UNCUYO
UNIVERSIDAD NACIONAL DE CUYO

FACULTAD DE INGENIERÍA

Programación Orientada a Objetos

Relaciones entre Clases

Esp. Ing. César Aranda

unidatos@gmail.com
cesar.aranda@ingenieria.uncuyo.edu.ar

Ingeniería en Mecatrónica

Objetivos y contenidos

- Analizar diagramas de clases en UML que incluyan relaciones de Herencia, Agregación y Composición
- Comprender las relaciones de Herencia y los mecanismos de implementación asociados
- Comprender las relaciones que impliquen colecciones de objetos
- Analizar diagramas de clases en UML que incluyan diferentes tipos de relaciones
- Discutir aspectos básicos de diseño

Relaciones entre Conceptos

- ✓ Relaciones Simples (dependencia de uso o de interacción)
- ✓ Relaciones de Asociación
- ✓ Relaciones de Jerarquías (Herencia y Agregación)

La mente humana clasifica los conceptos de acuerdo a dos dimensiones:

- Pertenencia
  - Una Rueda es parte de un Automóvil
  - Relación de la forma ***pertenece a*** (has_a)
- Variedad
  - Un Ford Fiesta es un tipo de Automóvil
  - Relación de la forma ***es un*** (is_a)

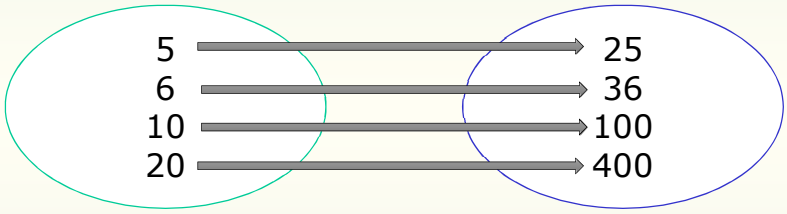
Vínculo

Un **Vínculo** es la relación que se establece entre 2 objetos al comunicarse entre sí mediante uno o más mensajes

5 al cuadrado es 25
6 al cuadrado es 36
10 al cuadrado es 100
20 al cuadrado es 400
…

Relación

Una **Relación** es la abstracción del conjunto de vínculos.



X R Y

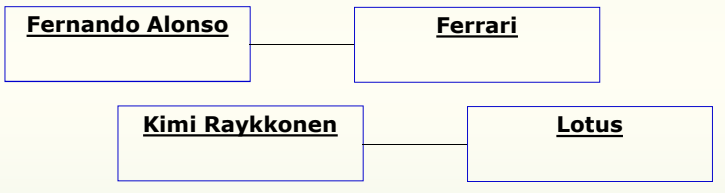
Un número es el cuadrado de otro

$$Y = f(X) = X^2$$

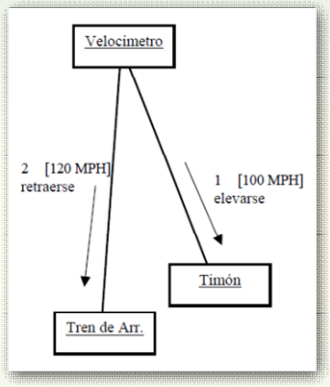
Asociación

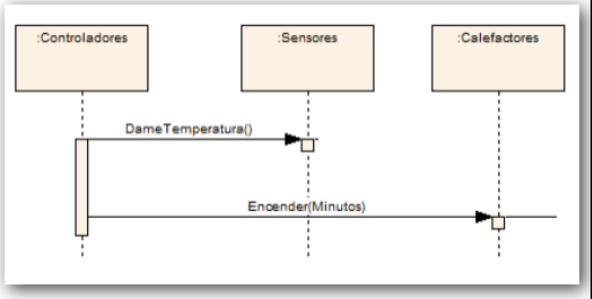
Expresa una conexión bidireccional entre objetos.

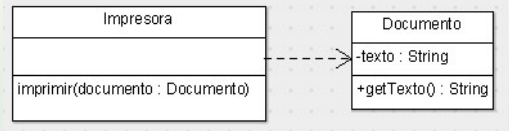
Es una abstracción de la relación que surge por los enlaces (vínculos) entre los objetos.

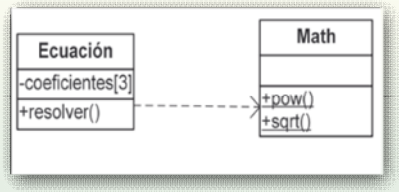


Relación de Asociación Simple (uso/dep.)

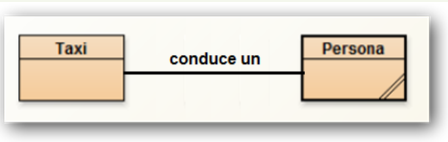


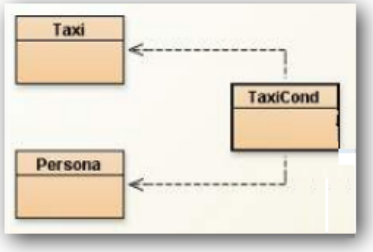


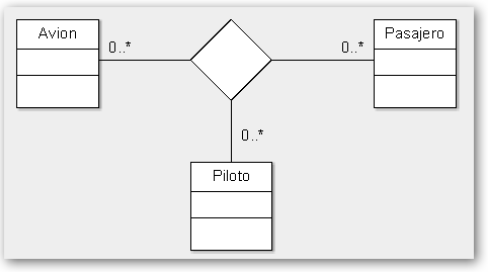




Relaciones de Asociación con Clases







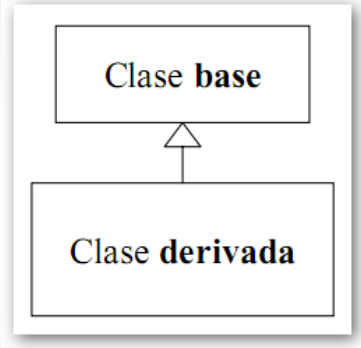
Herencia

- Es un mecanismo de la OO
- Permite clasificar los tipos de datos (abstracciones) por variedad
- Acerca la programación al modo de razonar humano
- Permite definir una clase modificando una o más clases ya existentes
- Reduce los tiempos y esfuerzos de programación y prueba

La clase de la que se parte en este proceso recibe el nombre de ***clase base*** (madre o superclase)

- La nueva clase obtenida se llama ***clase derivada*** (hija o subclase)

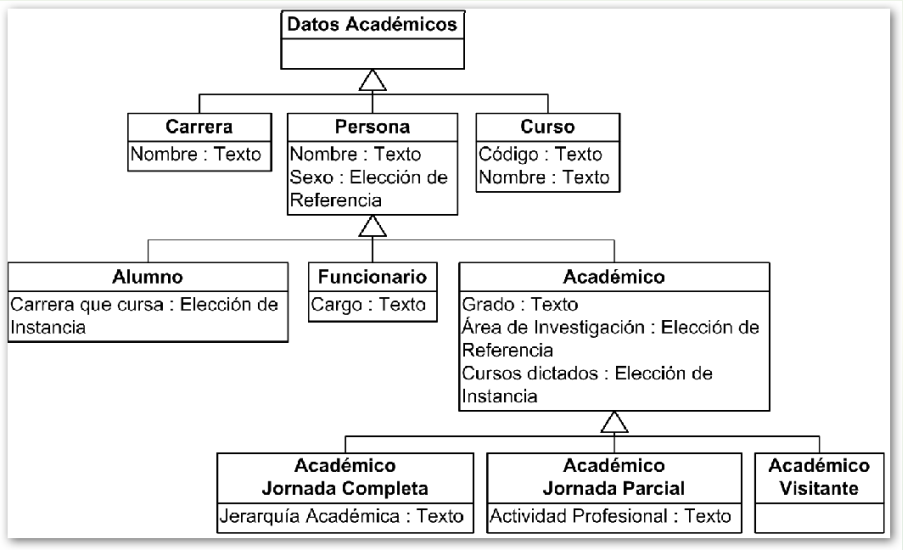
Clase Derivada Vs. Clase Base



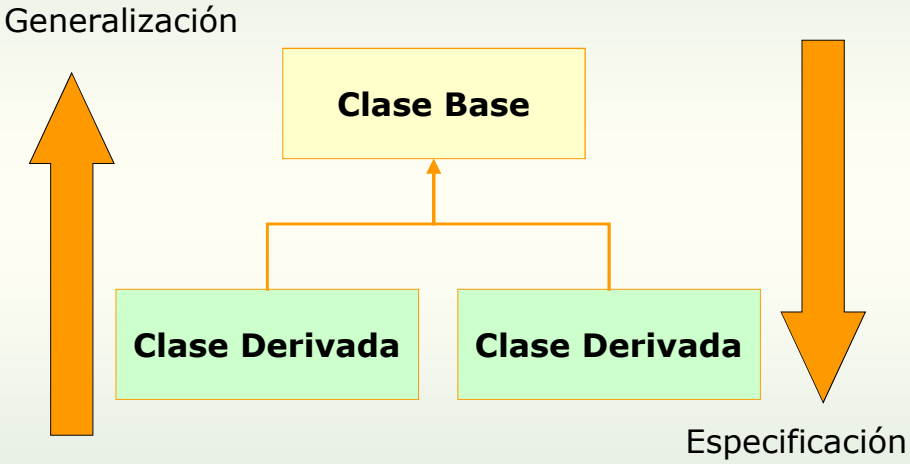
El proceso de herencia no afecta de ningún modo a la clase base

- La clase derivada ***hereda todas las características*** de la clase base.
- La clase derivada ***puede definir características*** adicionales.
- La clase derivada ***puede redefinir características heredadas*** de la clase base.
- La clase derivada ***puede anular características heredadas*** de la clase base.

Jerarquía de Clases = Ontología

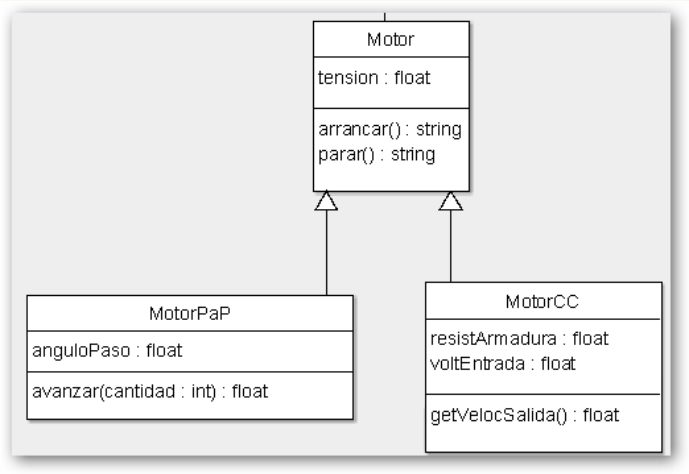


Sentido del Proceso de Abstracción



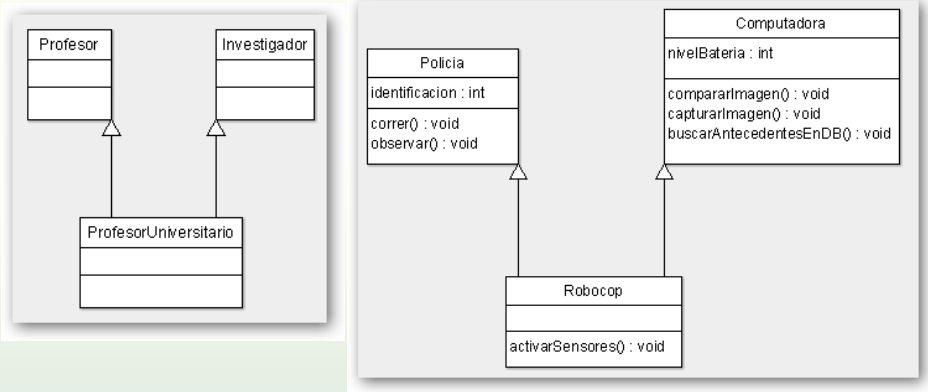
Herencia Simple

- Cuando una Clase Derivada se crea a partir de una ***Única Clase Base***

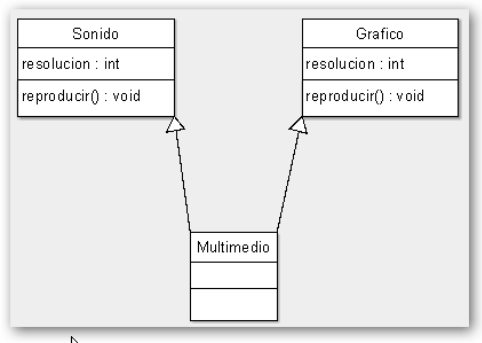


Herencia Múltiple

- Cuando la Clase Derivada se obtiene de ***Dos o Más Clases Base***



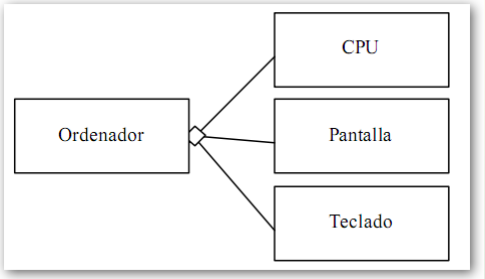
Dificultades de la Herencia Múltiple



- Herencia repetida
  - Por ej.: "Multimedio hereda 2 veces el atributo resolución"
  - Produce ambigüedad respecto a los atributos o los métodos.
  - Aumenta el tiempo de ejecución como consecuencia de tener que resolverse las colisiones.

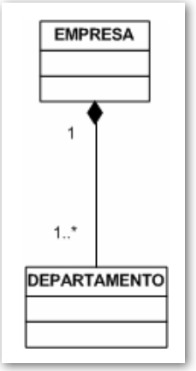
Agregación

- Representa una relación de tipo "parte de" entre objetos.
- En UML el símbolo de agregación es un rombo colocado en el extremo donde está la clase que representa el "todo".



Composición (o Agregación Inclusiva)

- Es una agregación con restricciones.
- Si un objeto parte no puede comunicarse directamente con objetos externos distintos al objeto agregado se dice que la misma es una relación de *agregación inclusiva*.
- En UML se simboliza este caso con un rombo negro.
- El ejemplo indica una agregación inclusiva debido a que no le esta permitido en esta solución a un departamento tener relación por fuera del objeto agregado.



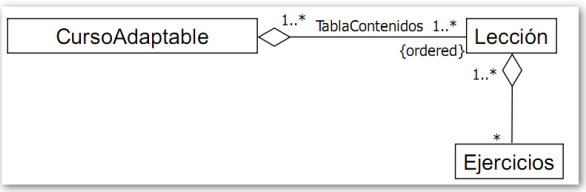
Agregación vs. Composición

- Agregación
  - Objeto agregado = Objeto construido a partir de otros.
  - El agregado es mayor que la suma de sus partes
  - Las interacciones del conjunto de objetos agregados se realizan a través de la interfaz del objeto agregado
  - Objetos componentes encapsulados dentro del agregado
- Composición
  - La composición es una forma fuerte de agregación
  - El ciclo de vida de las partes depende del ciclo de vida del agregado
  - Las partes no existen fuera de su participación en el agregado
  - La pertenencia fuerte implica objetos físicos que se unen para formar el compuesto

Agregación vs. Composición

- Agregación
  - Multiplicidad en las dos partes de la relación
  - Las partes pueden existir incluso después de que el agregado sea "desmontado" o destruido
  - Las partes pueden cambiar de un agregado a otro
- Composición
  - La multiplicidad en el "extremo" del compuesto es 1 ó 0..1
  - Multiplicidad en el "extremo" de las partes del compuesto
  - Si el agregado se "desmonta" o se destruye las partes no tienen existencia propia
  - Las partes no se pueden mover de una composición a otra

Ejemplo de Modelo con Agregación

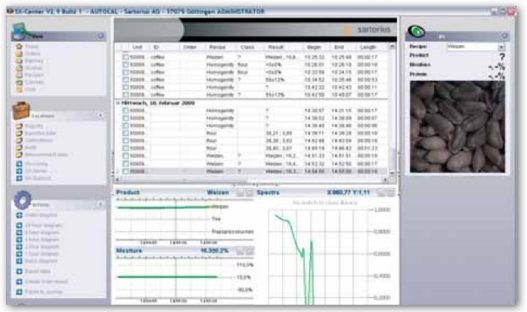


- Cursos adaptables. Se pueden crear combinando lecciones y ejercicios ya existentes creando una tabla de contenidos nueva
- La tabla de contenidos es única para cada curso
- La lección aparece en la tabla de contenidos para cada curso que la usa
- Las lecciones se desarrollan para un curso pero se pueden usar para construir otros cursos
- Los ejercicios se desarrollan inicialmente para un curso pero pueden ser utilizados con otras lecciones para otros cursos.

Ejemplo de Modelo con Composición

- Cursos fijos. Se crean y se entregan. Para crear un nuevo curso todo el material se crea desde de cero
- La tabla de contenidos es única para cada curso
- La tabla de contenidos hace referencia a cada lección del curso. Cada lección solamente aparece en la tabla de contenidos del curso para el que fue desarrollada
- Las lecciones se utilizan únicamente en el curso para el que fueron desarrolladas
- Los ejercicios se utilizan únicamente en las lecciones para las que fueron desarrollados

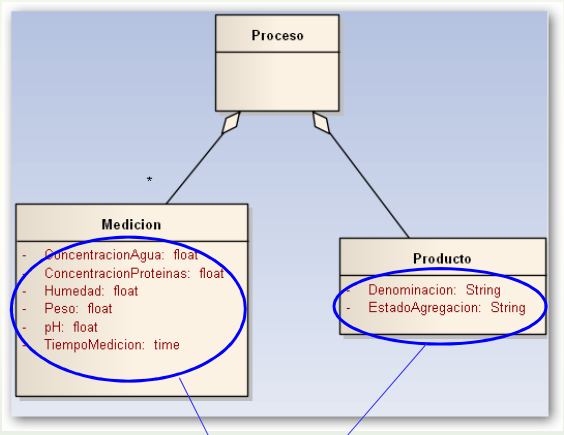
Caso de Aplicación en Metrología y Control: Análisis de parámetros físico-químicos



- Generalidades del Producto:
  - Nombre, estado de agregación, etc.
- Parámetros de Medición
  - Peso
  - Humedad %
  - pH
  - Concentración de componentes
    - agua, grasas, proteínas
  - Instante de la medición

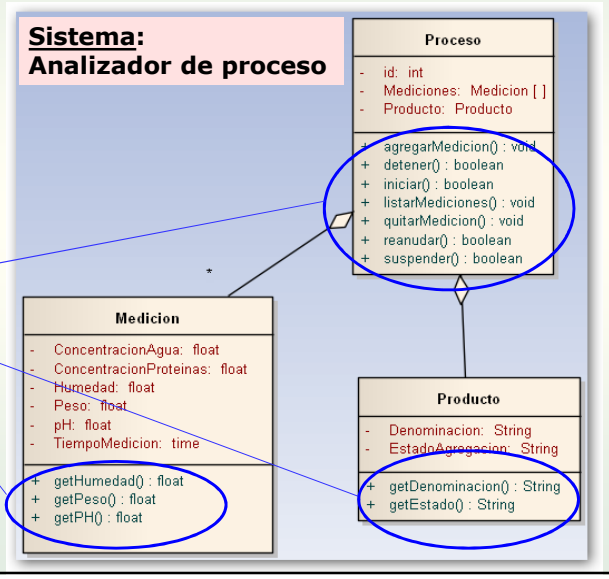
Modelo de Objetos como ED

Sistema: Analizador de proceso



Estructuras de Datos definidas por el programador

Modelo de Objetos: ED + Comportamiento



Funcionalidad definida por el programador

¿Cómo obtener el Diagrama de Clases?

1. Obtener el Conjunto de Clases Conceptuales Candidatas
2. Depurar el conjunto y obtener Clases Finales
3. Representar las Clases Conceptuales identificadas
4. Representar las Relaciones más importantes
5. Escribir un Nombre para cada relación
6. Representar Multiplicidad
7. Incorporar Atributos
8. Incorporar Operaciones
9. Incorpore información sobre Tipos de Datos y Argumentos

1. Obtener las Clases Conceptuales

Identificar frases nominales en descripciones del dominio

Frase nominal: Un sustantivo o un conjunto de palabras que actúan como tal

Por ejemplo, en el texto: "*Una cinta transportadora es un sistema de transporte continuo formado básicamente por una banda continua que se mueve entre dos cilindros.*"

Se observan los objetos: Cinta, Sistema de Transporte, Banda, Cilindros.

Alternativa: usar una Lista de Categorías

Categoría	Ejemplo
Objetos tangibles o físicos	Casa, Avión
Especificaciones, diseños o descripciones de las cosas	PlanoDeLaCasa, EspecificaciónDelProducto, DescripciónDelVuelo
Lugares	Tiendo, Aula
Transacciones	Venta, Pago, Reserva, Tranferencia
Líneas de la transacción	LíneaDeVenta
Roles de la gente	Cajero, Piloto, Jefe
Contenedores de otras cosas	Aula, Ciolectivo, Lata, Mochila
Contenidos	Pasajero, Artículo, ÚtilEscolar
Otros sistemas informáticos o electromecánicos externos al sistema	SistemaDeAutorización, ControlDeTra´ficoAéreo
Conceptos abstractos	Amor, Celos, Ansia, Acrofobia
Organizaciones	DepartamentoDeVentas, CompañiaAérea
Hechos	Reunión, Vuelo, Aterrizaje, Venta, Pago
Procesos (normalmente no se representan como conceptos, pero podría ocurrir)	VentaDeUnProducto, ReservaDeUnAsiento (no confundir con trasacciones)
Reglas y políticas	PolíticaDeReintegro, PolíticaDeCancelación
Catálogos	CatálogoDeProductos
Registros de finanzas, trabajo, contratos, cuestiones legales	Recibo, Remito, Factura, ContratoDeEmpleo, Expediente
Instrumentos y servicios financieros	LíneaDeCrédito, Stock
Manuales, documentos, artículos de referencia, libros	ManualDeReparaciones, ListaDeCambios
Relaciones	Amistad, Parentesco (podría estar en conceptos abstractos pero es bueno destacarlo ya que las relaciones no suelen ser consideradas al modelar)

2. Depurar el Conjunto de Clases

Incluir en el modelo aquellas que cumplan una o más de las siguientes propiedades:

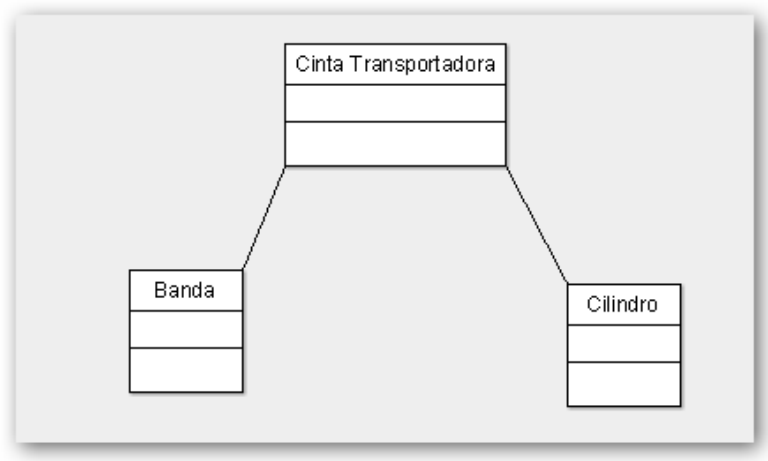
- Guardar información
- Necesidad de proveer un servicio
- Atributos múltiples
- Atributos comunes
- Operaciones comunes
- Requisitos esenciales

Eliminar las clases sinónimas

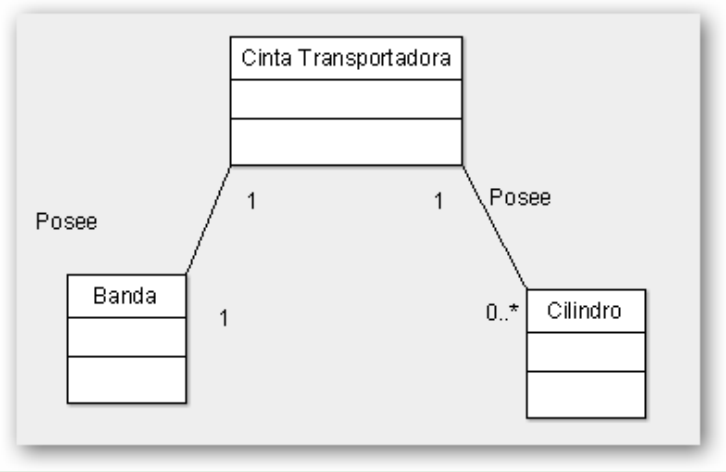
Eliminar las clases superfluas

En el ejemplo se suprime *Sistema de Transporte*

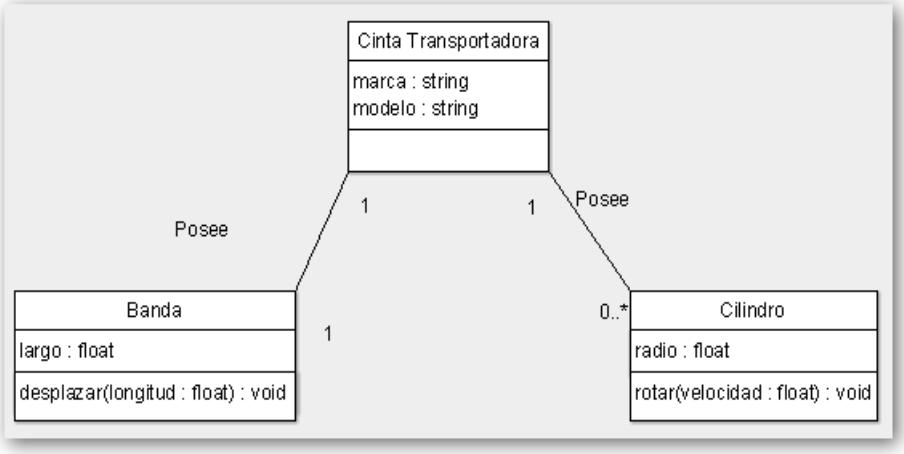
3, 4. Representar Clases y Relaciones



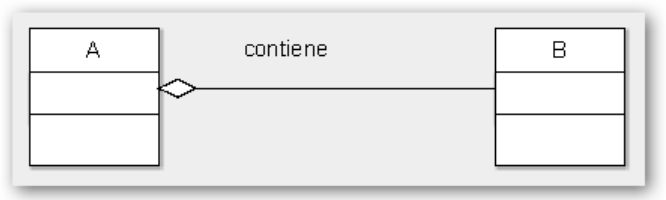
5,6. Indicar Relaciones y Multiplicidad



7,8,9. Añadir atributos, métodos y tipos



Agregación Bidireccional



public class A{

private B b[ ];

}

public class B{

opcional

private A a o public m (A a)

}

- El concepto componente se representa **siempre** como un atributo del concepto agregado.
- El acceso del componente hacia el agregado depende del diseño (con atributo o método).

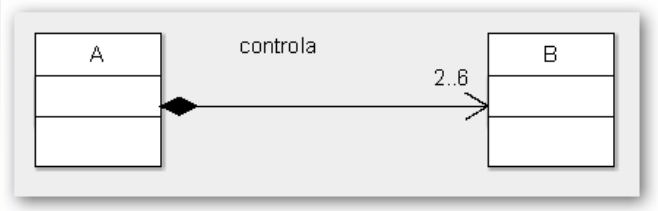
Agregación de Navegación Restringida


```
public class A{
private B b;
}
```

```
public class B{
...
}
```

- El concepto componente se representa **siempre** como un atributo del concepto agregado.
- El concepto componente **no accede** al agregado.

Composición Unidireccional




```
public class A{
private [ ] A b= new B[7];

        private
ctrl_Multiplicidad() {
        ...
        }

private ctrl_Integridad() {
                ...     }
}
```

```
public class B{
....
}
```

Lectura Complementaria y Referencias

BOOCH, G., RUMBAUGH, J., JACKOBSON, I. (2006): Lenguaje Unificado de Modelado, Manual de Referencia Uml 2.0, Addison-Wesley.

LARMAN, Craig (2003): UML y patrones. 2ª edición. Prentice Hall.

SCHMULLER, Joseph (1998): Aprendiendo UML en 24hs, Prentice Hall

MEYER, Bertrand (2000): Construcción de Software Orientado a Objetos, Prentice Hall. 2da edicion

BENNETT, Simon, MCROBB, Steve y FARMER, Ray (2007): Analisis y diseño orientado a objetos con UML, 3ra edición. McGraw Hill.

http://www.uml.org/

http://www.omg.org/