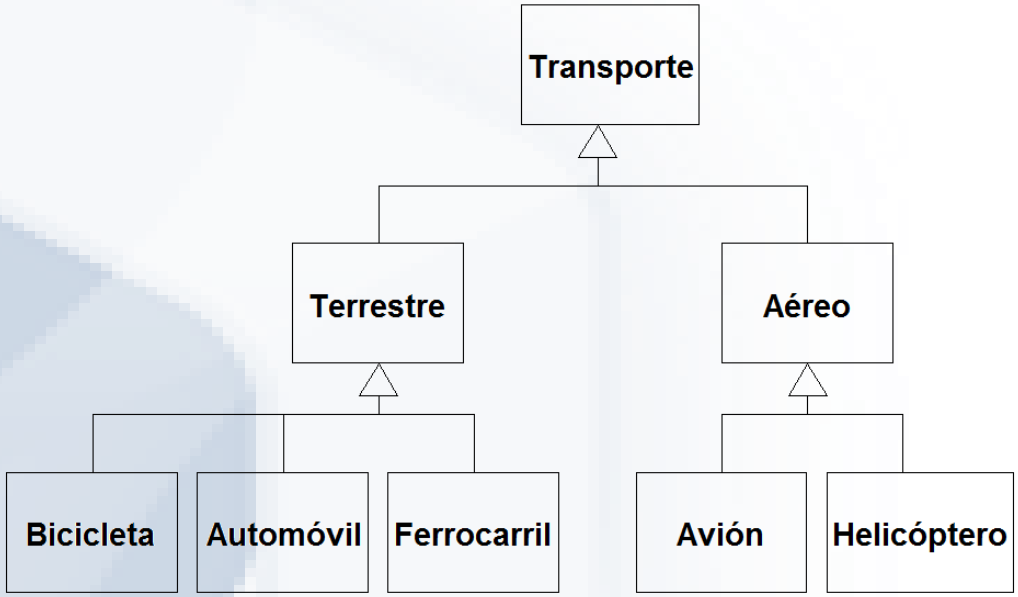
Generalización y Especialización

- Dos puntos de vista complementarios:
  - Generalizar es identificar los **elementos comunes** (atributos, asociaciones, operaciones) de varias clases y representarlos en una clase más general denominada **superclase**.
    - Elevar el nivel de abstracción, reducir la complejidad, organizar.
  - Especializar es capturar los **elementos específicos** de un conjunto de objetos dentro de una clase dada, que aún no han sido identificados en ella, y representarlos en una nueva clase denominada **subclase**.
    - Reutilizar un concepto añadiendo elementos nuevos.
- Es una relación pura entre clases:
  - No tiene instancias, ni multiplicidad.
  - La subclase hereda **todos** los miembros de la superclase.
  - Los miembros heredados de la superclase no se representan en la subclase (a menos que sean operaciones redefinidas).
  - Toda generalización induce una **dependencia** subclase → superclase.

Jerarquías de Clases



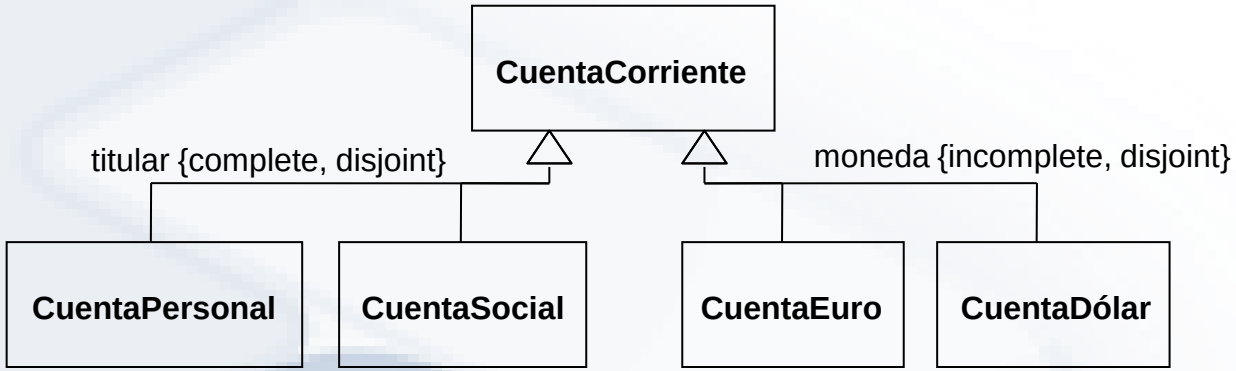
Representaciones alternativas:
- Relaciones binarias
- Estructura en árbol



Generalización:
- No-reflexiva
- Transitiva
- Asimétrica

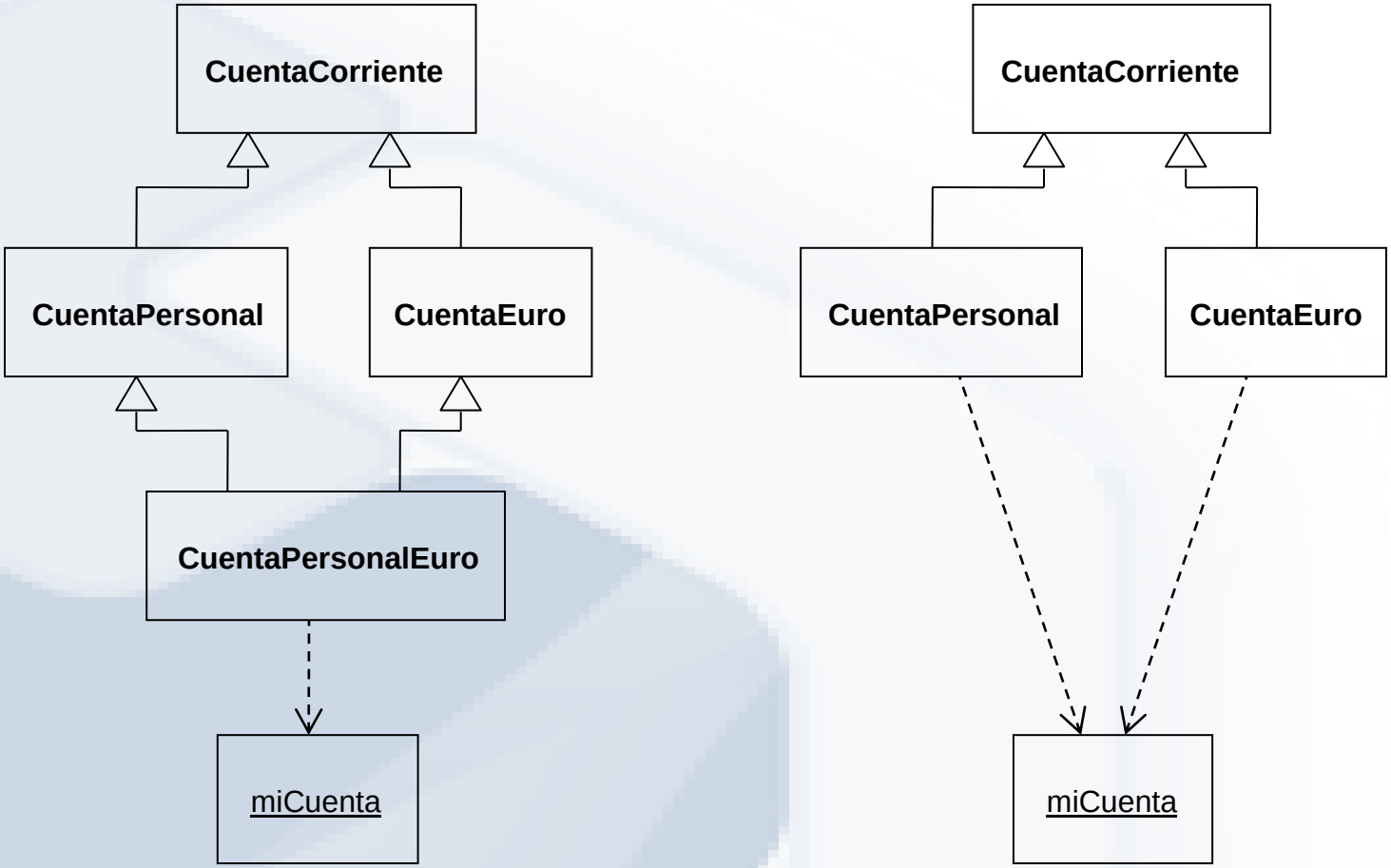
Dimensiones de Especialización

- Una superclase puede ser especializada en distintos **grupos de subclases** de acuerdo con criterios independientes: **discriminadores**.



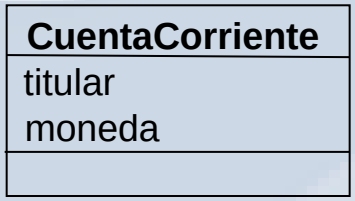
- Restricciones:
  - **disjoint/overlapping**: las subclases no pueden tener instancias en común/o sí.
  - **complete/incomplete**: no hay otras subclases/o sí.
- Valor por omisión: {incomplete, disjoint}
- **Partición propiamente dicha**: {complete, disjoint}

Generalización Múltiple vs. Clasificación Múltiple

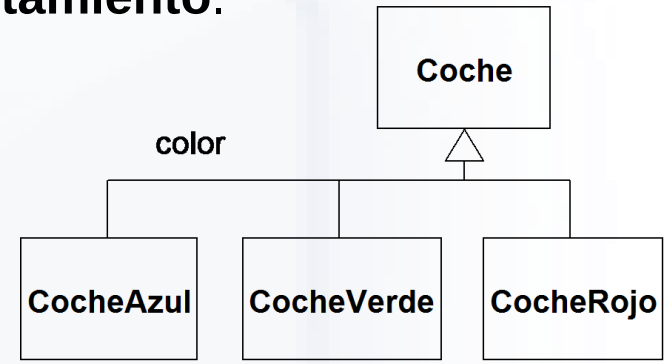


Subclase vs. Atributo

- ¿Cómo modelar las propiedades de los objetos? Regla general:
  - Propiedad cambiante o rango de valores muy grande: **atributo**.
  - Propiedad fija con valores enumerados: **especialización** (cada propiedad se traduce en un **criterio** de especialización, cada valor en una **subclase**).
    - También se puede modelar como un atributo con valor fijo.
- **Clasificación dinámica**: un objeto puede cambiar de clase.
  - Permite modelar un cambio de propiedad como una reclasificación.
  - Especialmente interesante para aprovechar el polimorfismo (cambiar de subclase).
  - No es habitual en los lenguajes de programación.
- Criterio final de especialización: **comportamiento**.



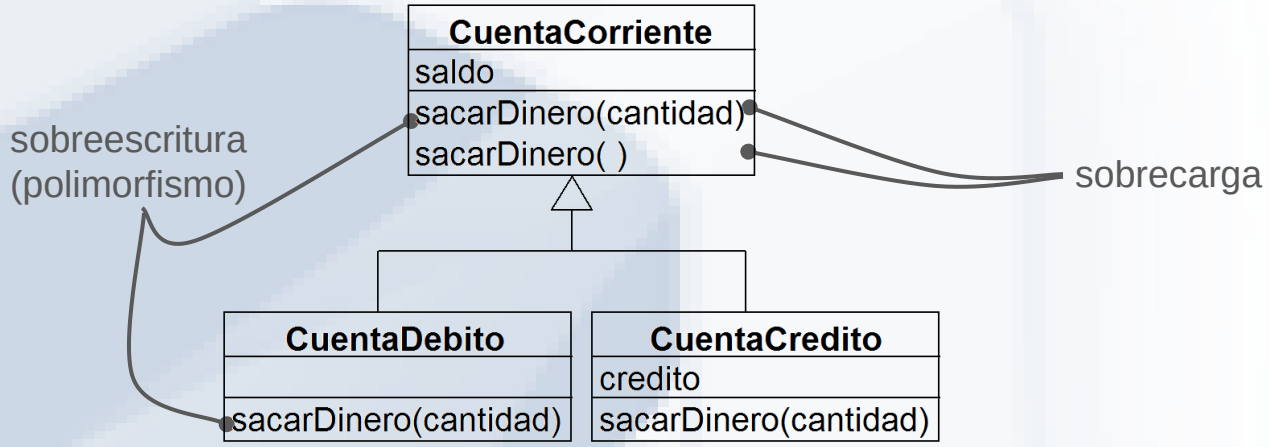
Alternativa a la doble especialización



¿Especialización exagerada?

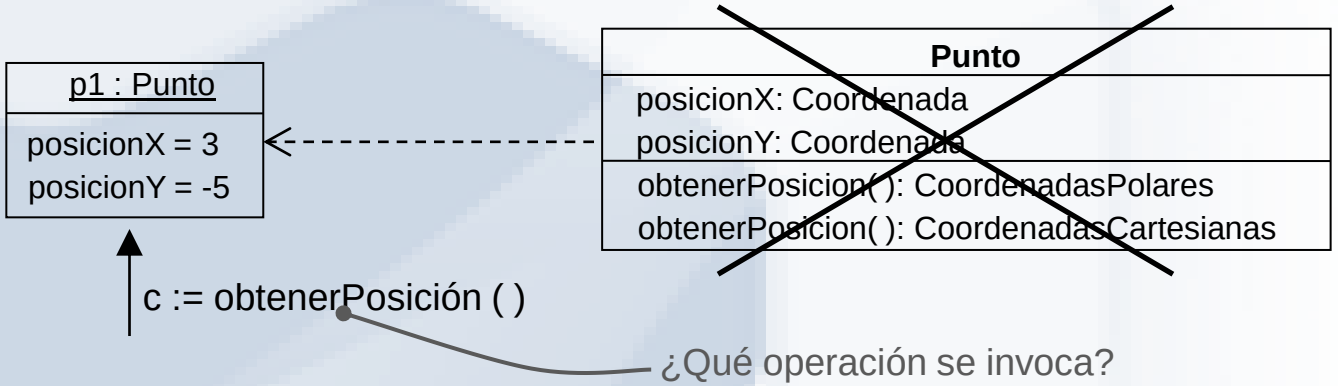
Polimorfismo de Operaciones

- Capacidad de ejecutar distintos métodos en respuesta al mismo mensaje.
- Una **operación polimórfica** es aquélla que tiene muchas implementaciones.
- No confundir **sobreescritura** con **sobrecarga** de operaciones.
  - Sobreescribir: redefinir en otra clase el **significado** de la misma operación.
    - El método se selecciona en tiempo de **ejecución**.
  - Sobrecargar: reutilizar el **nombre** de operación, pero con distintos parámetros.
    - La operación se selecciona en tiempo de **compilación**, no es polimorfismo.



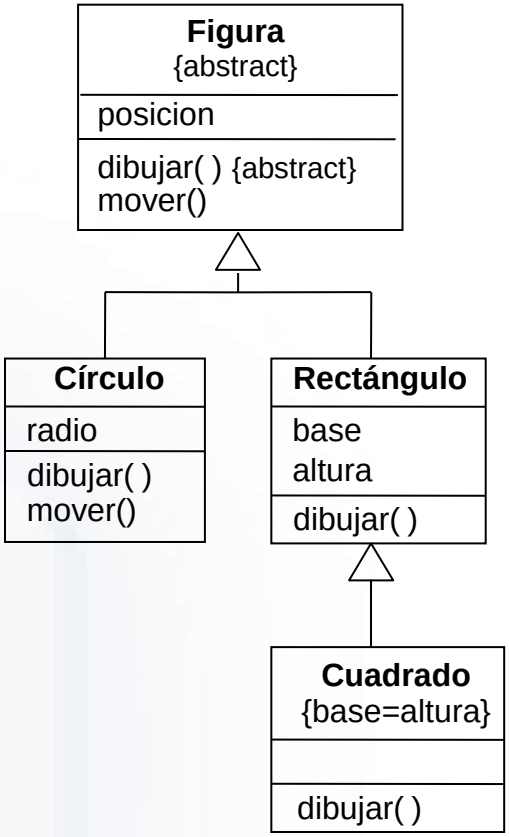
Signatura de Operaciones

- Una clase no puede tener dos operaciones con la misma **signatura** o firma, que consta de:
  - **Nombre** de operación, número (orden) y **tipo de los parámetros**.
- Los **nombres de los parámetros** no pertenecen a la signatura.
- El **tipo del valor de retorno** tampoco pertenece a la signatura, porque no sirve para distinguir qué operación hay que ejecutar.
  - No se puede usar para **sobreescribir** o **sobrecargar** operaciones.



Clases y Operaciones Abstractas

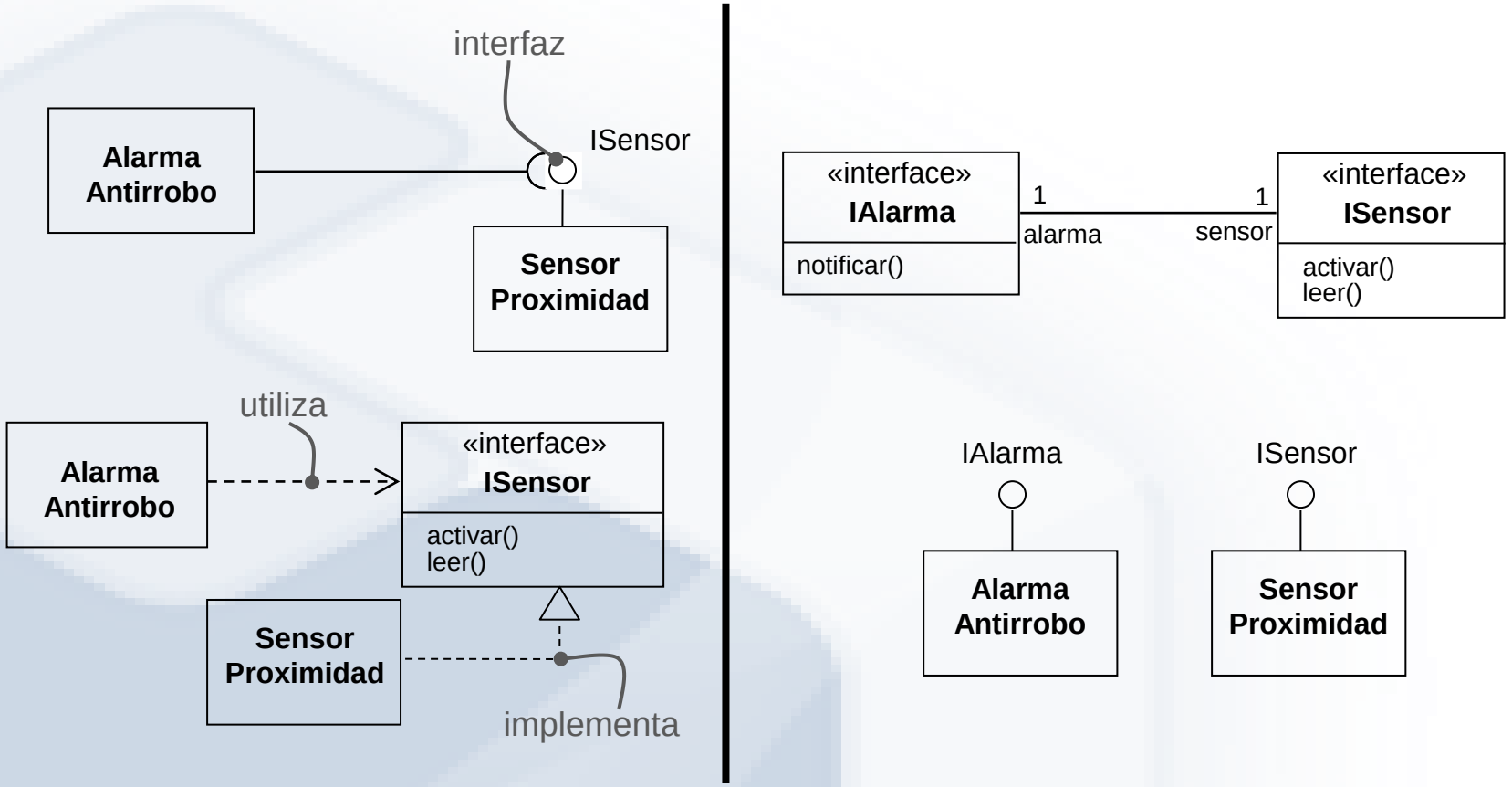
- **Operación abstracta**: se especifica la signatura, no la implementación.
  - Una clase con una o varias operaciones abstractas está incompleta: no puede tener instancias directas.
  - Tanto las operaciones abstractas como las concretas pueden ser sobreescritas (polimórficas).
  - Es más seguro sobreescribir una operación abstracta que una operación concreta (no hay peligro de cambiar su significado).
- **Clase abstracta**: está incompleta, no puede tener instancias directas.
  - Puede tener instancias indirectas a través de sus subclases concretas.
  - Una clase concreta...
    - No puede tener operaciones abstractas.
    - Debe proporcionar implementaciones para todas las operaciones abstractas heredadas.
  - Para indicar que una clase es abstracta se pondrá el nombre de la clase en **cursiva**, o bien se puede poner la palabra clave **{abstract}** a continuación o debajo del nombre de la clase.



Interfaces (I)

- Encapsulamiento: separación de interfaz e implementación en una clase.
  - Una clase puede realizar una o varias interfaces.
  - Una interfaz puede ser usada por una o varias clases.
- Interfaz: **conjunto de atributos y operaciones** que ofrecen un servicio coherente.
  - No contiene la implementación de las operaciones.
  - No se puede instanciar.
  - Una interfaz puede tener **asociaciones** navegables.
  - Análoga a una **clase abstracta** con todas sus operaciones abstractas.
  - Distinta de **interfaz natural**: conjunto de operaciones públicas de una clase (accesibles a través de cualquier asociación navegable hacia la clase).

Interfaces (y II)



Notaciones para uso y realización de interfaces

Generalización vs. Realización

- La realización puede entenderse como una **“generalización débil”**: se hereda la interfaz, pero no la implementación:
  - Reduce la dependencia.
  - Disminuye la reutilización.
  - Alternativa a la generalización múltiple, no soportada por muchos lenguajes.
- Las interfaces son elementos generalizables.
  - Jerarquías mixtas de interfaces y clases.
- Criterio de diseño: comprometerse sólo con la interfaz.
  - Declarar el tipo de las variables y parámetros de operaciones como interfaces, no como clases.
  - Servirá cualquier instancia compatible con la interfaz.
- Ejemplo:
  - `Figura f = new Cuadrado();`
  - `f.imprimir();`