



# Clase #5

## Modelado Básico de Objetos

Prof. Andrés Arcia  
Departamento de Computación  
Escuela de Ingeniería de Sistemas  
Facultad de Ingeniería  
Universidad de Los Andes

# Qué es Orientación por Objeto?

- Si hablamos de un software orientado por objetos, puede verse como una colección de objetos, donde cada uno está formado por estructura de datos más la especificación de cada una de esas estructuras.
- Ejemplo de objetos:
  - Una lápiz, una bicicleta, un elefante.
  - El segundo párrafo de la página 1 del cuento “El Principito”, El título de la obra más importante de D. Knuth “El arte de la programación”.
  - Una lista de elementos del tipo cadena, un arreglo de enteros.

# Cómo se concibe a un objeto?

- ♦ Un objeto es un ejemplo de una clase de objetos. Se pueden especificar atributos generales de una clase de objetos y luego ejemplificarlos (los objetos).
- ♦ Las clases de objetos se denominan *CLASES*.

























# Clases y Objetos

- ♦ La clase bicicleta es una definición abstracta, sin forma. Una bicicleta debe tener: Una biela, un cuadro, dos pedales, una o más ruedas, un volante, frenos.
- ♦ Un objeto rueda puede ser: Una bicicleta de montaña, Una bicicleta de ruta, Una bicicleta de velodromo, Una bicicleta de paseo para 2 personas, Una bicicleta especial de

# Un par de objetos bicicleta



# Atributos de la Clase Bicicleta

 Cuadro	 Cambio
 Horquilla	 Mandos
 Rear Shock	 Manillar
 Llantas	 Potencia
 Bujes	 Juego de dirección
 Radios	 Frenos
 Cubiertas	 Manetas de freno
 Pedales	 Sillín
 Bielas	 Tija del sillín
 Cadena	 Colores
 Piñiones	 Tallas
 Eje de Pedalier	
 Desviador	



# Operaciones de la Clase Bicicleta

- ♦ Hacer cambios de la velocidad.
- ♦ Frenar.
- ♦ Mover el volante a la izquierda.
- ♦ Mover el volante a la derecha.
- ♦ Ajustar freno.
- ♦ Observar el presión de aire de las llantas.
- ♦ Ajustar altura del asiento.

# Metodología de OxO: OMT

- ♦ **Análisis:** Desde la descripción del problema se pretende construir un modelo de la realidad del problema.
- ♦ **Diseño del Sistema:** Decidir como se organiza el modelo, cuales son los subsistemas. Se corrigen aspectos de la visión del sistema.



# Metodología de OxO: OMT

- ♦ Diseño de Objetos: Se decide cuales serán los objetos y como se interrelacionan. Se pone atención a: estructuras de datos, algoritmos, interfaces.
- ♦ Implementación: Se lleva a cabo en un lenguaje específico (C++ por ejemplo), lo antes descrito.

# Diagramas

- ♦ D. de Objetos: Proporcionan grafismos formales para modelar objetos, clases y sus relaciones entre si.
- ♦ D. de Clases: Resume todas las posibles instancias de objetos.
- ♦ D. de Instancias: Especifica como se relacionan un conjunto de objetos.

# Ej. de Diagramas



Figura  
Geométrica

A UML class diagram element representing a class. It is a light gray rectangle with a thin black border. The text 'Figura' is on the top line and 'Geométrica' is on the bottom line, both in blue font.

Clase



(Fig. Geometrica)  
Triangulo

A UML object diagram element representing an object. It is a light gray rounded rectangle with a thin black border. The text '(Fig. Geometrica)' is on the top line and 'Triangulo' is on the bottom line, both in blue font.



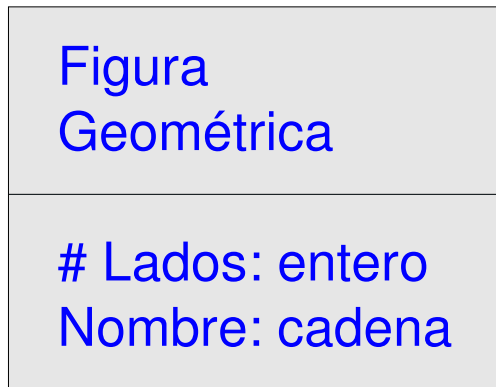
(Fig. Geometrica)  
Cuadrado

A UML object diagram element representing an object. It is a light gray rounded rectangle with a thin black border. The text '(Fig. Geometrica)' is on the top line and 'Cuadrado' is on the bottom line, both in blue font.

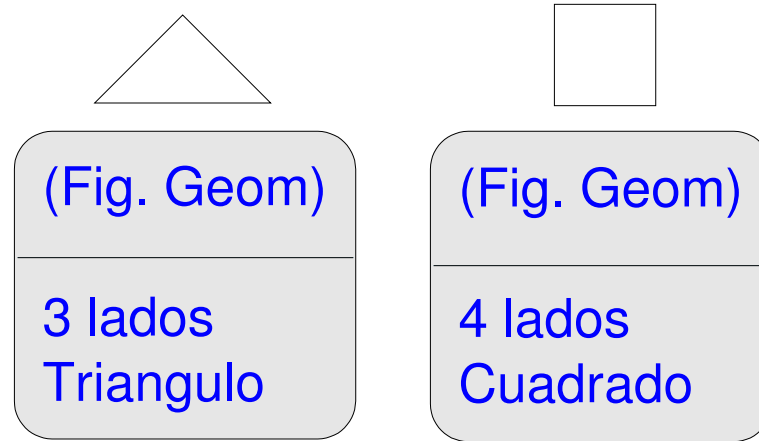
Objetos

Podría añadirsele atributos, que de ninguna manera deben confundirse con otros objetos, deberían ser valores puros de datos. Un valor posee identidad cuando es único. Ej. Radio del rin de una rueda es un atributo, mientras que Marca de la bicicleta no lo es. Piense que la marca puede convertirse en el objeto compañía.

# Diagramas para Atributos



Clase + Atributos



Objetos + Valores

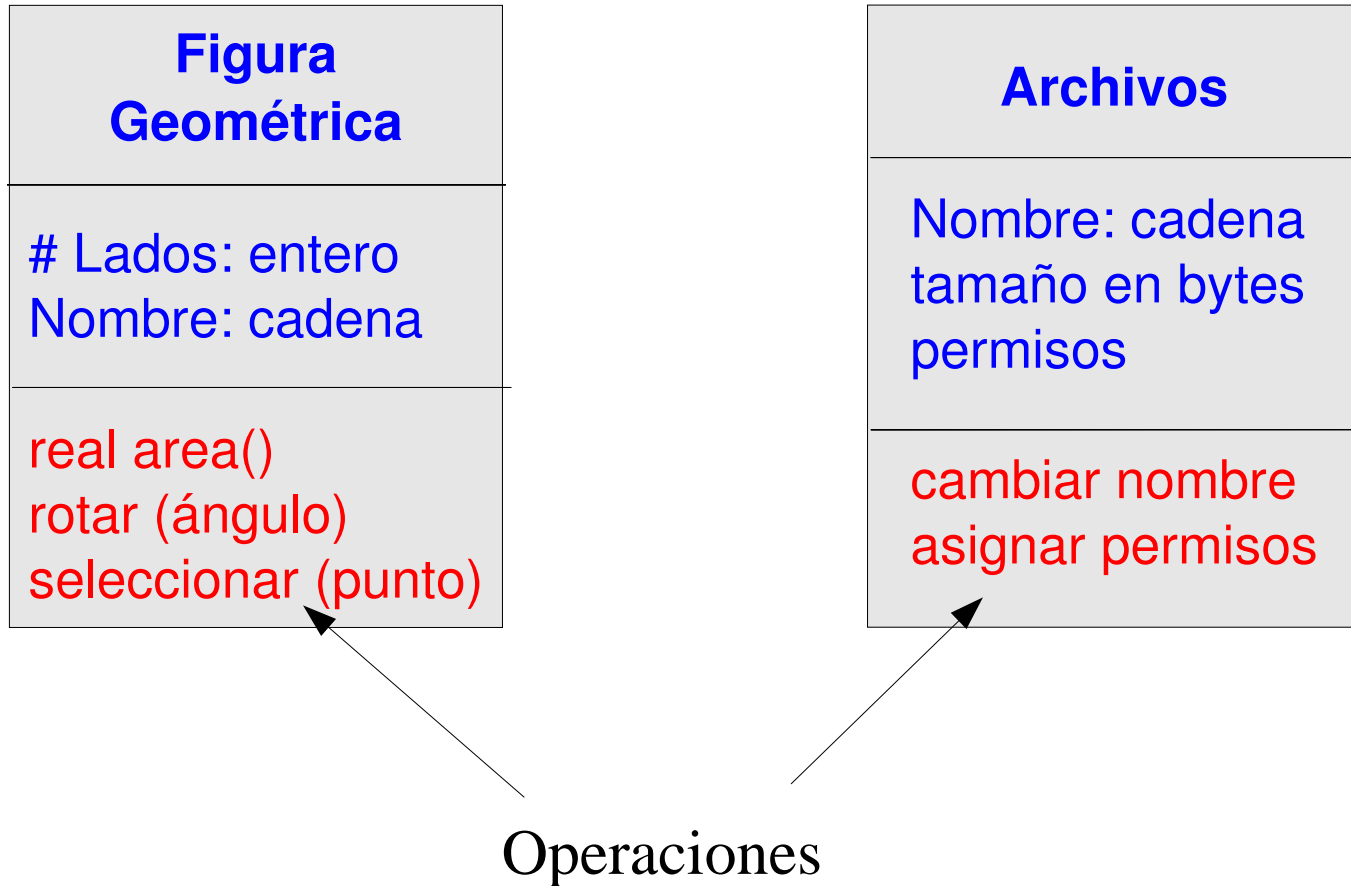
# Sobre los atributos

- ♦ La mayoría de las veces no hace falta escoger identificadores únicos para los objetos, los lenguajes de prog. los proveen automáticamente.
- ♦ Los Id únicos son simplemente una comodidad.

# Operaciones y métodos

- ♦ Las operaciones y los métodos son las acciones que permitirán darle sentido a los datos de los objetos.
- ♦ Pueden ser referidas como acciones genericas o generales, luego se podrá especificar su conducta.

# Ejemplo de Operaciones



# Modelo para la notación de Clases

## Nombre de la Clase

nombre del atributo 1: tipo = valor por omisión.  
nombre del atributo 2: tipo = valor por omisión.  
...

nombre del método 1 (argumentos): tipo resultado.  
nombre del método 2 (argumentos): tipo resultado.  
...



# Asociaciones

- ♦ Mediante enlaces y asociaciones se pueden establecer las relaciones entre las clases que modelan al sistema.
- ♦ Representan conexiones físicas o conceptuales.
- ♦ En el argot matemático se conocen como *tuplas*, ó listas de elementos relacionados.
- ♦ Son bidireccionales.

# Asociaciones

- ♦ Suelen ser los *apuntadores* en los lenguajes de programación. Sin embargo, no piense en ellos mientras está modelando.

# Asociaciones Uno a Uno

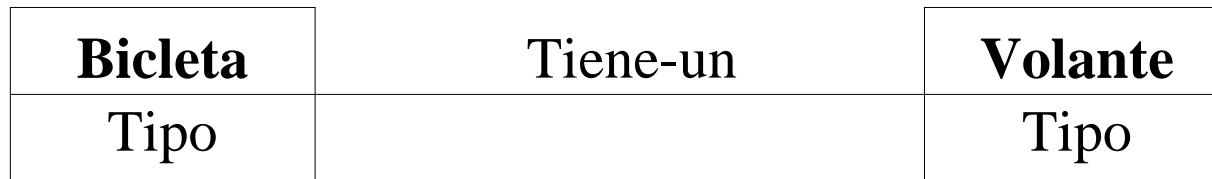


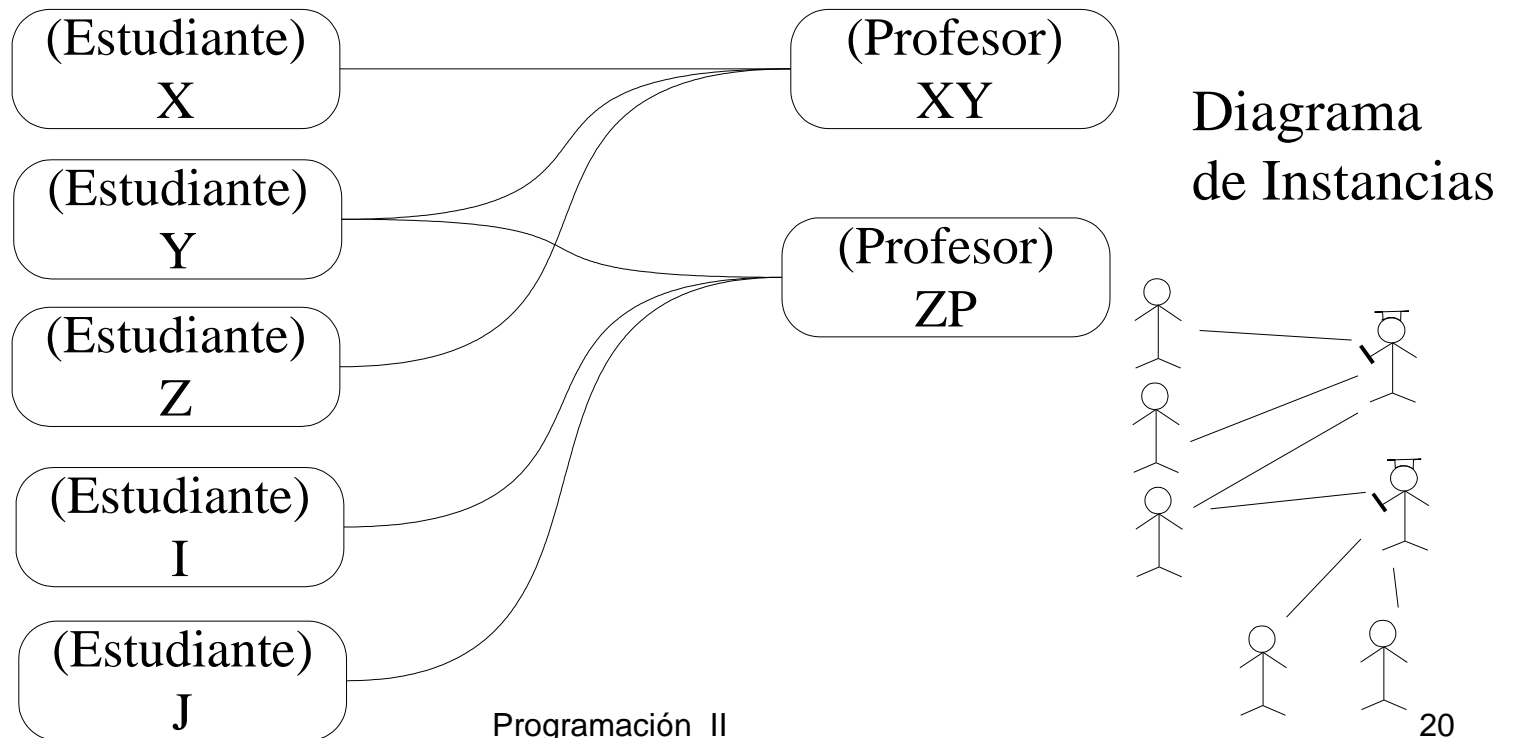
Diagrama  
de Clases



Diagrama  
de Instancias

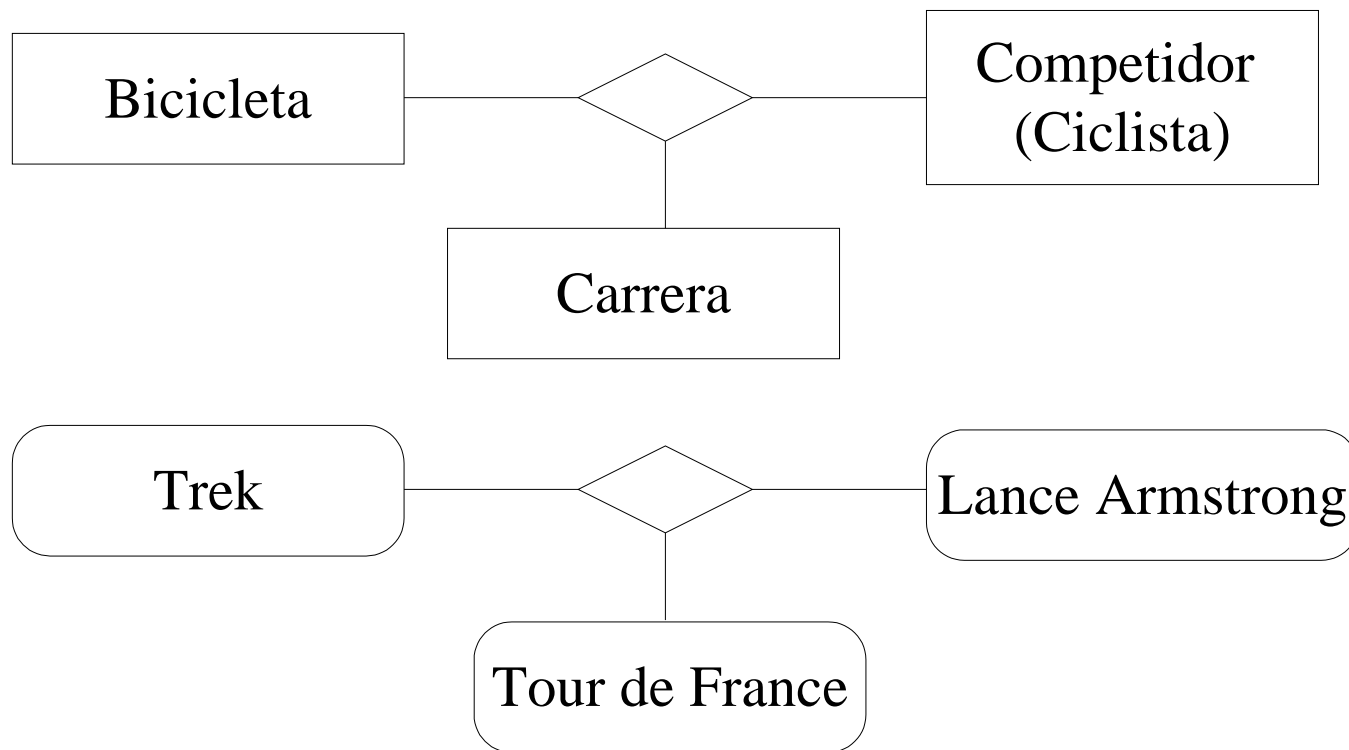


# Asociaciones muchos a muchos



# Asociaciones Ternarias

Observe que para que este tipo de asociación tenga sentido, ninguno de las tres clases puede estar fuera.



# Importancia

- ♦ Es utilizado en muchos campos, Ej. Las Bases de Datos.
- ♦ Puede ser utilizado en muchos otros campos: modelado de aspectos de la vida real, programación OoO, etc.
- ♦ No es una manera infalible de modelar, hay quienes opinan que las relaciones entre clases no deben existir.

# Multiplicidad

- ♦ Dice que cantidad de instancias de clase están relacionadas con otras clases de relación explícita.
- ♦ Expresa una cantidad finita o infinita de elementos relacionados.
- ♦ Hay símbolos especiales para expresar la relación de multiplicidad.

# Multiplicidad Cero o Uno



Un auto puede tener o no un triángulo de seguridad.

Para poder escribir la multiplicidad, primero se determinan: objetos, clases y asociaciones.

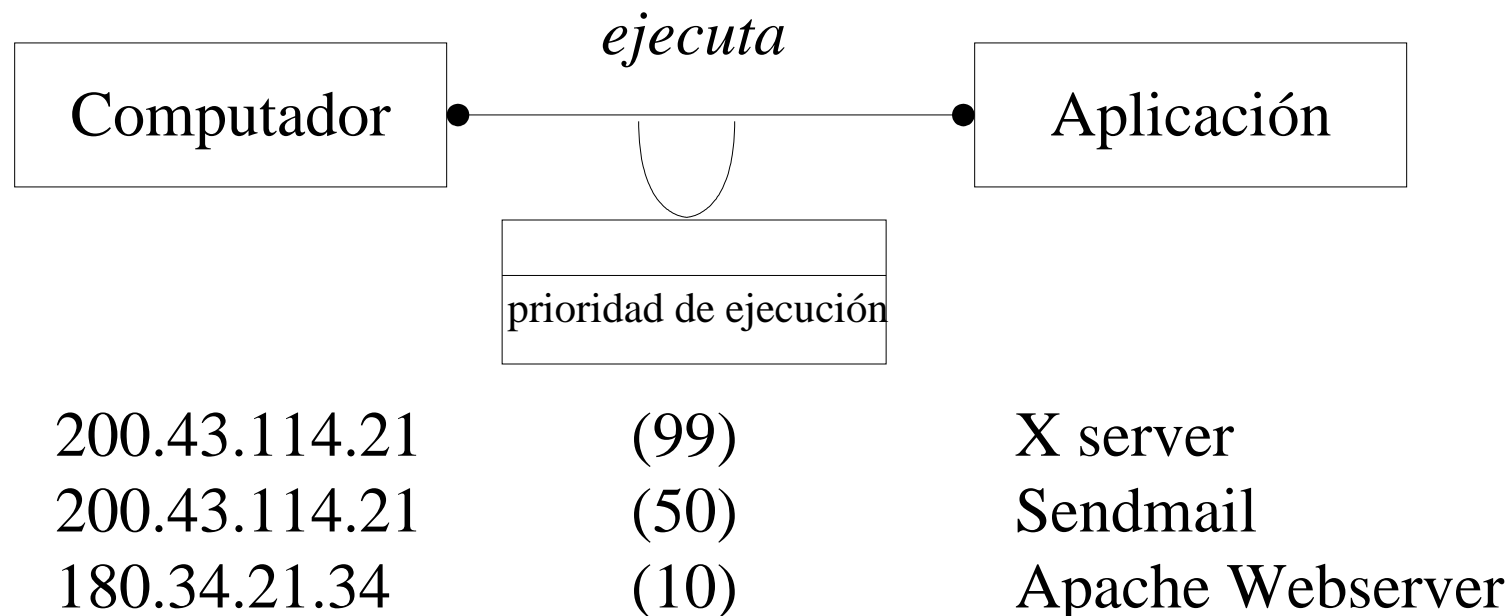


# Atributos en los Enlaces

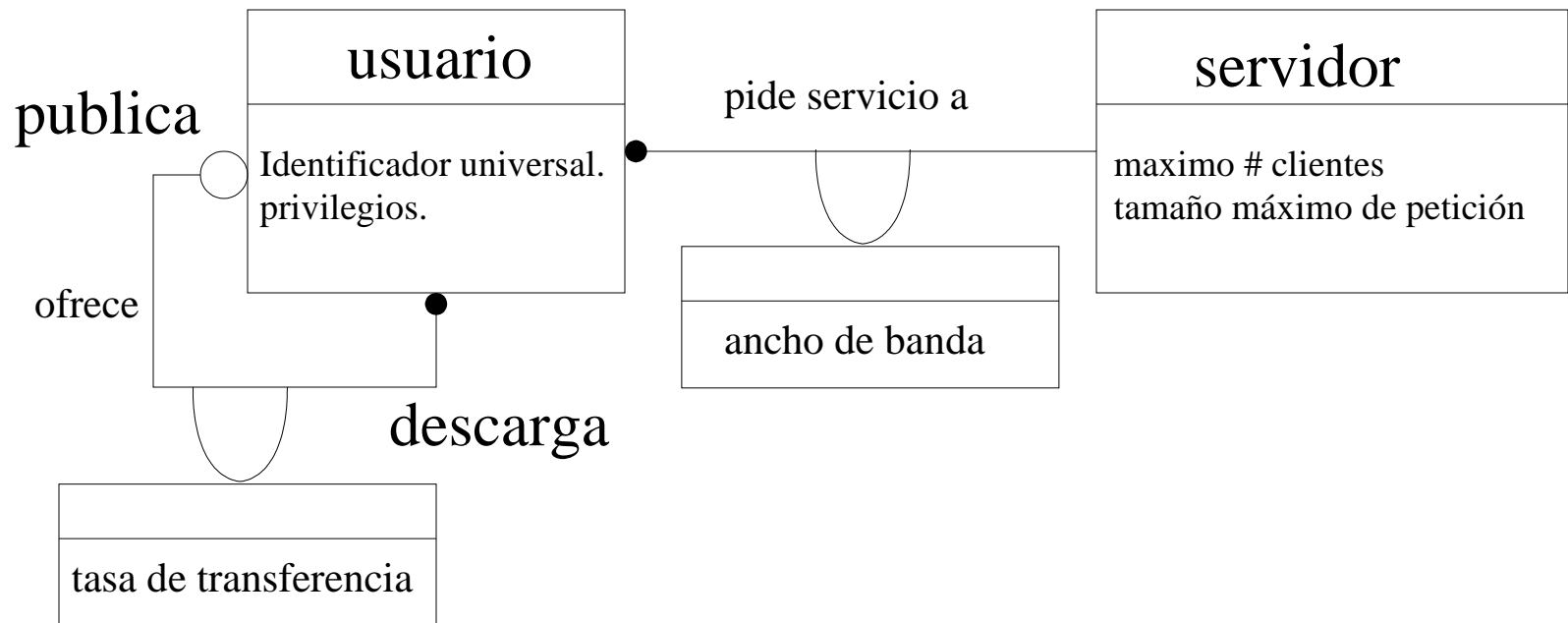
- ♦ Un atributo es un descriptor inherente a los objetos de una clase. Así mismo, un *atributo de enlace* es un descriptor inherente a los enlaces de asociación.
- ♦ Vea el gran parecido entre los atributos de objetos y atributos de enlaces.
- ♦ Las asociaciones del tipo muchos a muchos proporcionan el escenario ideal para colocar atributos a los enlaces.

# Atributos en los Enlaces

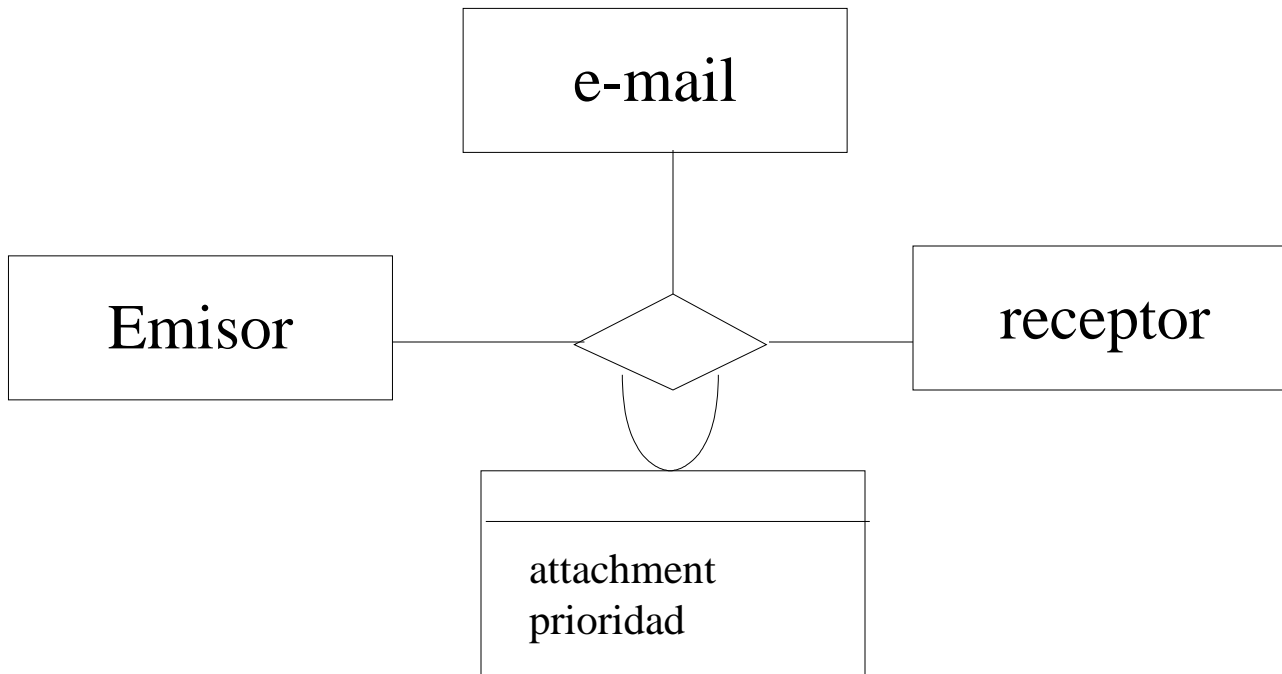
- ♦ Los atributos son propiedad del enlace, y pierde sentido (información) si se asociara a alguno de los objetos.



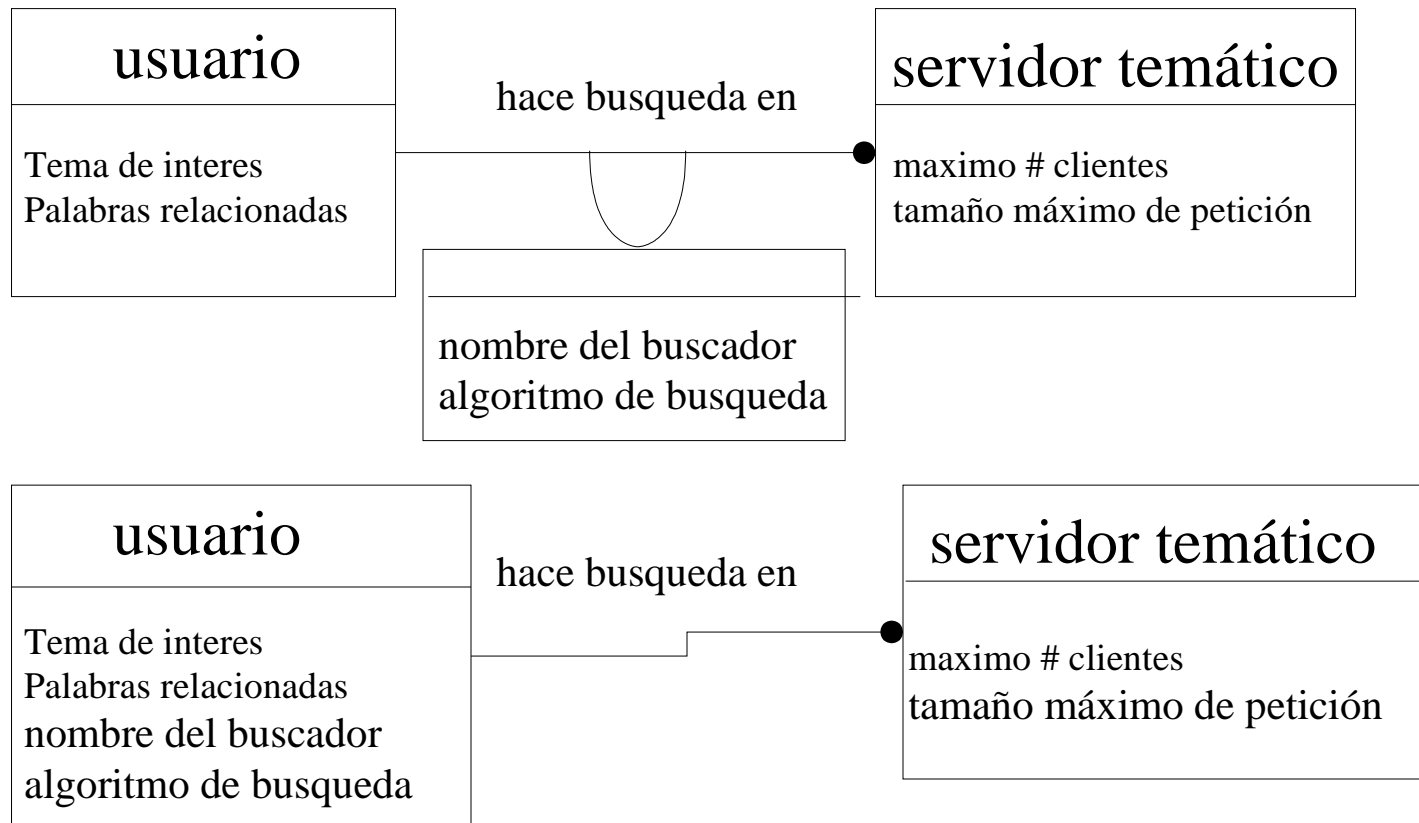
# Atributos dentro de Asoc. uno a muchos



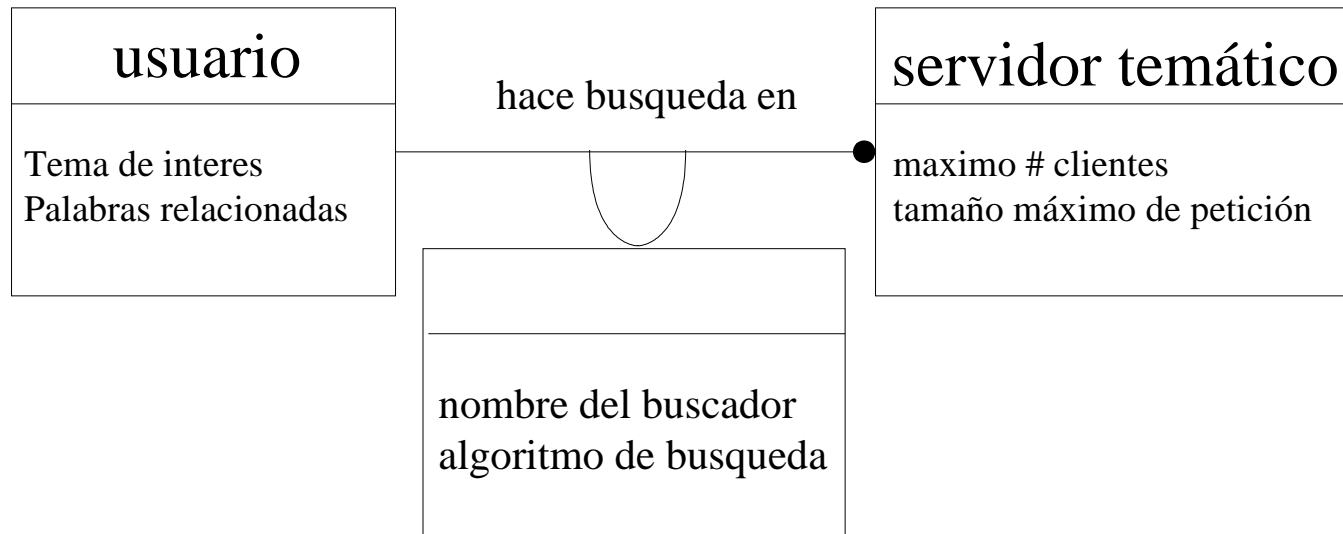
# Atributos para los enlaces en asociaciones ternarias



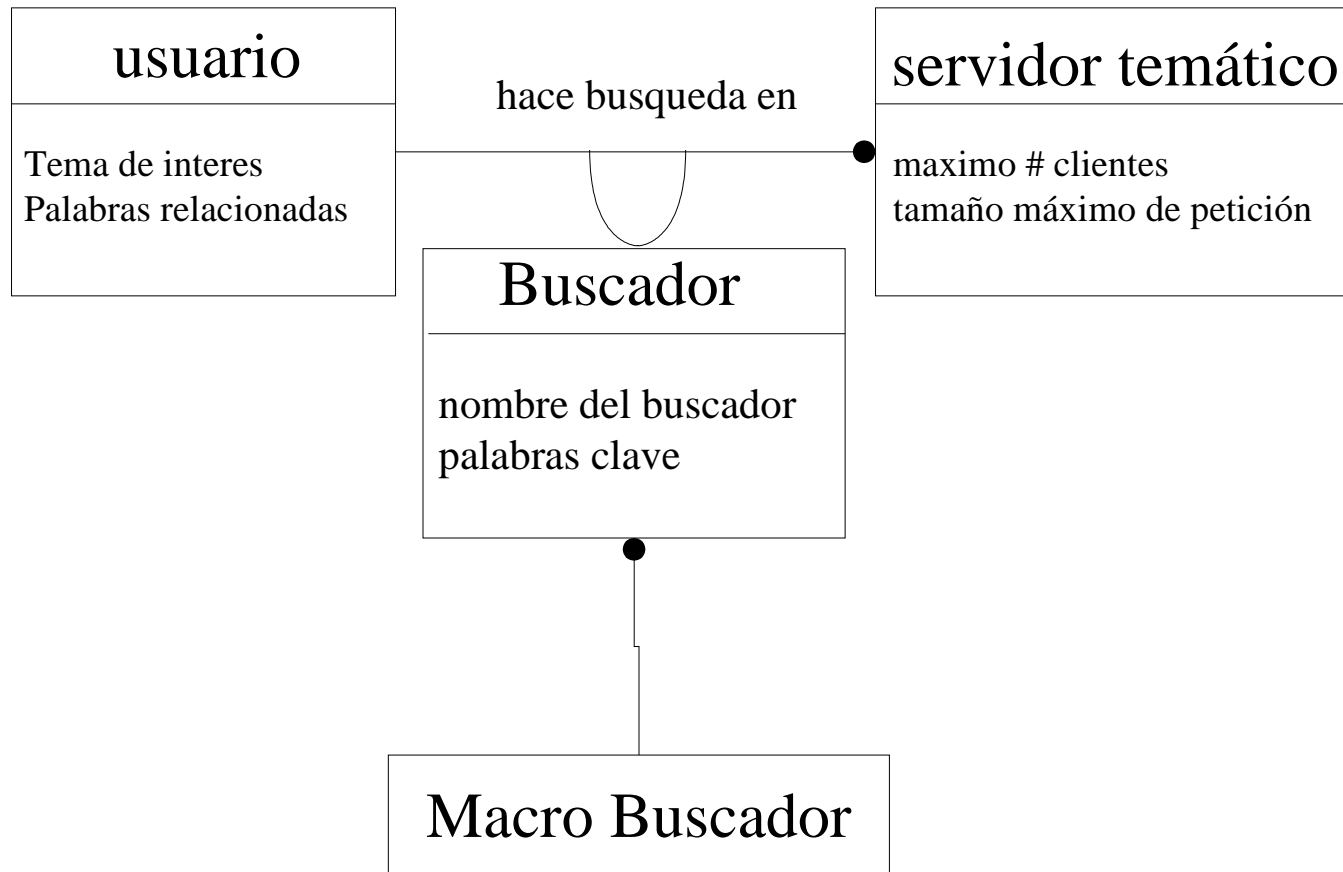
# Atributos de enlace vs. atributos de objetos.



# Asociaciones en Forma de Clase



# Asociaciones en Forma de Clase



# Nombres de Rol

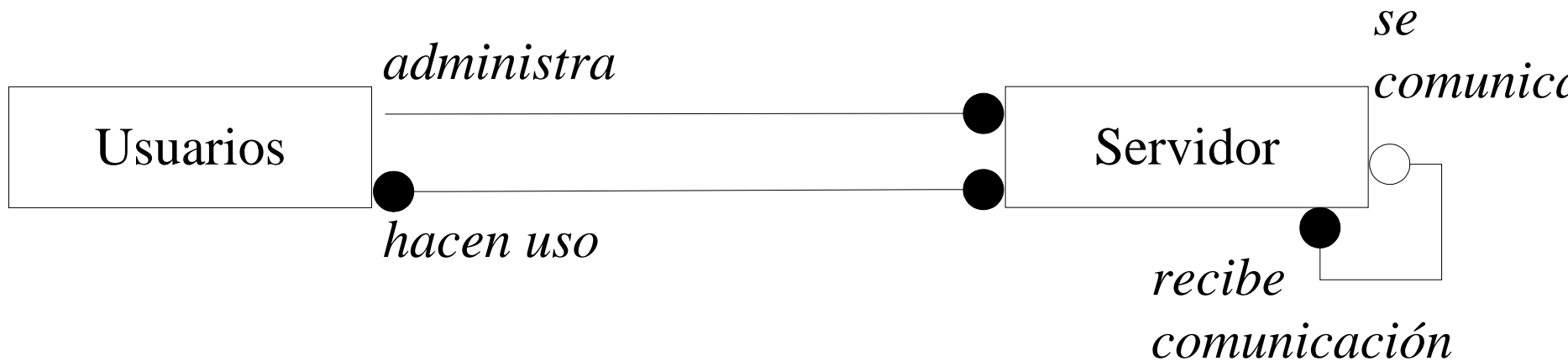
- ♦ Un rol es se situa en un extremo de la asociación y la identifica univocamente.
- ♦ Se denomina binario si existen valores en ambos extremos de la asociación.





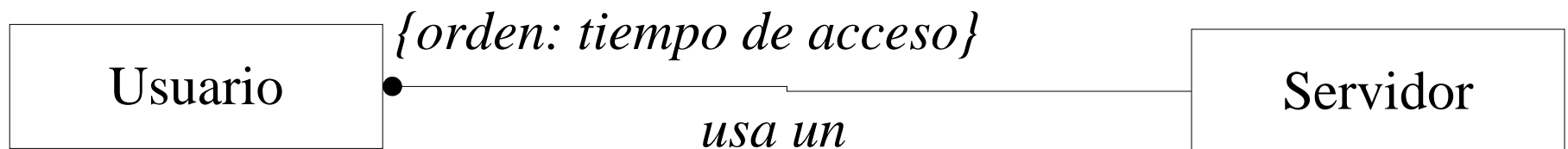
# Nombres de Rol

- ♦ Un rol es un atributo propio de la clase que contenga la frase que describa el rol.



# Clasificación

- ♦ Los objetos en el lado “muchos” no tiene un orden explicito y se puede calificar de conjunto.
- ♦ Esto nos dice que es posible expresarlo explicitamente.



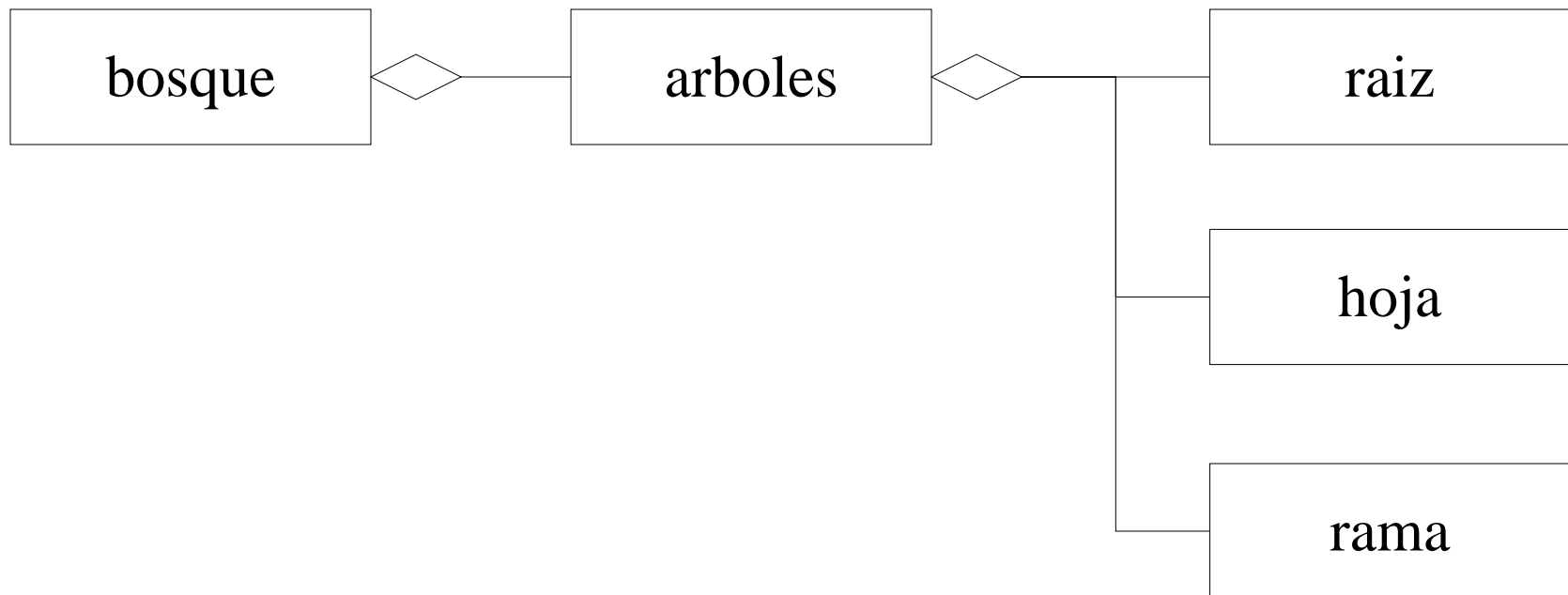
# Agregación

- ♦ Varios componentes más pequeños y de cierta forma independientes forman uno más grande (el conjunto forma un ente **sinérgico**). Por ejemplo: las funciones, argumentos, bloques, son componentes agregados a un programa.

# Agregación

- ♦ Hay un par de propiedades que se cumplen con la agregación:
  - ✧ Transitividad: Si  $A \supset B$  y  $B \supset C$ , entonces  $A \supset C$ .
  - ✧ Antisimetría: Si A es parte de B, B no es parte de A.
- ♦ Para utilizar la agregación, deben existir propiedades comunes de los componentes que luego permitan hacer ***sinergia***.

# Ejemplo de Agregación (controversial)



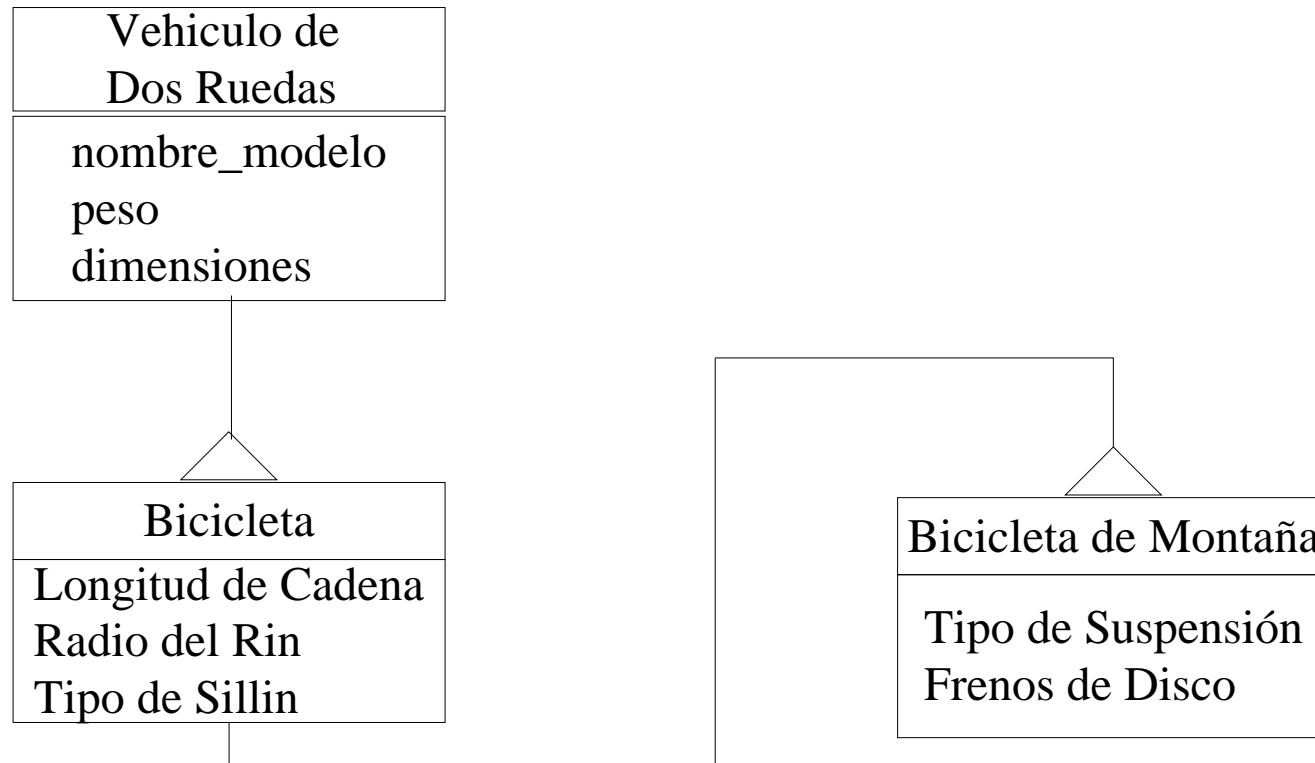
# Ejercicios propuestos

- ♦ Investigue a fondo los siguientes objetos y trate de establecer un modelo OMT, trate de profundizar en cada uno de ellos:
  - ★ Computador
  - ★ Bicicleta
  - ★ Automovil
  - ★ Una carrera universitaria
  - ★ Lápiz (parece simple, pero pruebe a ver...)

# Generalización y Herencia

- ♦ Generalización: Es la relación entre una clase y una o más versiones refinadas de esa misma clase. Cuando se refina, la nueva clase, será una **subclase** que partio de una **superclase**. Se dice que cada subclase hereda las características (datos + operaciones) de la superclase.

# Ejemplo





# Herencia: Reutilización de Código

- ♦ Anulación: Una subclase puede anular (to override) todas o algunas características de una superclase si se asigna igual nombre. Esto se hace para obtener una conducta más parecida a las necesidades de la clase refinada (subclase).

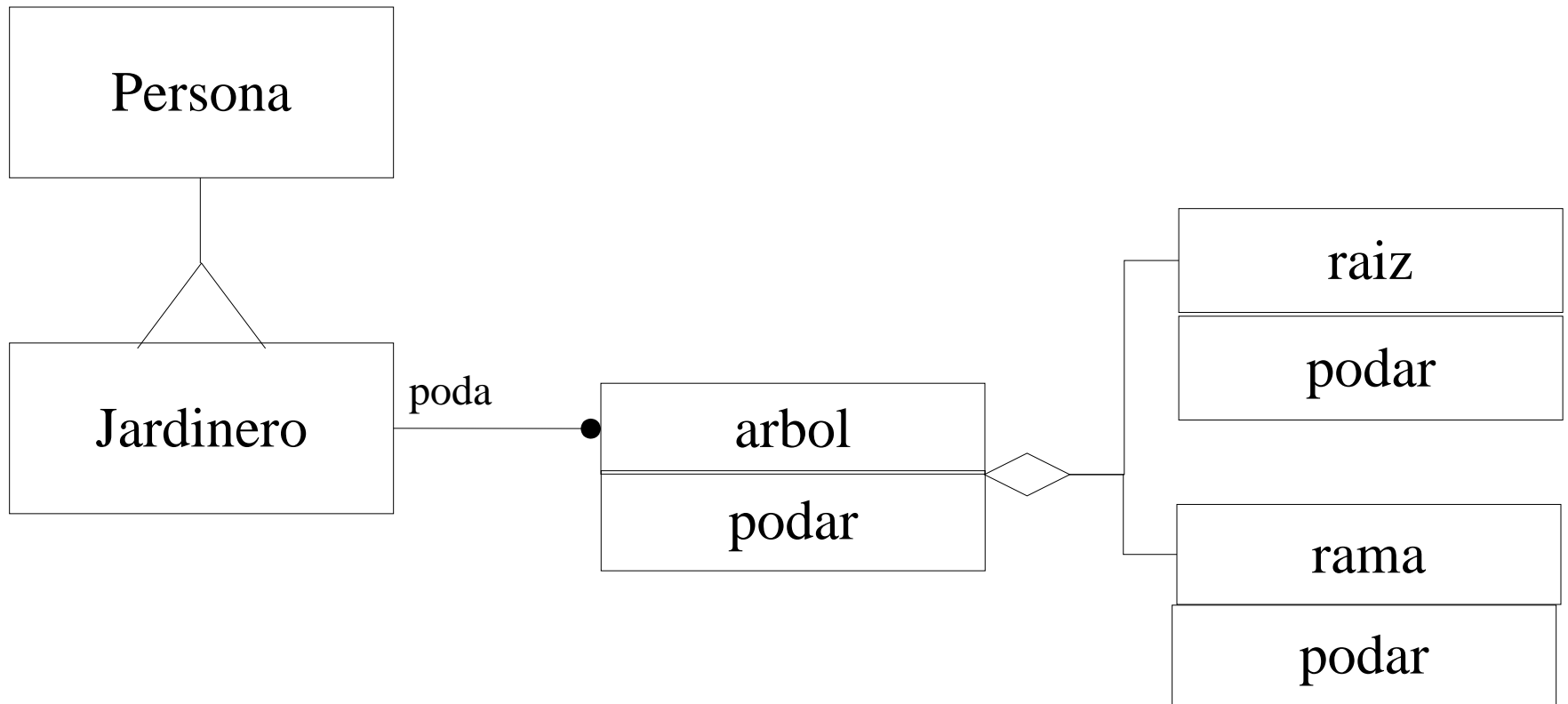
# Modelo de Objetos

- ♦ Modulos = clases + asociaciones + generalizaciones + bloques de construcción (clases + asociaciones).
- ♦ Hojas: El grafismo de un modelo completo ocupa un espacio que a veces se excede del espacio donde se exhibirá. Para obtener trozos más pequeños se agrupan por hojas.

# Consejos para programar a Objetos

- ♦ Comprenda bien el problema.
- ♦ Haga el máximo esfuerzo para que el modelo sea sencillo y completo.
- ♦ Adopte un sistema de nombramiento.
- ♦ No ate su modelo a un lenguaje de programación específico.
- ♦ Evite generalizaciones muy anidadas.
- ♦ Revise varias veces el modelo.
- ♦ Documente bien su modelo.

# Ejemplo





# Clase Abstractas

- ♦ Es una clase que no tiene instancias directamente. Es el contrario de una clase concreta.
- ♦ Generalmente se utilizan para definir interfaces generales que podrían ser especializadas en subclases.

# Herencia Múltiple

- ♦ Permite que una clase tenga más de una superclase.
- ♦ Este es un mecanismo que hace más poderosa la especificación de las clases.

