Unidad III
Técnicas de POO

Técnicas de la POO

- Abstracción
- Encapsulamiento y ocultación de datos
- Herencia
- Polimorfismo
- Reusabilidad o reutilización de código

Abstracción

Nivel general(Interfaz)

Nivel detallado(implementación)

- Propiedad que considera los aspectos más notables del problema y expresa la solución en esos términos.
- Representa la información de la interfaz con el usuario.
- La abstracción se representa con una clase que implementa la interfaz correspondiente.
- Los grados de la abstracción se denominan niveles de abstracción.

Abstracción

- La interfaz será una clase que nos diga como utilizar los servicios que nos provee (comportamientos).
- La clase también implementa la interfaz, donde se detalla como realmente se construye el servicio, pero es desconocido por los usuarios (otras clases) de los servicios.
- Las demás clases (usuarios) ocuparan simplemente los servicios de la interfaz, nunca sabrán como se implementan.

Ejemplo: El auto me permite acelerar “eso me interesa….QUE HACE”

Y como funciona realmente ese mecanismo “eso no me interesa….COMO LO HACE”

La abstracción en SW:

- Se puede utilizar código sin tener conocimiento de la implementación fundamental.

  *Ejemplo: cuando en C utilizamos la función **sqrt** (de la librería "Math.h"), no conocemos su algoritmo real. Este puede ser mejorado y no nos daríamos cuenta ya que siempre seguiremos teniendo un resultado.*

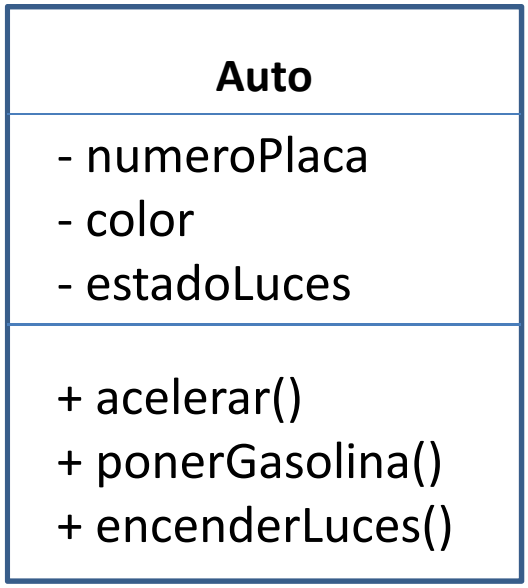
- La abstracción es el principio fundamental, tras la reutilización.

- Solo se pueden reutilizar elementos en los cuales se haya abstraído su esencia del mundo real.

- Es decir en el análisis del sistema hay que concentrarse en "qué hace" y no en "cómo lo hace".

Abstracción

- En el análisis de un auto, sólo interesa conocer **qué** servicios que presta (operaciones), no **cómo** hace para ejecutarlos.



Encapsulamiento y Ocultación de Datos

- La encapsulación es la **reunión** en una estructura, de todos los elementos que en un nivel de abstracción se consideran parte de una misma entidad (categoría o clase).

- También es agrupar los datos y operaciones relacionados bajo una misma unidad de programación (**cohesión alta**, o bien, las características están fuertemente relacionadas).

- La encapsulación **oculta lo que hace un objeto** de lo que hacen otros objetos; por eso se le llama también ocultación de datos.

- La **interfaz** (operaciones o métodos) de una clase es como un contrato en la que ofrece sus servicios a otros componentes externos (por ejemplo, otras clases).

Encapsulamiento y Ocultación de Datos

- De este modo los clientes de un componente (clase) solo necesitan conocer cuales son los servicios de su interfaz (métodos) y como utilizarlos(necesita parámetros o no). No necesitan conocer como se implementan.

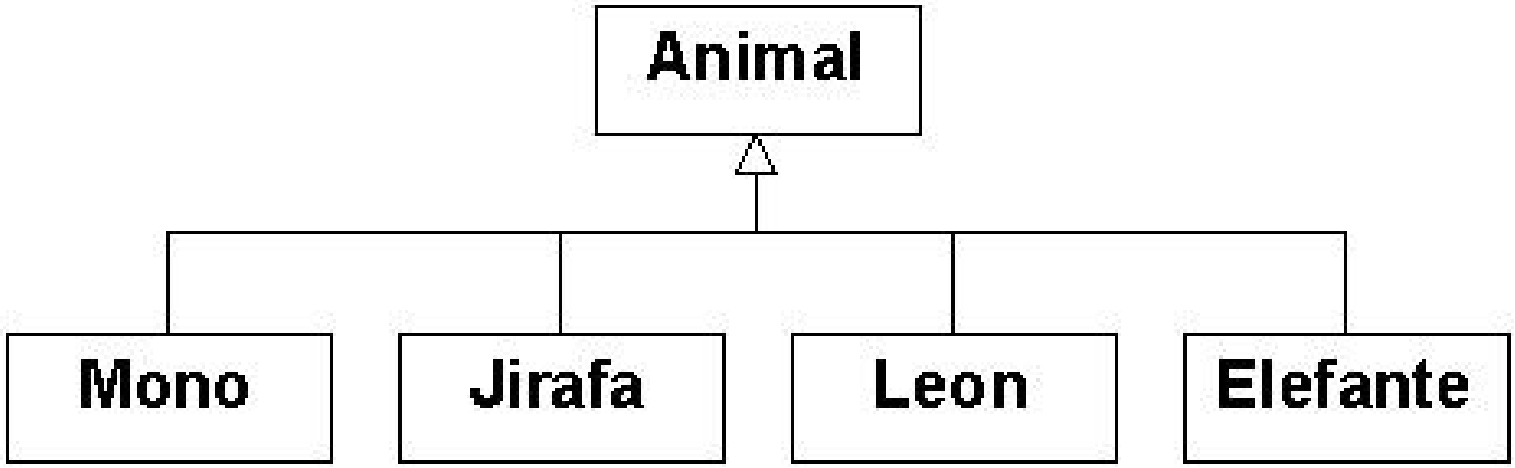
- Así, se puede modificar la implementación de la interfaz en una clase sin afectar a las restantes clases relacionadas con ella, solo es necesario mantener o conservar la interfaz.

- Por lo que, la interfaz indica que se puede hacer con el objeto (caja negra).

- La interfaz pública es estable, pero la implementación se puede modificar.

Herencia

- Clase **Padre**: Animal.
- Clases **hijas**: Mono, Jirafa, Leon, Elefante.
- Las clases se dividen en subclases.



Herencia

- Trata de modelar la herencia como en la vida real.
- La idea es que las clases hijas (subclases) comparten características con la clase padre (superclase o clase principal).
- Además de las características compartidas, cada subclase tiene sus propias características.
- La clase padre también se le conoce como: principal, superclase o base.
- Las clases hijas también se les conoce como: subclase o derivada.

Herencia

- Las técnicas de herencia se representan con la relación “es-un”.

  En nuestro ejemplo podemos decir: un mono “es-un” animal.

- *Un mono tiene sus propios comportamientos: sube a los árboles, salta de un árbol a otro, etc.*

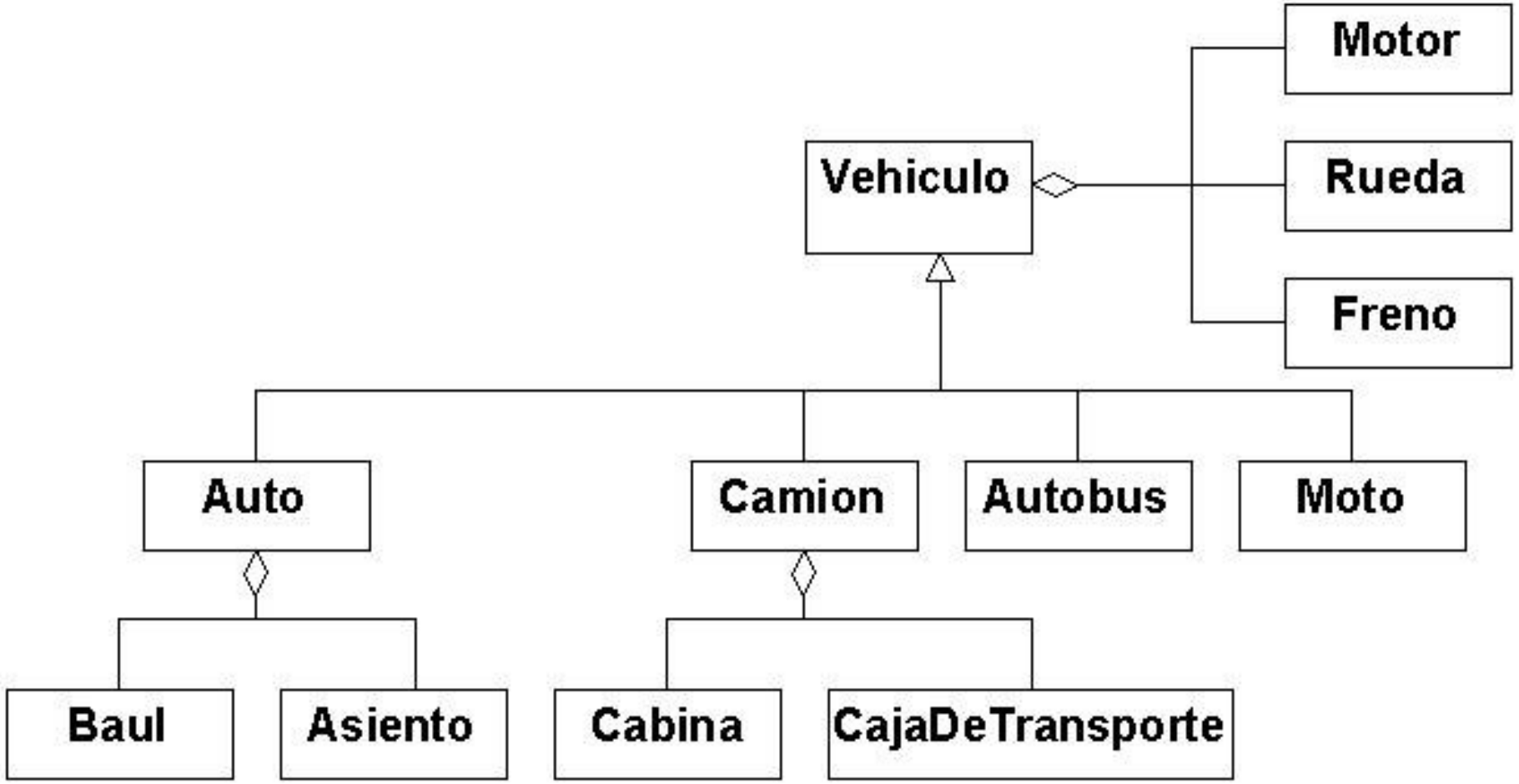
- *Comparte comportamientos con la jirafa y el león: comen, duermen, ser reproducen, etc.*

- *Y también comparten características heredadas de la clase animal:*

  altura, peso, número de patas, etc

Herencia

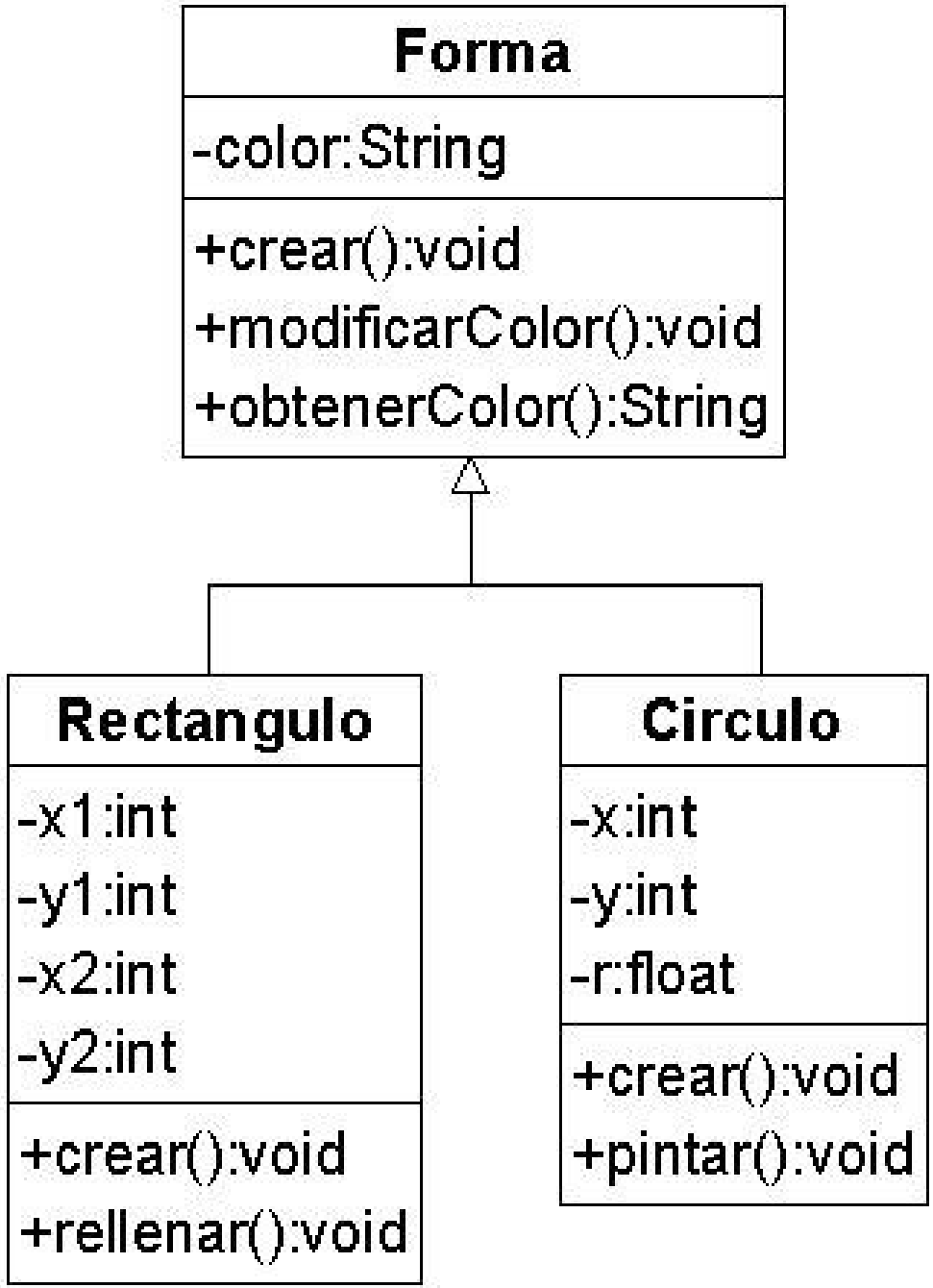
- Ejemplo 1: **Camion** hereda Motor, Rueda y Freno.



Herencia

- Ejemplo 2

Rectangulo y **Circulo** heredan el atributo color y los métodos para modificar y obtener el color.



Herencia

- Las clases modelan la realidad en base a que los objetos de esta contienen atributos y comportamiento.

- La herencia modela en base a que los objetos se organizan en jerarquías.

- La jerarquía en base al modelo se define como relación de ***generalización o "es un".***

- En POO la relación de generalización se denomina **herencia**.

Herencia

- Cada clase derivada hereda las características (atributos y operaciones) de la clases base; y además, agrega sus propias características.

- La clase base puede a su vez ser clase derivada de otra clase

Jerarquía de Clases: Generalización / Especialización

- La jerarquía de clases gestiona la complejidad ordenando objetos en árboles de clases con niveles crecientes de abstracción.

- Las jerarquías de clases más conocidas son la generalización y la especialización.

- La relación de generalización es un concepto fundamental de la POO y consiste en relacionar una superclase “padre” con una o varias subclases “hijas”.

- Se le llama también extensión o herencia.

Jerarquía de Clases: Generalización / Especialización

- La relación de generalización se representa con una flecha que comienza en la subclase y termina en la superclase.
- En UML esta relación se conoce como generalización y en POO como herencia.
- Estas relaciones no tienen nombre ni multiplicidad.
- Hay que tomar en cuenta que la herencia es transitiva: una clase hereda de todas las clases antecesoras.

Jerarquía de Clases: Generalización / Especialización

- La generalización es una relación de herencia entre dos clases; es decir una clase hereda atributos y métodos de la otra.

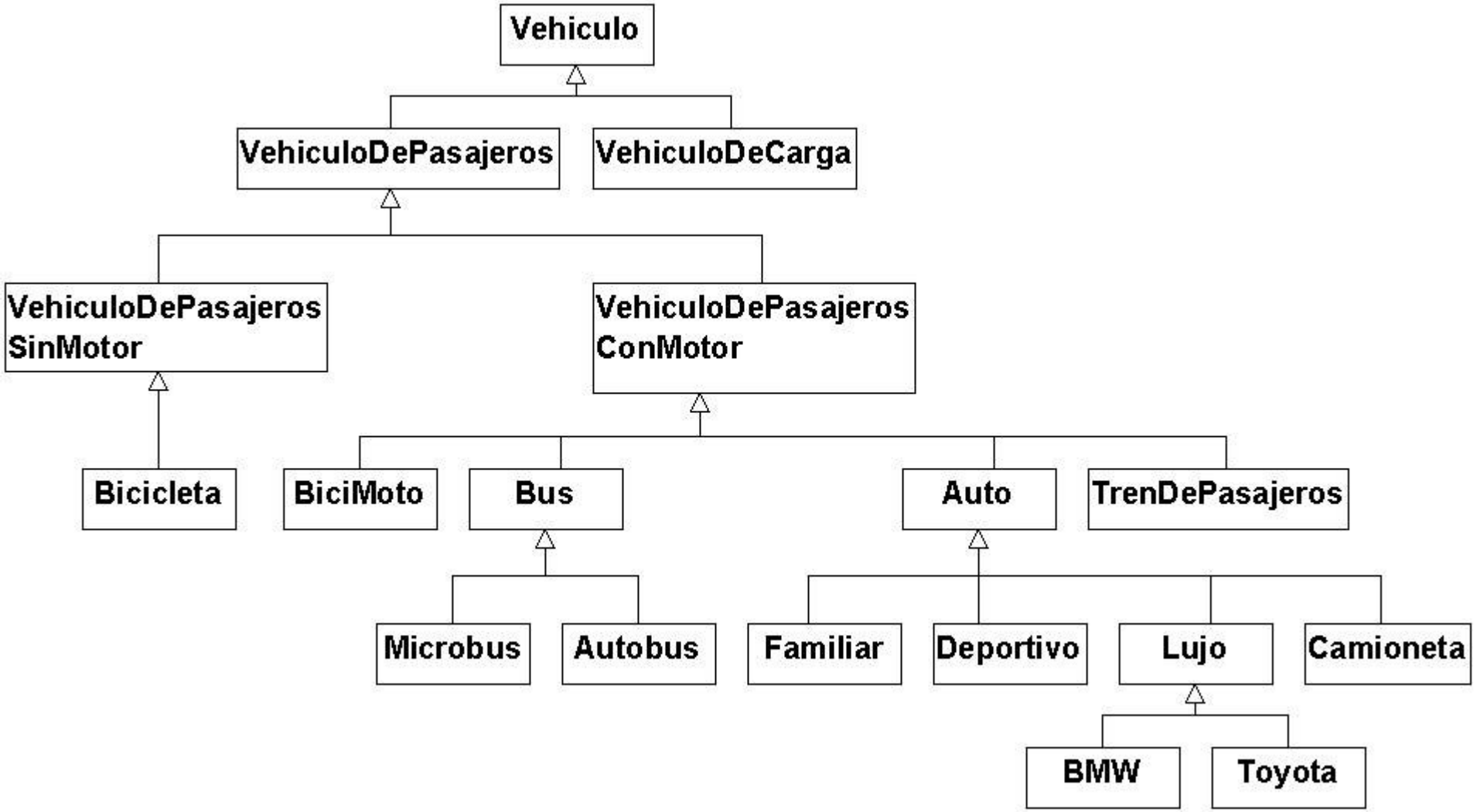
- La especialización captura características específicas de un conjunto de objetos que no han sido distinguidas por las clases de nivel superior.

- La generalización es un relación "no reflexiva" es decir una clase no se deriva de ella misma.

- La generalización es asimétrica, es decir si B se deriva de A, A no se puede derivar de B.

Jerarquía de Clases: Generalización / Especialización

- Ejemplo 1



Superclases y Subclases

- Las clases con propiedades comunes se organizan en superclases.
- Una superclase es una generalización de las subclases.
- Una subclase representa una especialización de la superclase.
- La subclase hereda atributos y comportamientos de la superclase.
- Subclase: hija, derivada
- Superclase: padre, base