Introducción al Lenguaje de Modelado UML

Prof. Roxana Giandini

LIFIA - Facultad de Informática - UNLP

Lenguaje Unificado de Modelado

Modelos en Ingeniería

- Todas las ingenierías han usado y usan modelos científicos (matemáticos, físicos, etc.) para alcanzar mayor confiabilidad con mucho éxito.

- Sin embargo, en las ingenierías “clásicas” la transmisión de las intenciones de diseño pueden tener errores (malos materiales, personas con diferentes talentos que pueden comprender mal, etc.).

Modelos en Software

- Usamos modelos con diferentes propósitos:
- Para entender
- Para comunicar
- Para documentar, bosquejar y planear

Modelos como “Bosquejos”

- Para analizar y predecir

Modelos como “Guías”

- Eventualmente para producir software

Modelos como “Programas”

Modelado

- Ayudan a la comprensión de sistemas complejos
- Indican QUE hará el sistema pero NO COMO lo hará
- Ayuda a la corrección de errores
- Ayuda a la evolución y reuso
- Esencial para la comunicación entre miembros de un equipo

Unificado

- A mediados de los noventa existían muchos métodos de Análisis y Diseño OO
  - Mismos conceptos con distinta notación
  - Mucha confusión
  - No existía un lenguaje líder de modelado
- En 1994 { G. Booch (Método Booch); J. Rumbaugh (OMT); I. Jacobson (Proceso Objectory) } deciden unificar sus métodos: ***Unified Modeling Language (UML)***
- Proceso de estandarización promovido por OMG en 1997

Lenguaje

- En la unificación se adopta un **Lenguaje Gráfico estándar**

- Como toda definición de Lenguaje, posee:

  - **Sintaxis:**

    determina las reglas de construcción de los diagramas de modelado.

    Provee base formal para los diagramas (metamodelo, OCL).

  - **Semántica:**

    otorga la descripción detallada del significado conceptual de cada elemento de modelado.

¿ Qué es entonces UML ?

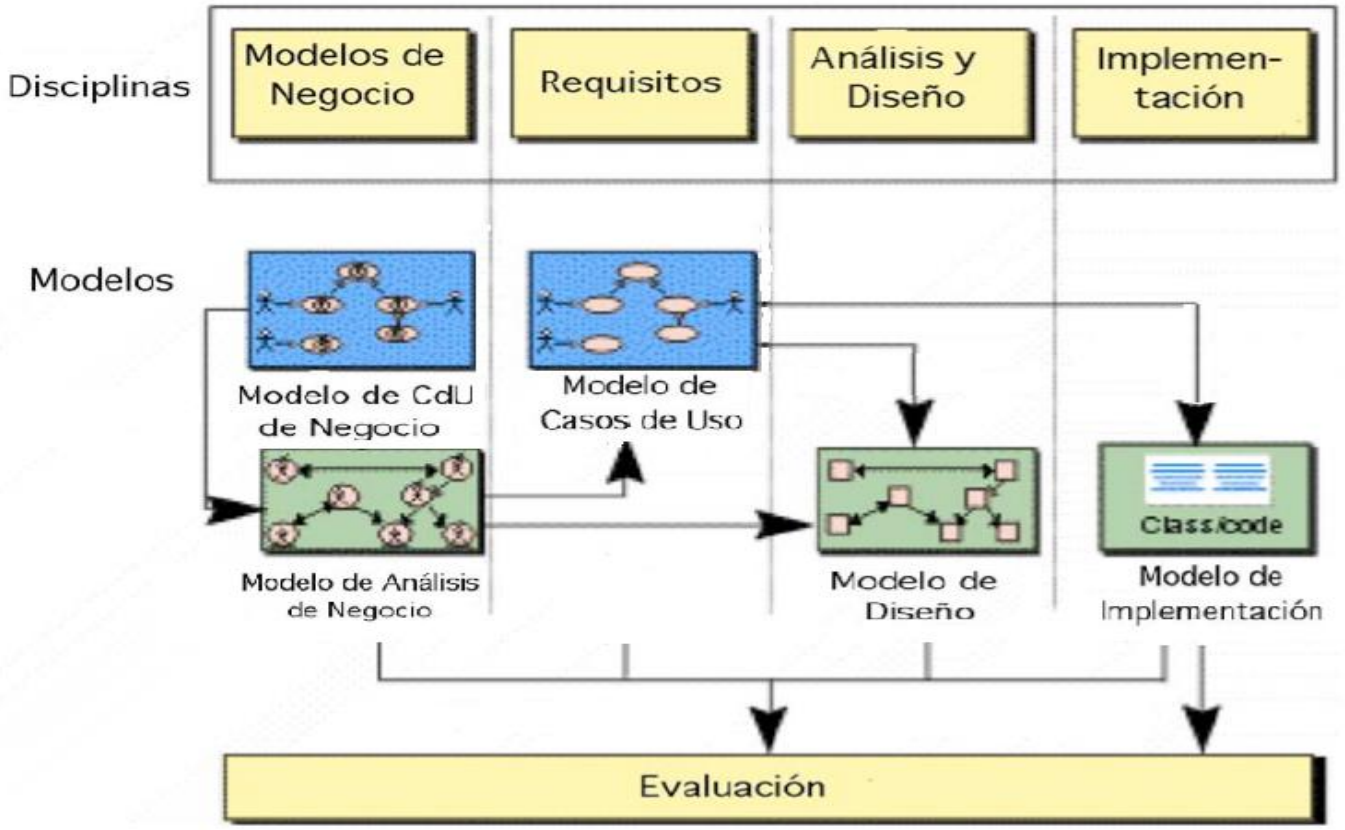
UML es un lenguaje estándar para:

- visualizar
- especificar
- construir
- documentar

los artefactos de un sistema.

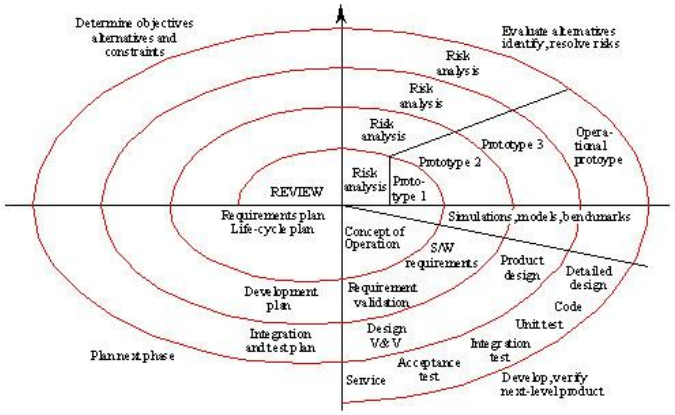
Lenguaje Unificado de Modelado

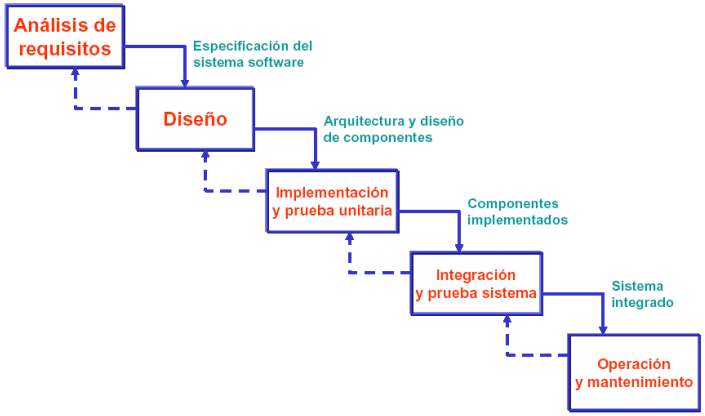
- Útil en las diferentes etapas del ciclo de vida del desarrollo de sistemas.

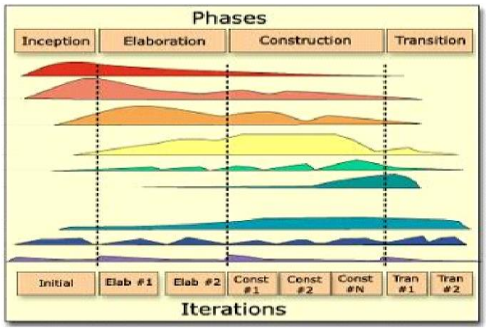


Lenguaje Unificado de Modelado

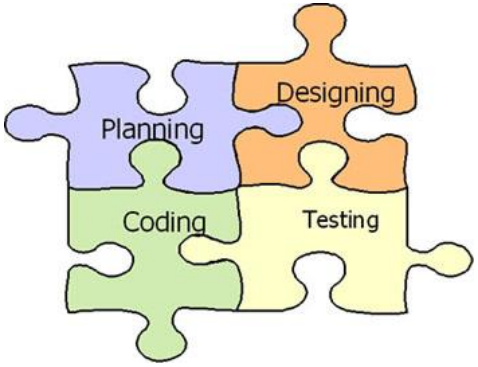
- Independiente del proceso de desarrollo de software.







MSF



Lenguaje Unificado de Modelado

- Independiente del lenguaje de implementación.

C++

python™

Microsoft .net

Ada

Java

¿Qué elementos definen un Método de Modelado?

- Un estilo → O.O.
- Un lenguaje → UML
- Un proceso → UP
- Herramientas → Together / UMLet / Rational Modeler/ EA

Utilidades de UML

UML permite:

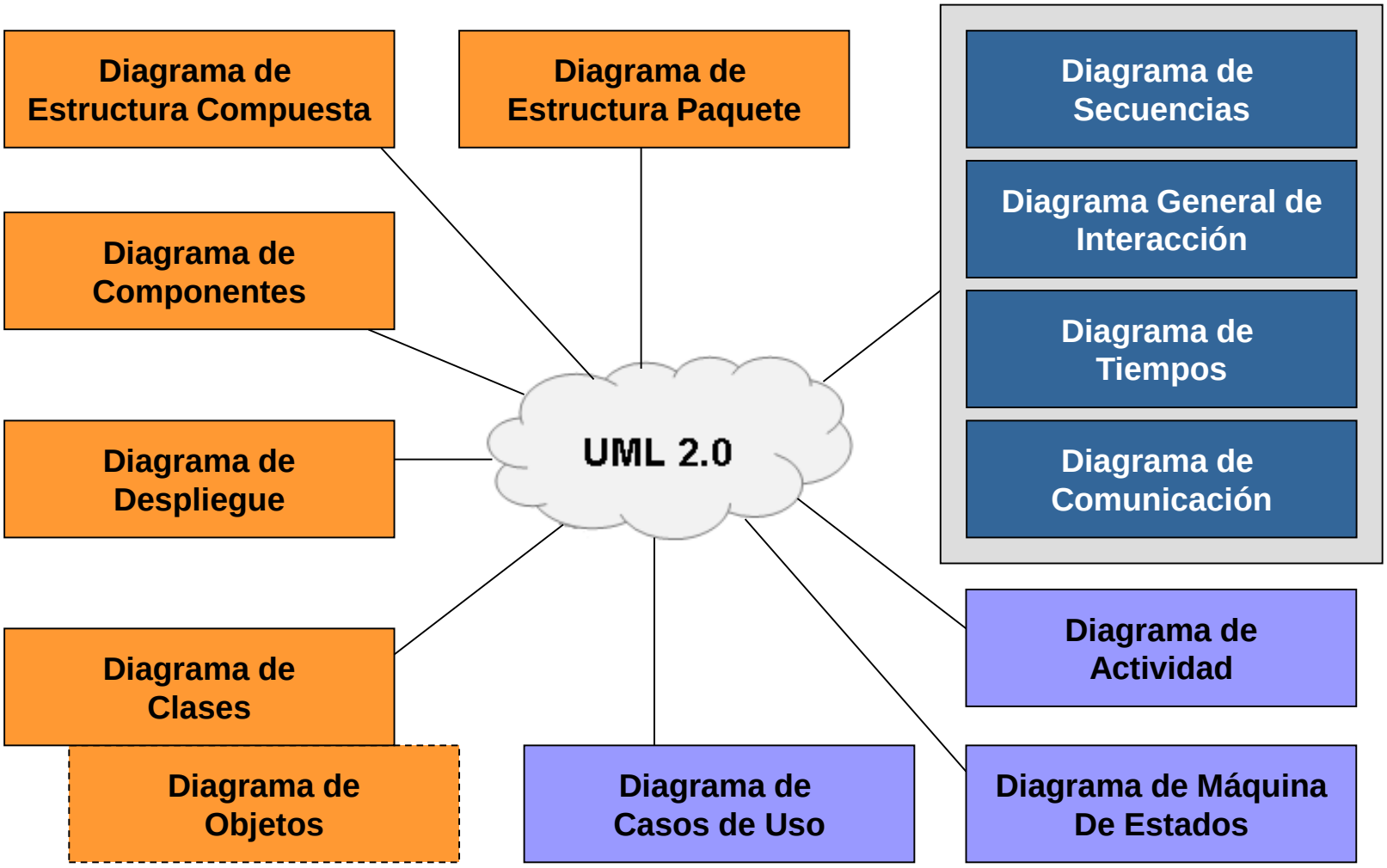
- Definir los límites del sistema y sus principales funciones, mediante **Casos de Uso y actores.**
- Ilustrar el funcionamiento de un caso de uso mediante **Diagramas de Interacción**
- Representar la estructura de un sistema mediante **Diagramas de Clases.**
- Modelar el comportamiento de los objetos mediante **Diagramas de Máquinas de Estados.**

Utilidades de UML

UML permite:

- Describir la arquitectura de la implementación física con **Diagramas de Componentes y Despliegue.**
- Extender la funcionalidad de elementos estándar mediante **estereotipos.**
- Proveer **base formal** para los diagramas (metamodelo, OCL).

Diagramas de UML 2.0



UML: bloques de construcción

- Elementos
  - Estructurales
    - Clase - Interfaz - Estructura Compuesta - Caso de Uso
    - Clase Activa - Componente - Nodo
  - De Instancia
    - Objetos
  - De Comportamiento
    - Interacción - Máquina de Estados - Actividad
  - De Agrupación
    - Paquetes - Frame
  - De Anotación
    - Notas
- Relaciones
  - Dependencia
  - Asociación
  - Generalización
- Diagramas
  - Diagrama de Clases
  - Diagrama de Objetos
  - Diagrama de Estructura Compuesta
  - Diagrama de Paquetes
  - Diagrama de Componente
  - Diagrama de Despliegue
  - Diagrama de Casos de Uso
  - Diagrama de Máquina de Estados
  - Diagrama de Actividades
  - Diagrama de Secuencia
  - Diagrama de Comunicación

Clasificación de los Diagramas UML

Diagramas de Casos de Uso

Diagramas Estáticos

- Diagrama de Clases
- Diagrama de Objetos
- Diagrama de Estructuras Compuesta
- Diagrama de Paquetes

Diagramas de Comportamiento

- Diagrama de Máquina de Estados
- Diagrama de Actividades
- Diagrama de Interacción:
  - Diagramas de Secuencia
  - Diagramas de Comunicación
  - Diagramas de Tiempo
  - Diagramas de Vista Global

Diagramas de Implementación

- Diagrama de Componentes
- Diagrama de Despliegue

Diagramas de Interacción

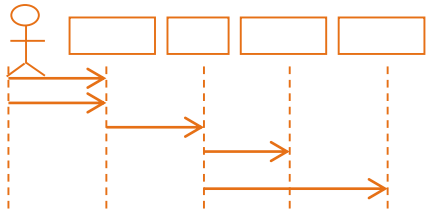
Diagramas de Interacción

Definición:

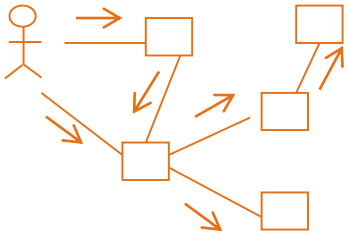
Un diagrama de interacción describe una interacción, que consta de un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar, para realizar un comportamiento.

Diagramas de Interacción: tipos

- **Diagramas de Secuencia:**



- **Diagramas de Colaboración:**



Diagramas de Interacción: contenido

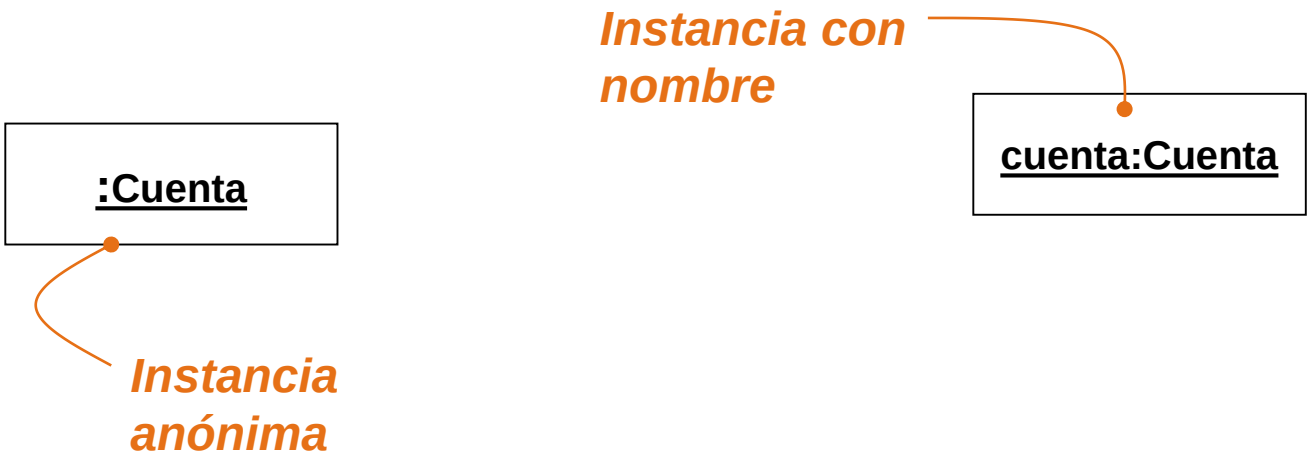
- Objetos
- Enlaces
- Mensajes

  - **Pueden contener:**
    - notas y restricciones.

Diagramas de Interacción: objetos

- Los objetos que participan en una interacción son o bien elementos concretos o bien elementos prototípicos.

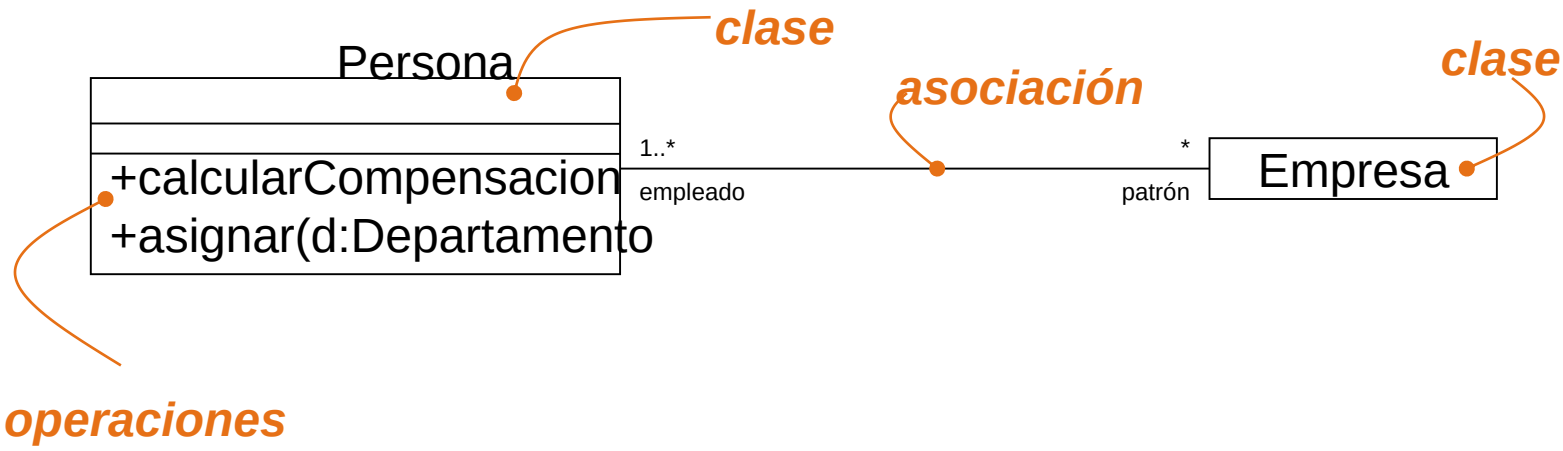
- **Ejemplos**

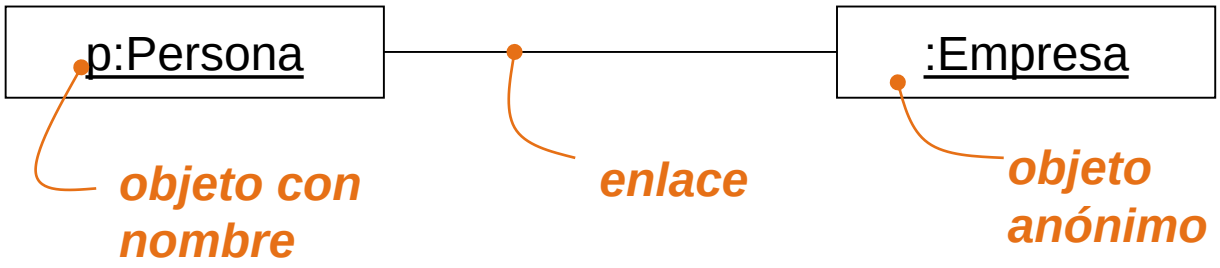


Diagramas de Interacción: enlaces

- Un enlace es una conexión semántica entre objetos.

- **Ejemplos**





Diagramas de Interacción

Diagramas de Secuencia

Diagrama de Secuencia

- **Definición:**

  Un diagrama de secuencia destaca la ordenación temporal de los mensajes.

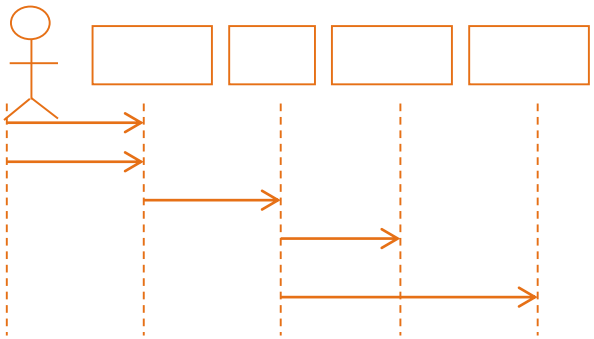
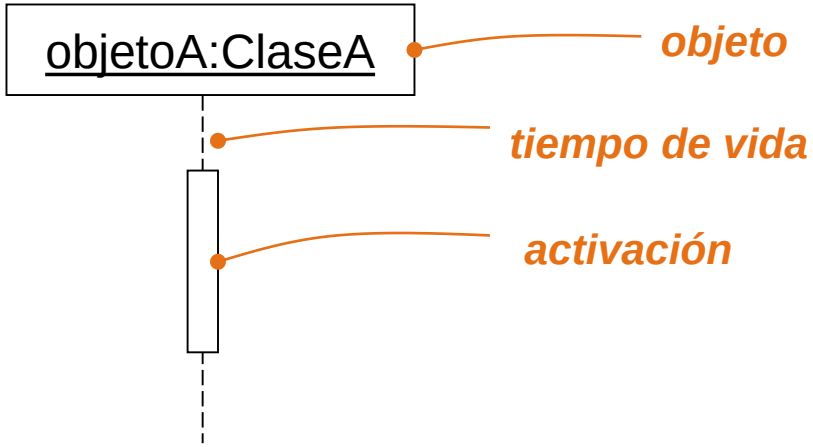


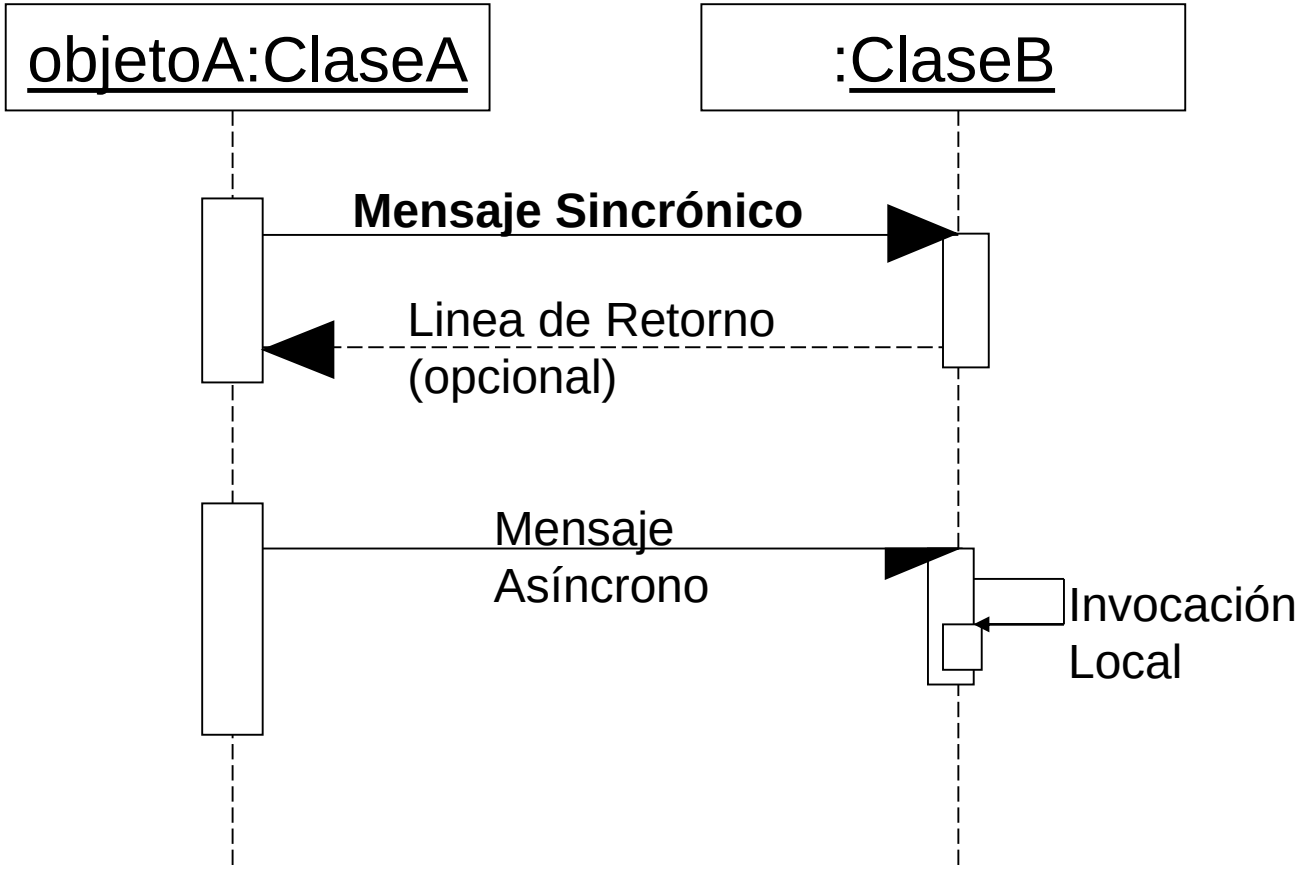
Diagrama de Secuencia

- Cada objeto cuenta con una *línea de vida*, que muestra el tiempo de vida del mismo.
- La *activación* de un objeto representa la ejecución de una operación que realiza el mismo, a través del envío de un *mensaje*.

- **Notación:**



Mensajes


objetoA:ClaseA
:ClaseB
Mensaje Sincrónico
Linea de Retorno
(opcional)
Mensaje
Asíncrono
Invocación
Local

Mensaje sincrónico

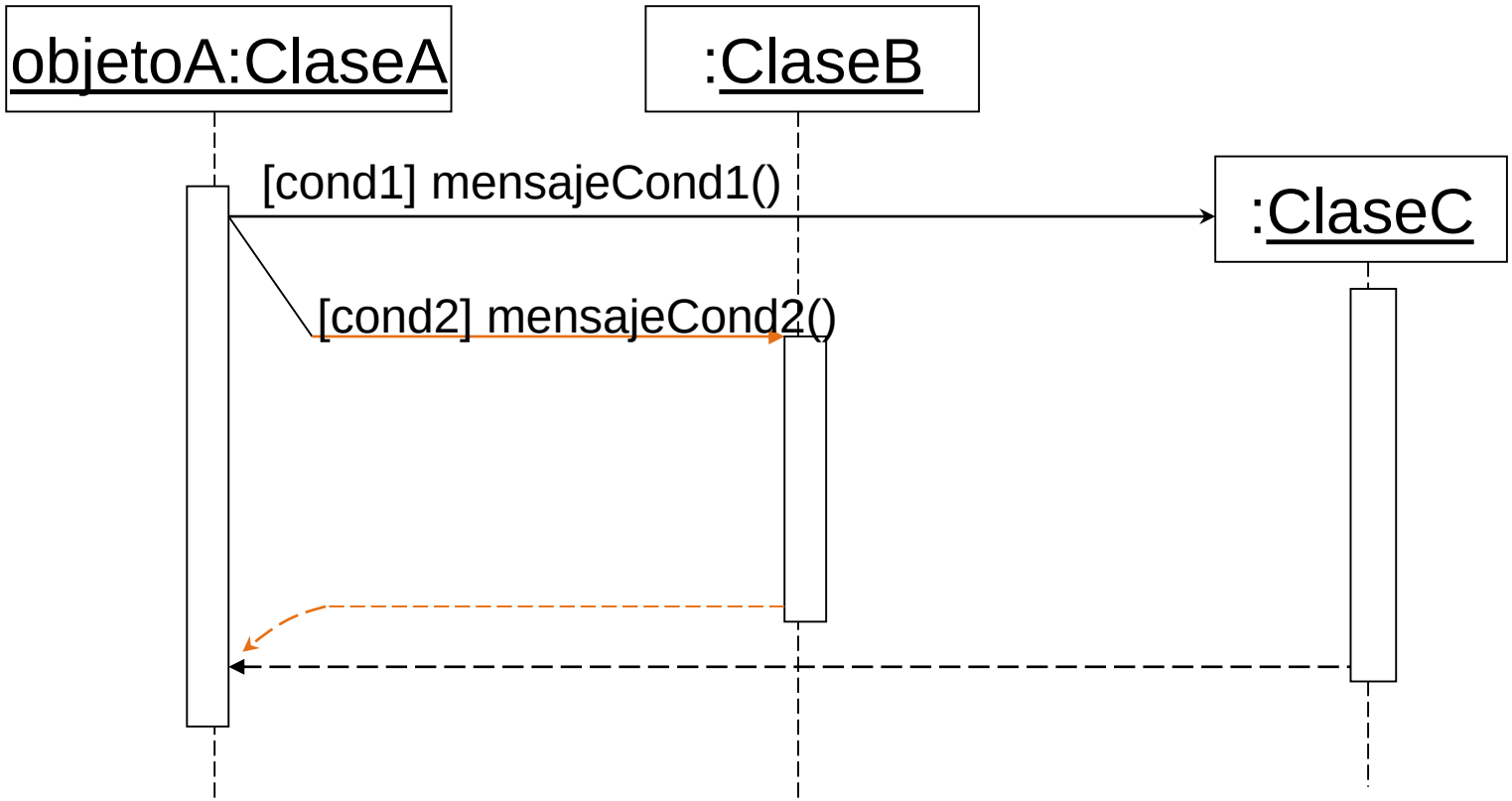
- **Sintaxis:**

  [Número de secuencia] [condición] * [expresión iteración] [valor de retorno :=] nombre del mensaje (parámetros)

- Puede indicarse el fin de su ejecución con una línea punteada del objeto receptor al emisor: Línea Retorno/ Resultado Esta línea es OPCIONAL (en caso que se especifique correctamente la Activación del objeto Receptor).

- El Resultado o Valor de Retorno también puede especificarse a izquierda del nombre del mensaje, en una asignación (ver Sintaxis), en caso que No se haya especificado la línea Retorno.

Bifurcaciones



Iteración o bucle

- **Sintaxis:**

  * [expresión-iteración] mensaje

- **Ejemplo:**

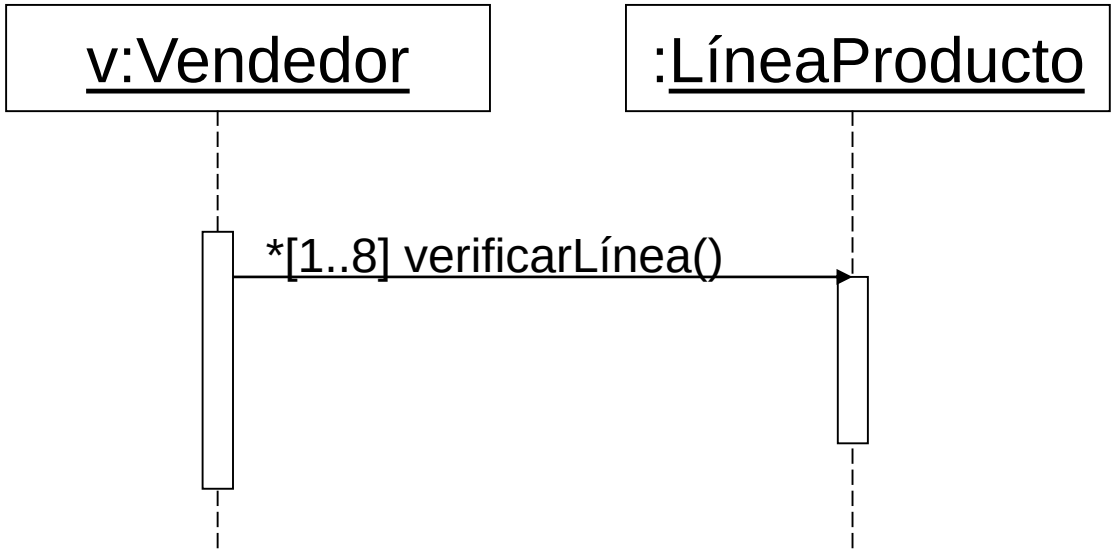


Diagrama de Secuencia: construcción

- **Primer Paso:** Se colocan los objetos que participan en la interacción en la parte superior del diagrama.

- **Ejemplo**

:Cliente

a:AyudaPlanificación

Diagrama de Secuencia: construcción

- **Segundo Paso:** se colocan los mensajes que estos objetos envían y reciben, en orden de sucesión en el tiempo, desde arriba hasta abajo.
- **Ejemplo**

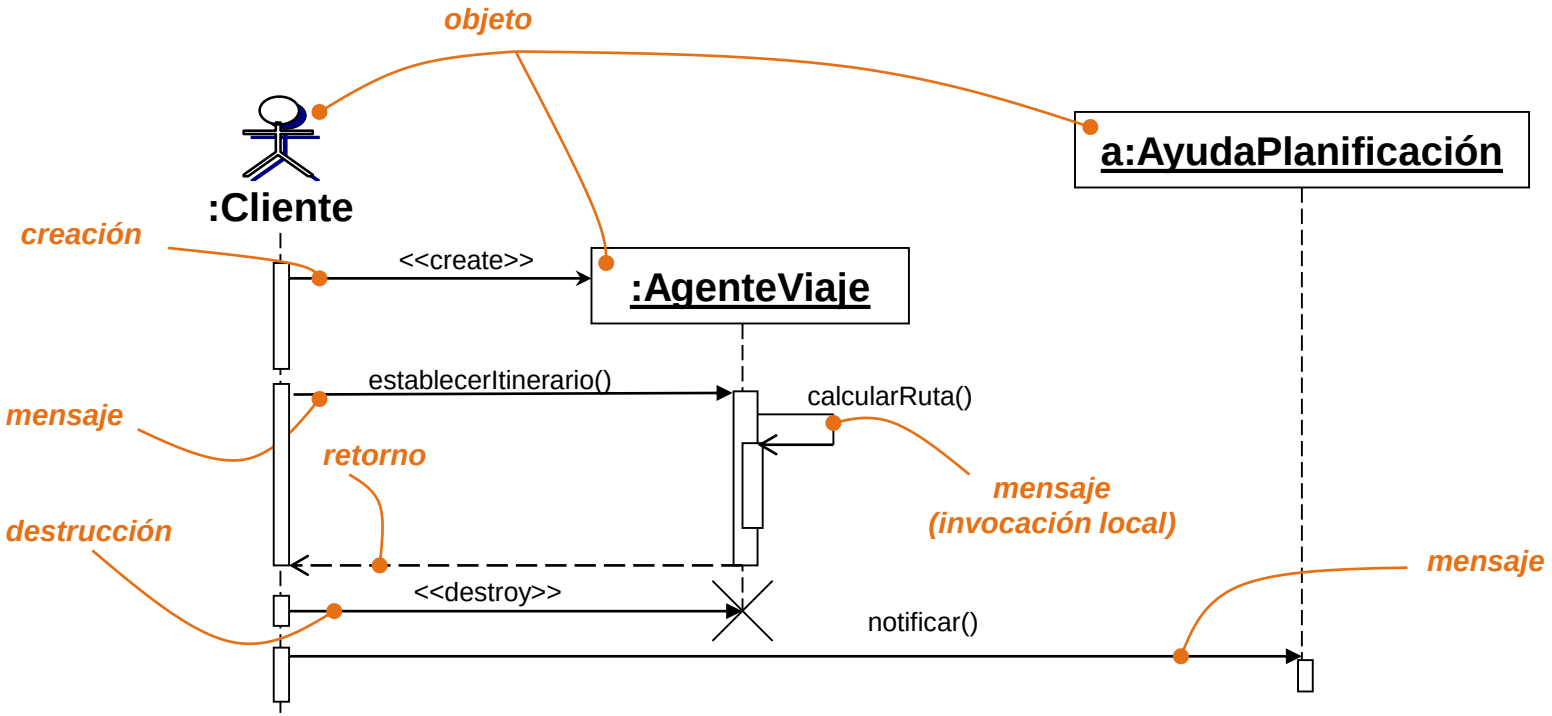


Diagrama de Secuencia: notación

- Se encierra en un rectángulo y se le agrega una etiqueta con **sd** seguido del nombre.

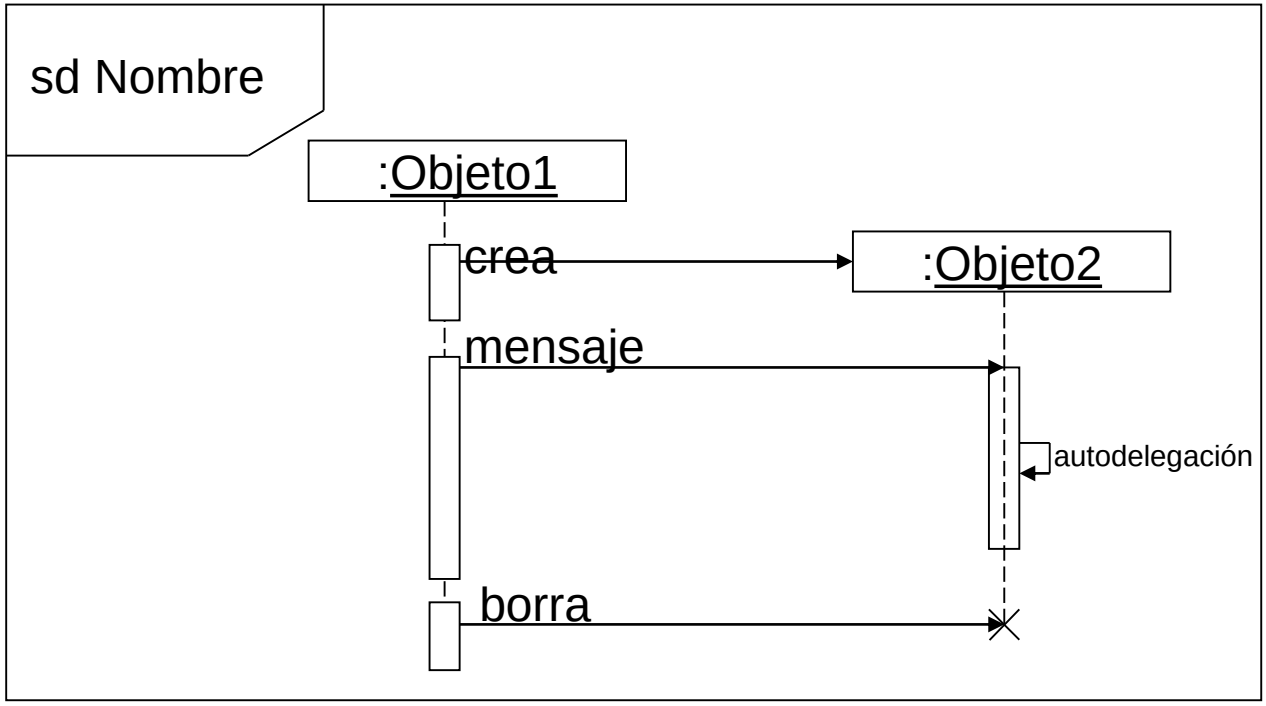


Diagrama de Secuencia: fragmentos

- **Definición:**

  mecanismo a través del cual se puede realizar la especificación de bloques repetitivos, opcionales, alternativos, entre otros.

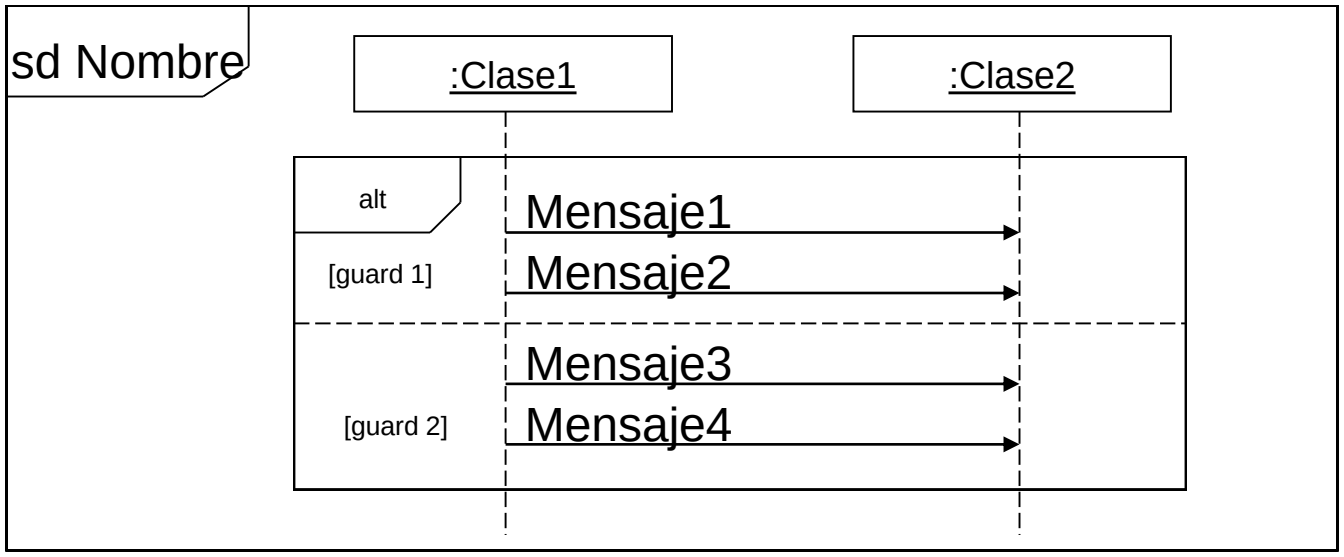
- **Operadores más utilizados:**

  - ref opt alt loop

Fragmento: alternativa

- **Notación**: se encierra en un rectángulo (*frame*), se le agrega una etiqueta con el operador **alt** y se colocan las guardas.

- **Ejemplo**



Fragmento: bucle

- **Notación**: se encierra en un rectángulo (*frame*), se le agrega una etiqueta con el operador **loop** y la cantidad de iteraciones (opcional).

- **Ejemplo**

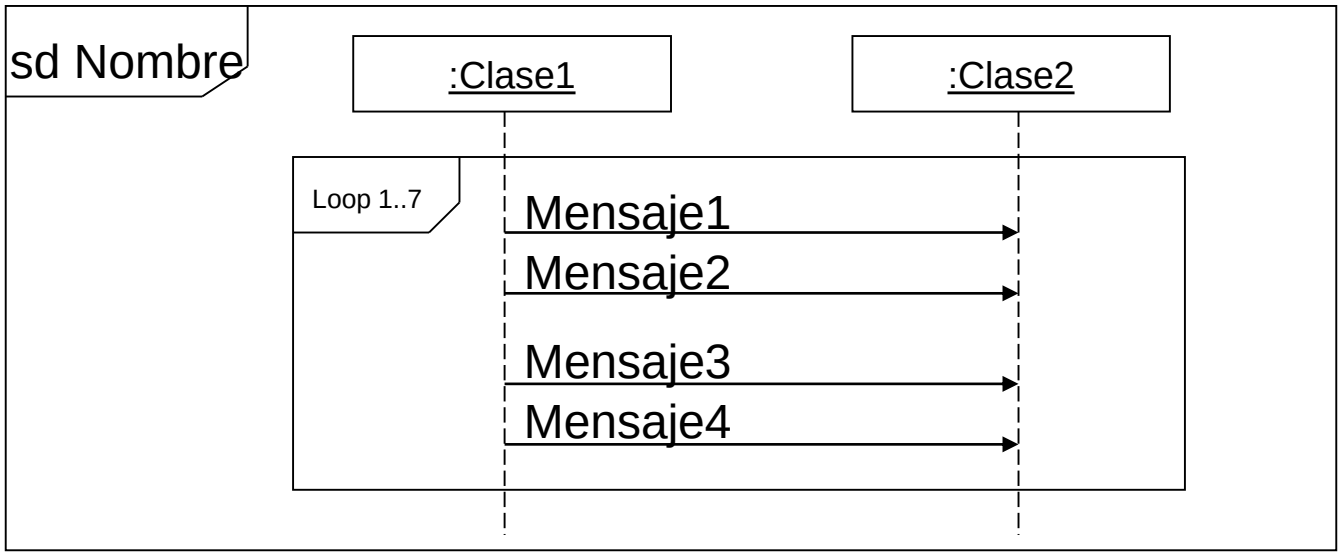
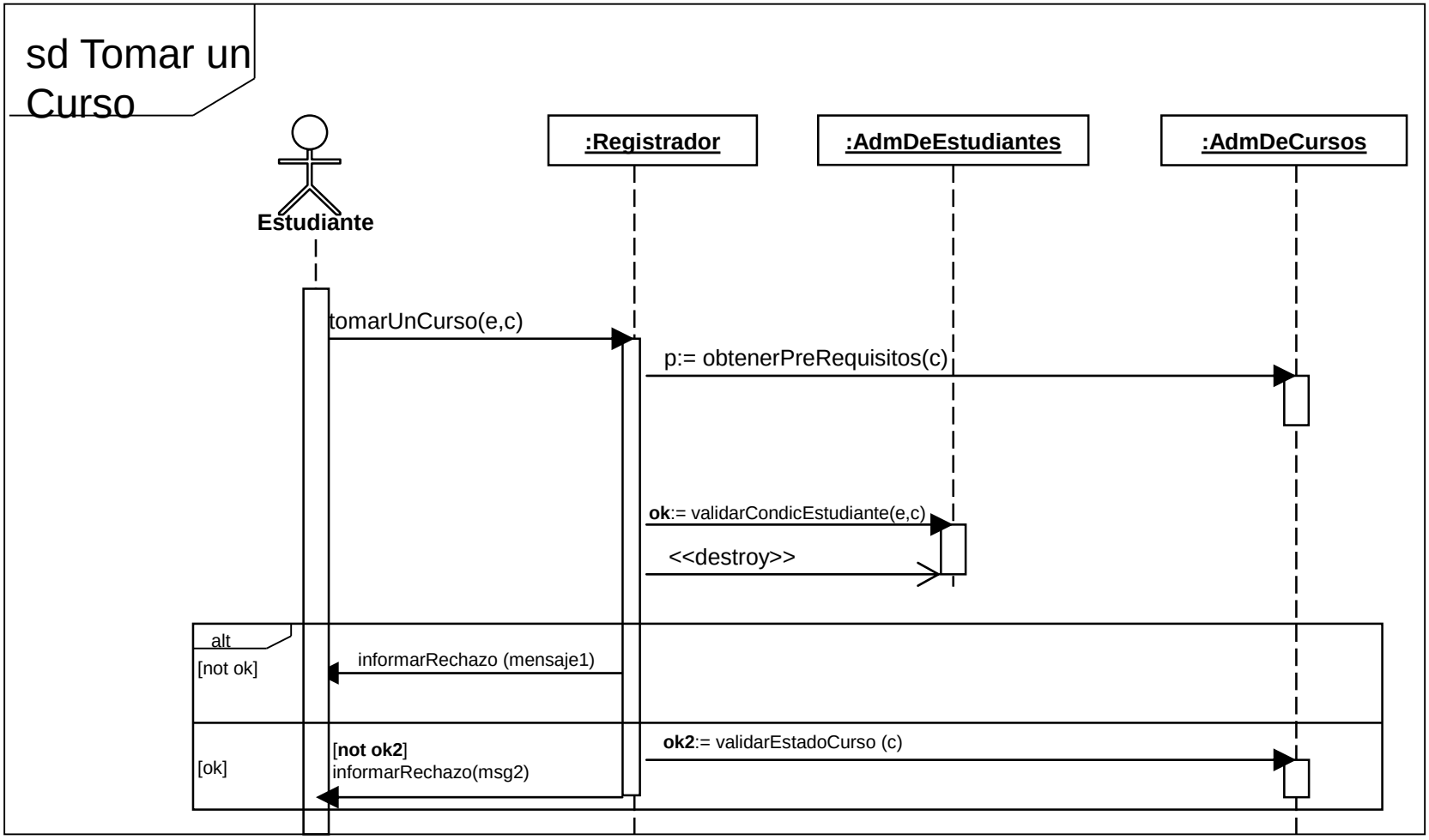
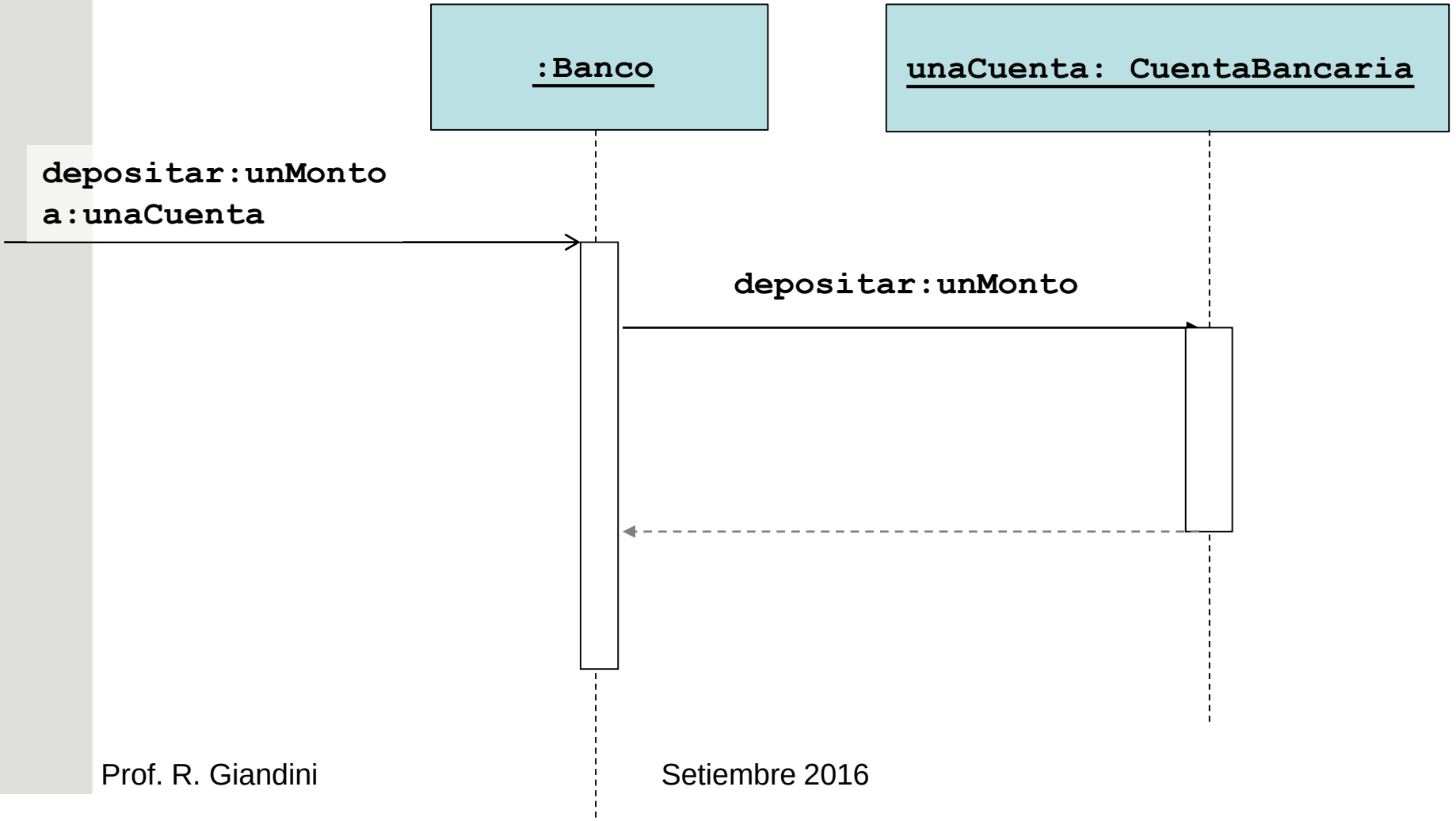


Diagrama de Secuencia: ejemplo



Ejemplo: Depósito bancario



Ejemplo: Transferencia bancaria

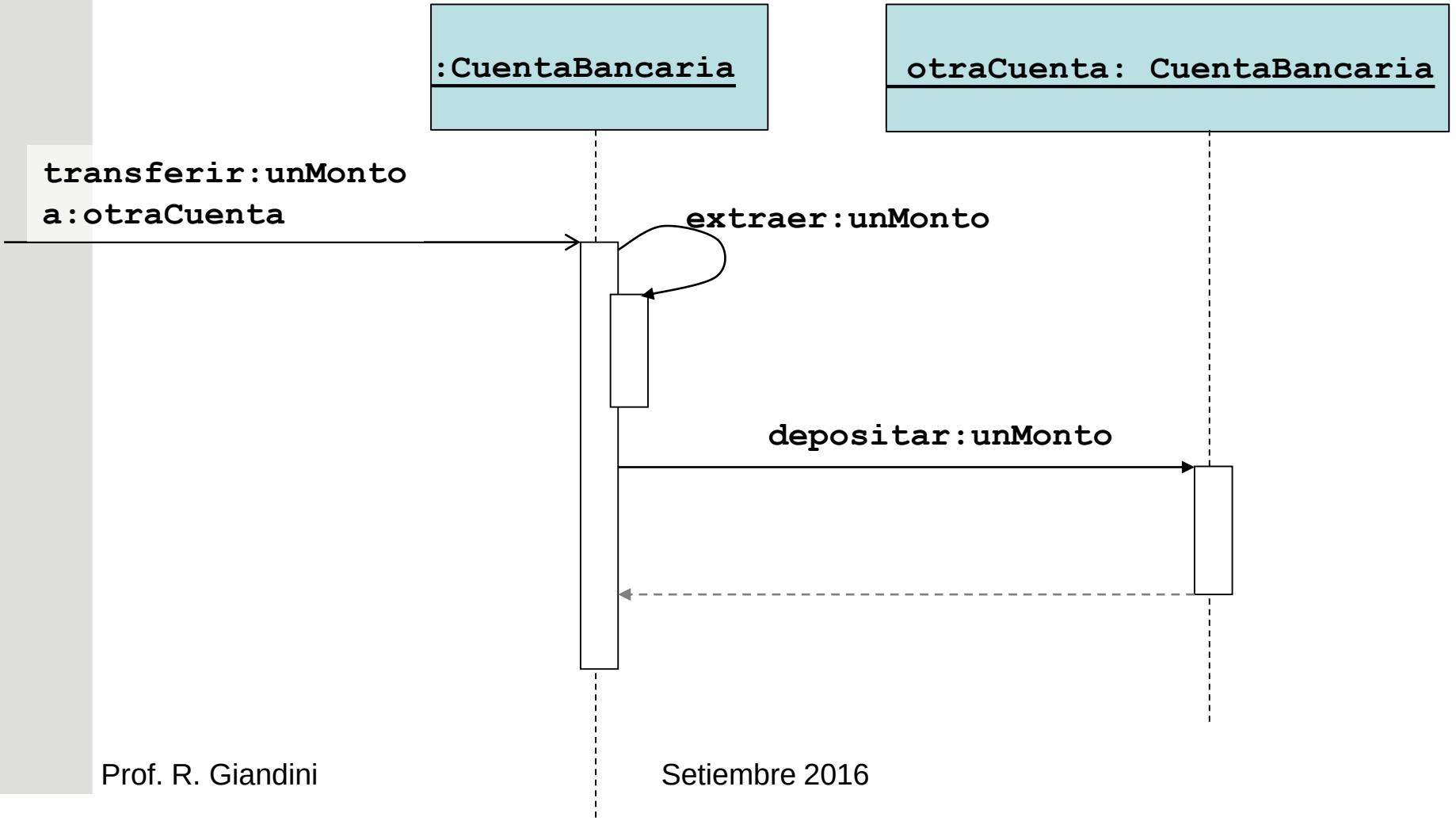


Diagrama de Clases

Diagrama de Clases

Definición:

Un diagrama de clases muestra las clases del sistema y sus relaciones. Describe la vista estática de un sistema.

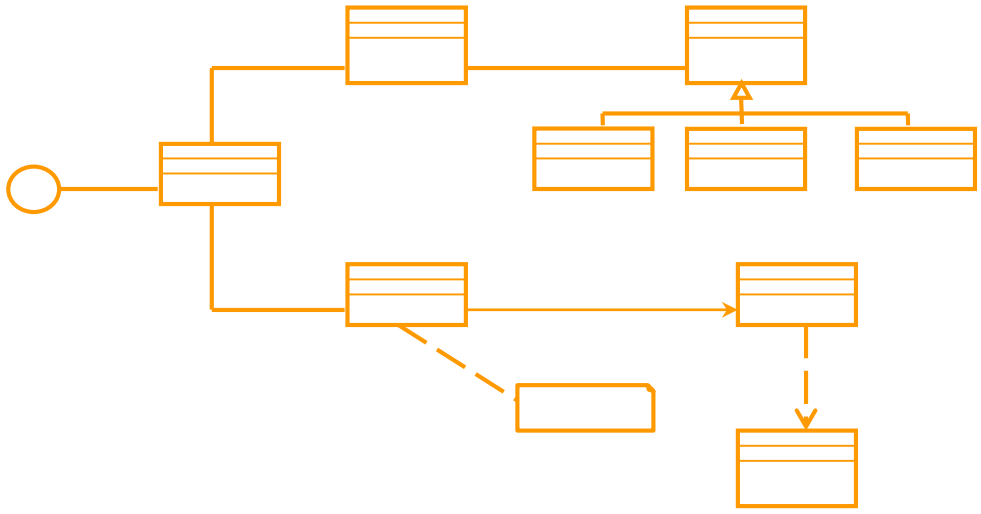
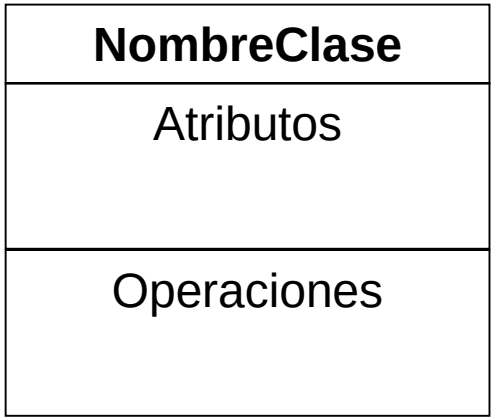


Diagrama de Clases: clase

- **¿Qué es una clase?**

  Una clase es una descripción de un conjunto de *objetos* que comparten los mismos atributos, operaciones, relaciones y semántica.

- **Estructura:**

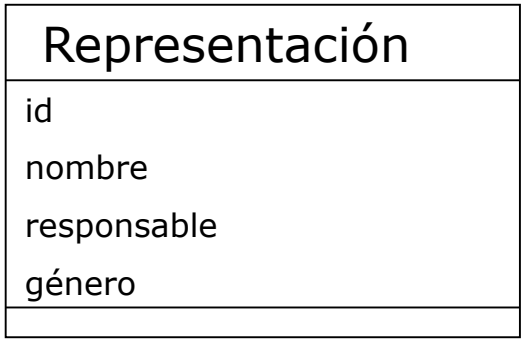


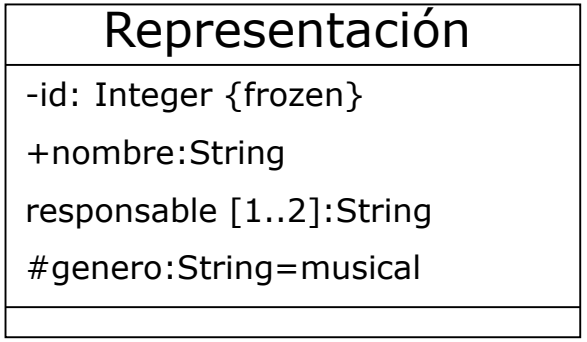
Clase: atributo

Sintaxis

[visibilidad] nombre [multiplicidad][:tipo][=valor inicial][{lista_propiedades}]

Ejemplos





Clase: operación

- **Sintaxis**

[visibilidad] nombre[(lista de parámetros)][:tipo de retorno][{propiedades}]

  - Declaración de un parámetro:

    [dirección] nombre :tipo [multiplicidad][=valor]

    - Donde dirección puede ser:
      - **in**: la operación puede modificar el parámetro y el que llama no necesita volver a verlo.
      - **out**: la operación coloca o cambia el parámetro y lo devuelve al que llama.
      - **inout**: la operación utiliza el parámetro y puede cambiarlo para devolverlo.

Clase: operación

Propiedades

- isQuery
- ....

Ejemplos

Representación
nuevaRep() nombreRep() estaVigente()

Representación
+nuevaRep(**in** nombre:String) #nombreRep() estaVigente(): Boolean {isQuery}

Diagrama de clases: relaciones

¿Qué son las relaciones entre clases?

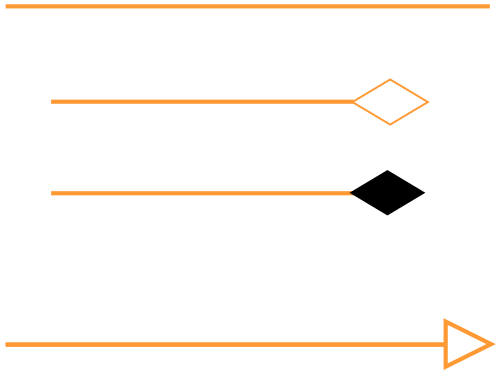
Una relación es una conexión semántica entre objetos. Proveen un camino de comunicación entre ellos.

¿Qué tipos de relaciones usaremos?

- Asociación
- Generalización

Diagrama de clases: relaciones

- **Notación:**
  - Asociación
    - Agregación
    - Composición
  - Generalización

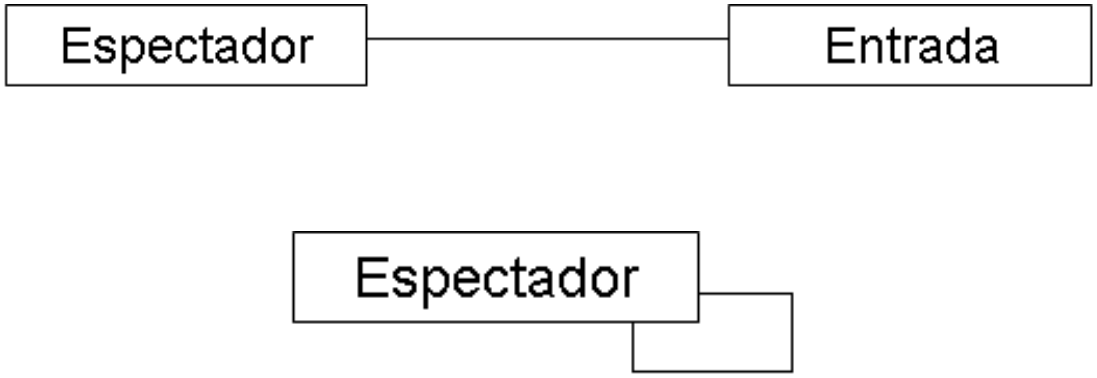


Relaciones: Asociación

¿Qué es una Asociación?

Es una relación estructural que especifica que los objetos de un elemento están conectados con los objetos de otro.

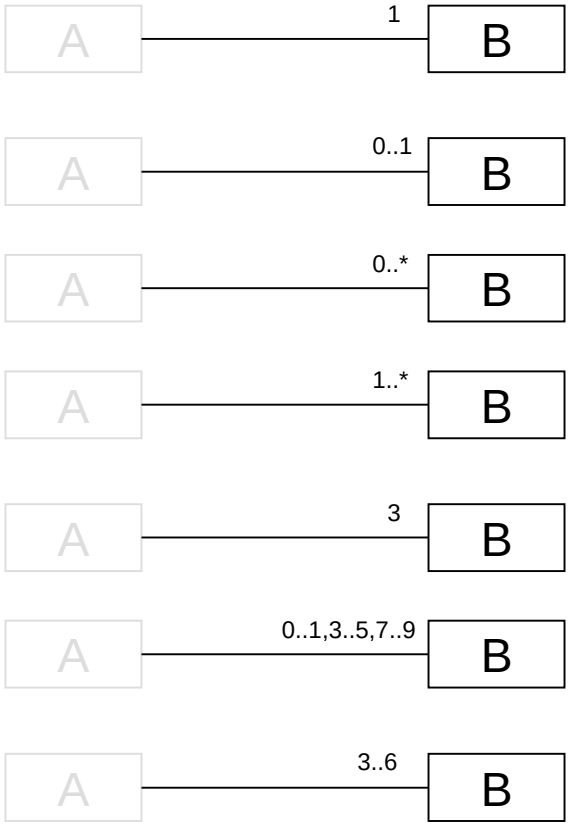
Ejemplo



Asociación: propiedades (cont.)

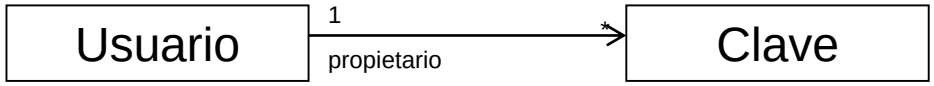
Multiplicidad: indica “cuántos” objetos pueden conectarse a través de una instancia de una asociación.

- **Se puede indicar una multiplicidad de:**
  - **Exactamente Uno:**
  - **Cero o Uno:**
  - **Muchos:**
  - **Uno o Más:**
  - **Número Exacto:**
  - **Lista:**
  - **Rango:**

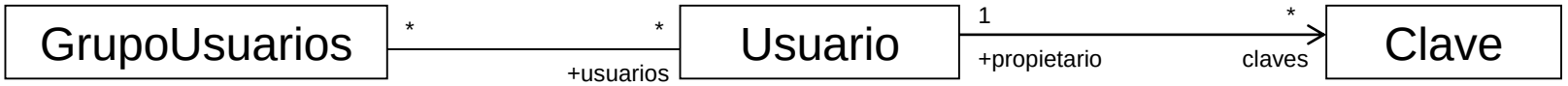


Asociación: propiedades

- **Rol:** son las caras que presentan las clases a las demás.
- **Navegabilidad**: sirve para limitar la navegación a una sola dirección.

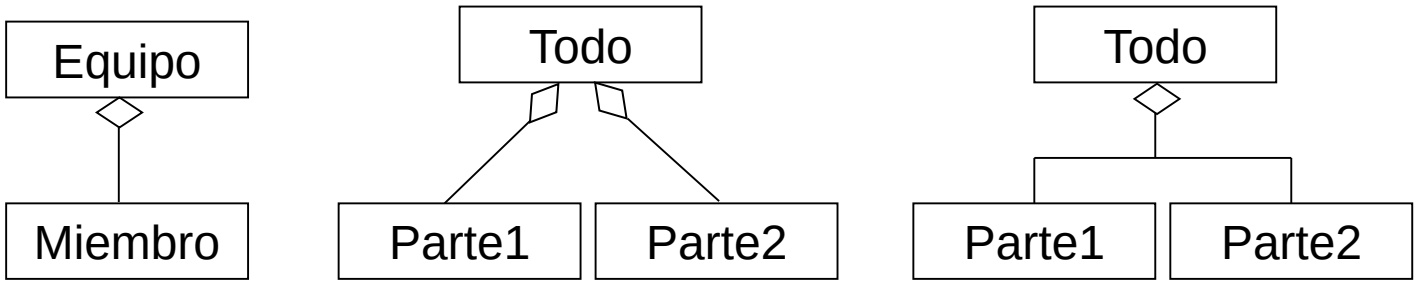


- **Visibilidad:** sirve para limitar la visibilidad a través de esta asociación relativa a los objetos externos a ella.
  - Pública (+)
  - Protegida (#)
  - Privada (-)

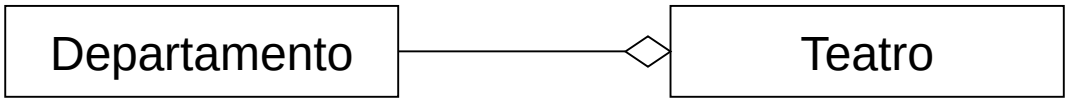


Asociación: propiedades (cont.)

Agregación: es una asociación especial, una relación del tipo "todo/parte" dentro de la cual una o más clases son partes de un todo.



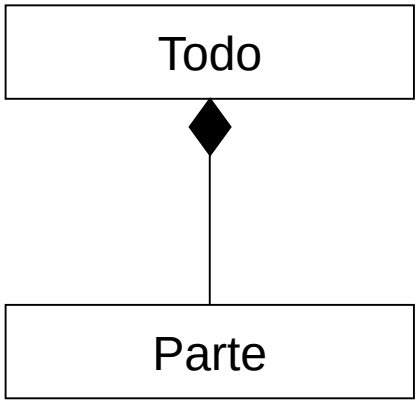
Ejemplo:



Asociación: propiedades (cont.)

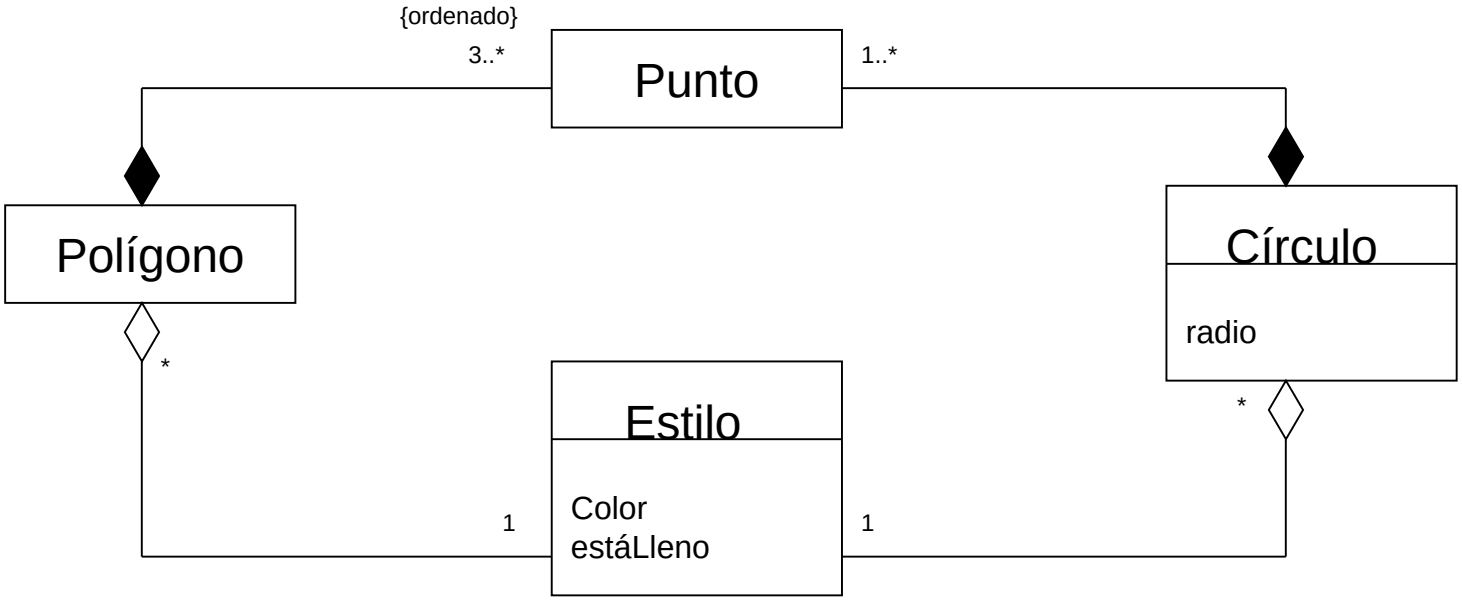
Composición: es una forma de agregación, con fuerte sentido de posesión y tiempo de vida coincidentes de las partes con el conjunto.

- Una parte puede pertenecer solamente a una composición (un *todo*).
- Cuando el todo desaparece, también lo hacen sus partes.



Asociación: propiedades (cont.)

Ejemplos de Agregación y Composición

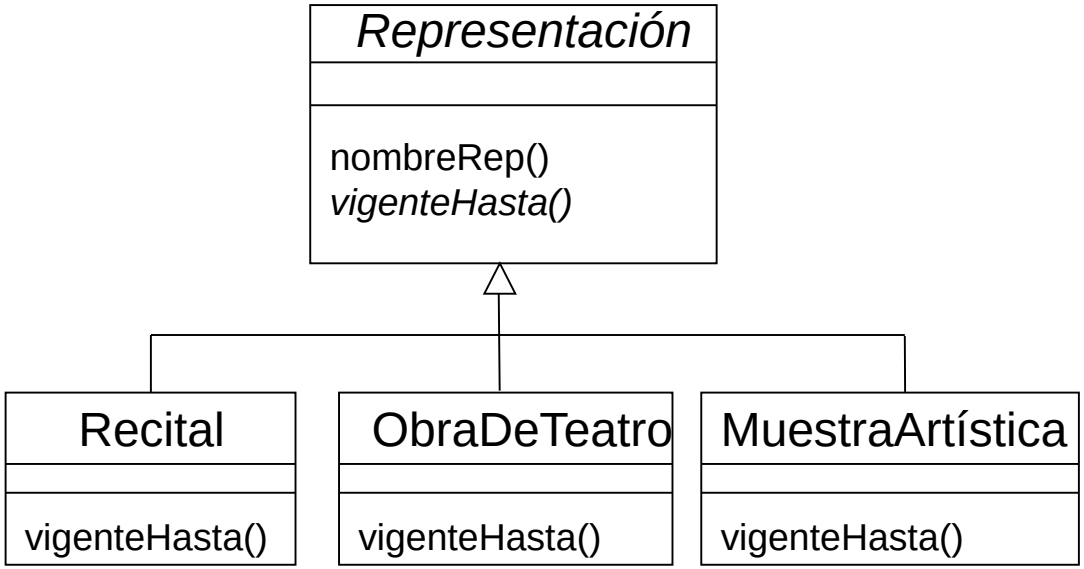


Relaciones: Generalización

¿Qué es una Generalización?

Es una relación entre un elemento general (llamado superclase o padre) y un caso más específico de ese elemento (llamado subclase o hijo).

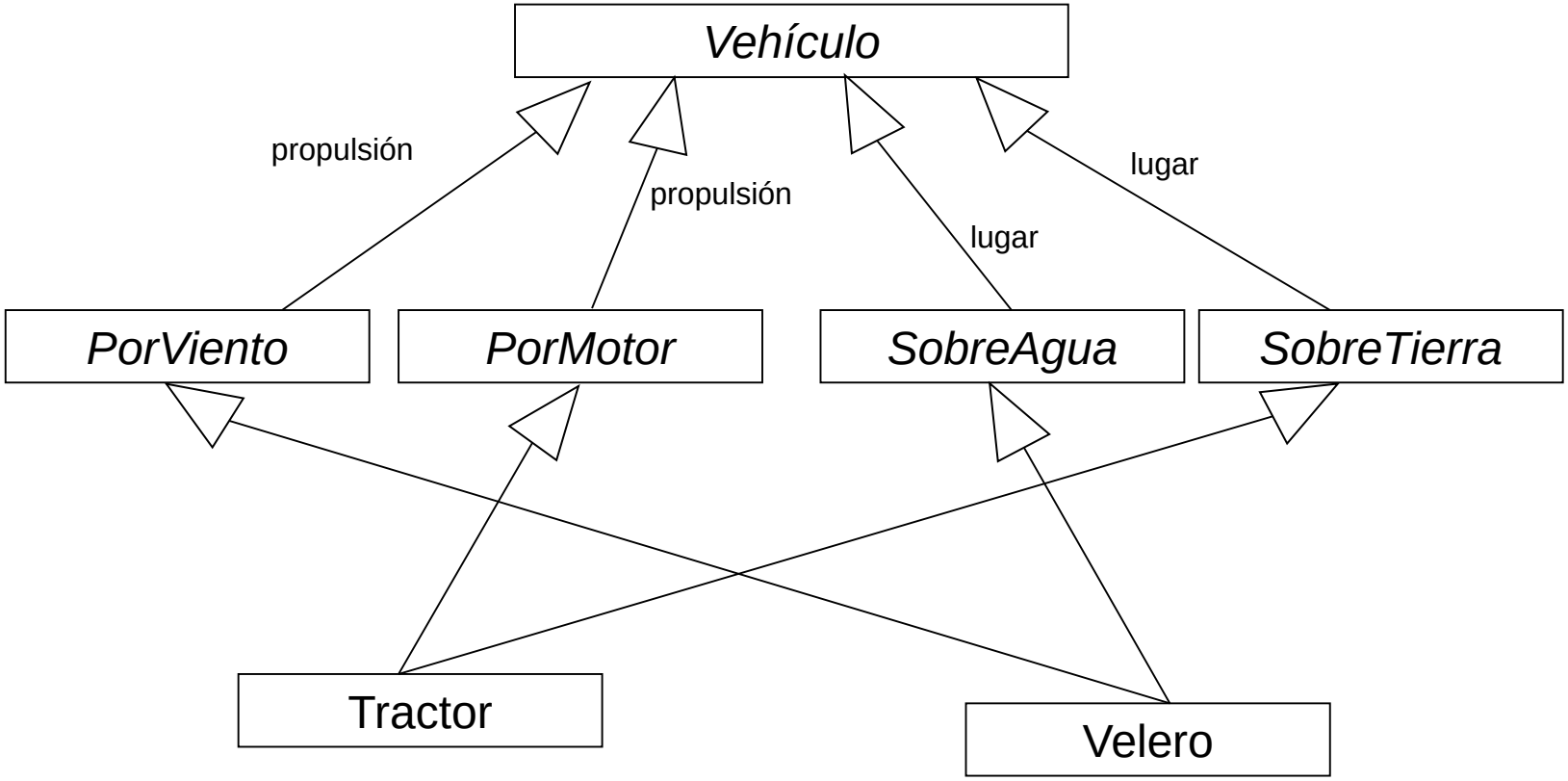
Ejemplo



- En UML puede representarse herencia múltiple

Generalización

- **Ejemplo con discriminador**



Elementos de anotación

- **¿Qué es una nota?**

  Es un símbolo para mostrar restricciones y comentarios junto a un elemento o una colección de elementos.

- **Ejemplo**

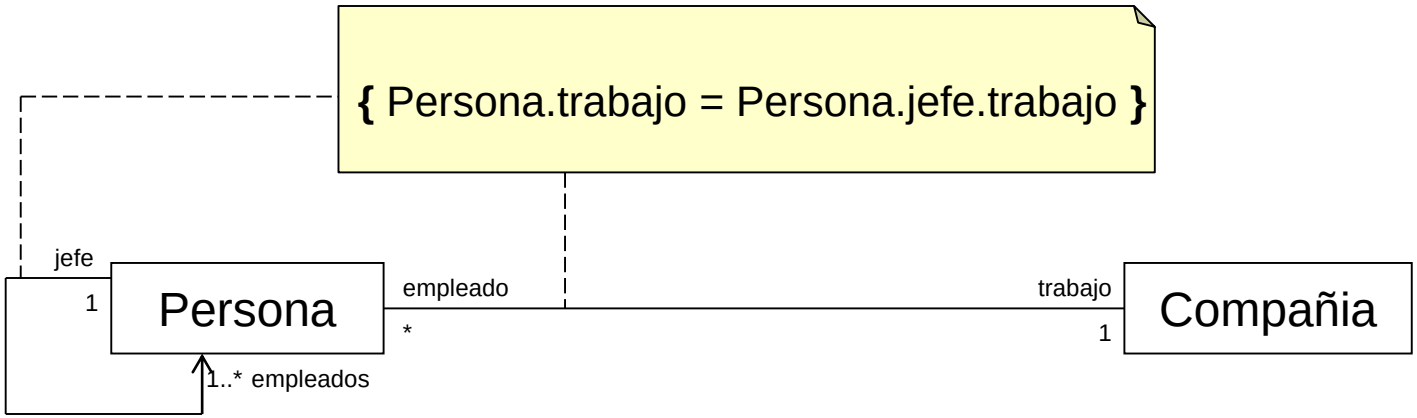


Diagrama de clases - Ejemplo