

# Introducción a UML

Arturo Zambrano  
Master en Ingeniería de Software  
Facultad de Informática - UNLP

# Contenidos

- 
- Diagrama de Clases
- Diagrama de Secuencia
- Diagrama de Máquinas de Estado

- UML: Unified Modeling Language
- UML es un lenguaje para visualizar, especificar y documentar los artefactos de un sistema de software
- UML puede usarse en diferentes etapas del ciclo de vida del desarrollo y en diferentes tecnologías de implementación
- UML es independiente del proceso de desarrollo de software
- UML fue adoptado por OMG ..

- UML permite:
- Definir los límites del sistema y sus principales funciones, mediante casos de uso y actores
- Representar la estructura de un sistema mediante diagramas de clases y paquetes que definen módulos.
- Ilustrar el funcionamiento de un caso de uso mediante diagramas de interacción.
- Modelar el comportamiento de los objetos mediante diagramas de transición de estados.
- Definir restricciones del dominio (OCL).
- Extender UML utilizando el metamodelo

# Diagrama de Casos de Uso

Los Casos de Uso son útiles para representar requisitos funcionales. Fueron propuestos inicialmente por Ivar Jacobson e incorporados a UML.

## Objetivos:

- Dar una descripción de lo que deberá hacer el sistema
- Dar un base para realizar las pruebas del sistema
- Definir una base para rastrear errores, simplificando los cambios y extensiones del sistema.

# Diagramas de Casos de Uso

## Contenido:

- Casos de Uso



Nombre del caso de uso

- Actores



Nombre\_Actor

<< actor >>

Nombre\_Actor

- Relaciones



Dependencia



Asociación



Generalización

- Un Caso de Uso es, en esencia, una interacción típica entre un usuario y un sistema.
- El Caso de Uso capta una función visible para el usuario.
- El Caso de Uso logra un objetivo concreto para el usuario.



## Notación:

- Asociación



- Generalización



- Dependencia



- Inclusión



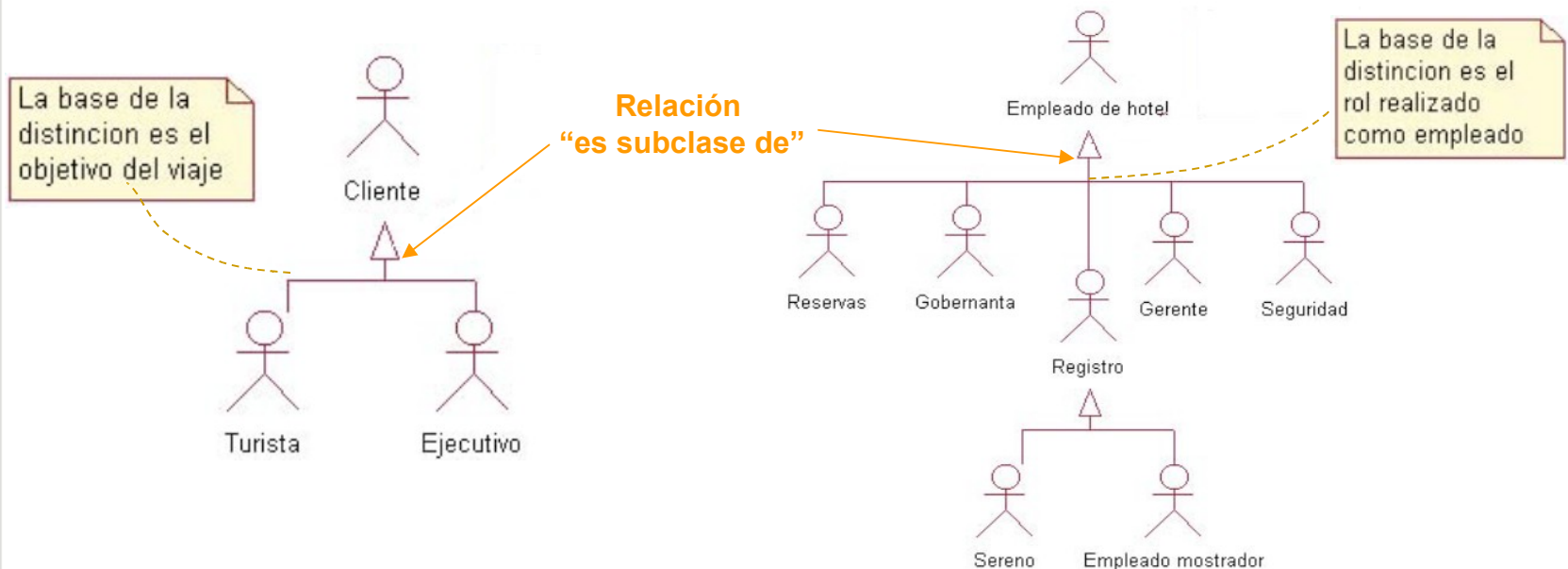
- Extensión



# Relaciones entre Actores

Se pueden establecer relaciones de generalización entre actores:

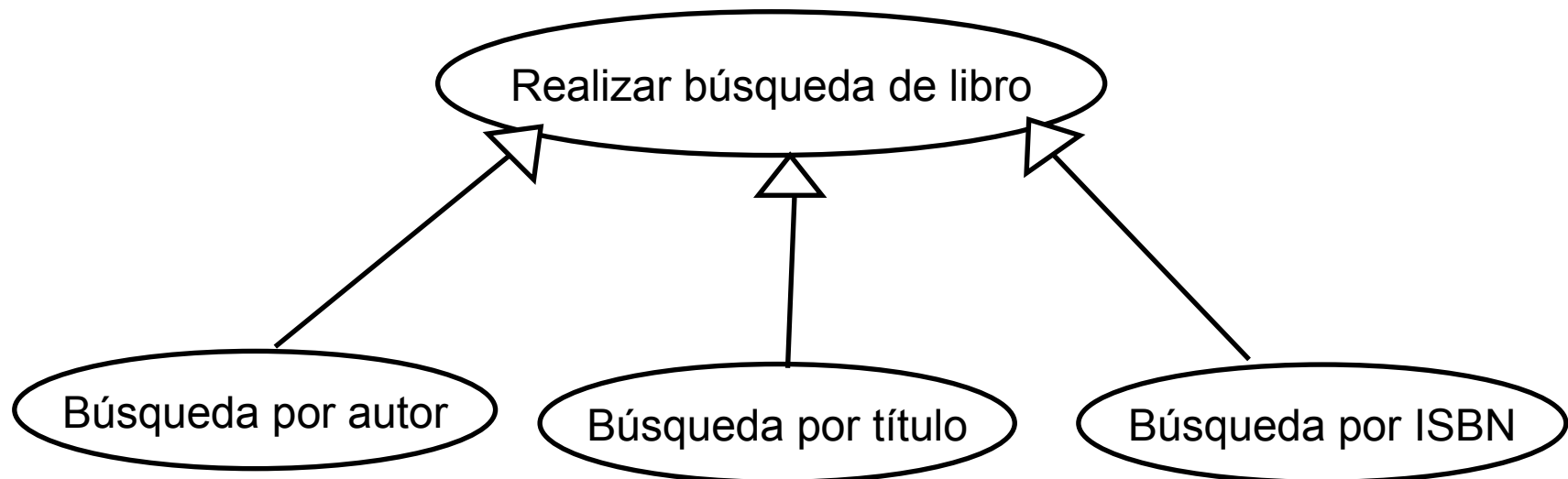
- El actor **general** describirá el comportamiento de un rol más general.
- Los actores **especializados** heredan el comportamiento del actor general y lo refinan de alguna forma.



Una instancia de un actor **especializado** siempre se puede utilizar en aquellos casos en los que se espera una instancia del actor **general**.

## Generalización:

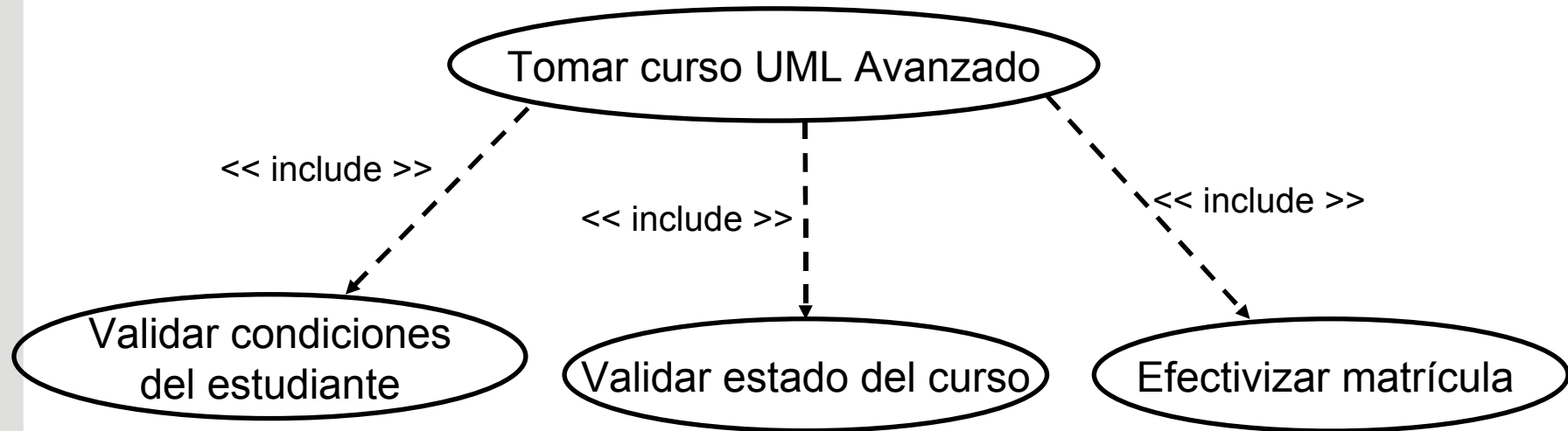
Se utiliza cuando un Caso de Uso se puede especializar en uno o más Casos de Uso hijos.



## Inclusión:

- El Caso de Uso incorpora el comportamiento de otros Casos de Uso como parte de su propio comportamiento en un determinado momento del curso de su acción.
- Se entiende que algunos de los pasos en una situación dentro de un Caso de Uso son los mismos que los de otro Caso de Uso, y en lugar de listar los mismos pasos, tan sólo indicamos el Caso de Uso de donde provienen.

# Ejemplo

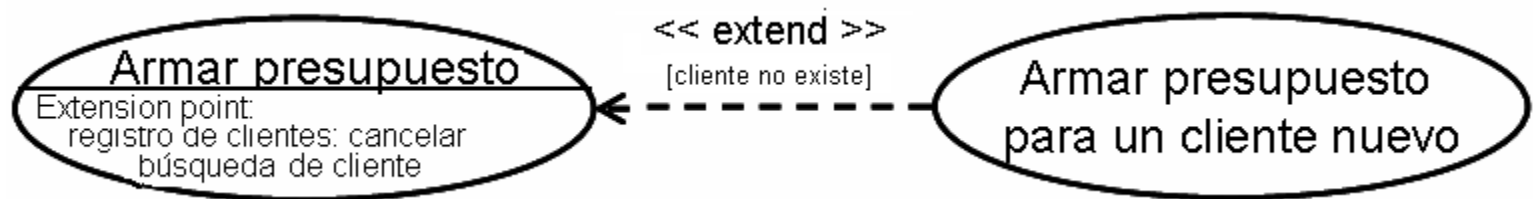


No debe pensarse como una simple descomposición.

Los casos incluídos deben ser significativos per se.

## Extensión:

- El Caso de Uso se define como una extensión incremental de un caso de uso base en uno o más puntos de extensión.
- Se entiende que se agregan pasos a un Caso de Uso existente, y esto se hace creando un nuevo Caso de Uso, que enriquece al existente, pero no lo modifica.



## Composición general No estandarizada:

- Resumen de la Identificación.
- Descripción de Recursos.
- Definición de Conversación.
- Requisitos de Interfaz de usuario
  - Requisitos funcionales.
  - Requisitos no funcionales.

## Pre-condiciones:

- la representación solicitada existe en el teatro.
- hay asientos disponibles para la función solicitada.


## Post-condiciones:

- el vendedor posee más efectivo en la caja registradora
- el nº de asientos disponibles para dicha representación es menor que al comienzo del C.U.

Qué elemento se puede desprender de las postcondiciones?



# Conversación del C.U. “Comprar entradas por reserva”

ACCIONES DEL 	RESPUESTA DEL SISTEMA	
	Curso Normal	Curso Alternativo
1. – Comprar Entrada x Reserva		
	2. – <u>Validar la reserva</u>	
		2.1 – La reserva está vencida → RECHAZAR LA COMPRA.
	3. – <u>Efectivizar Compra</u>	
	4. – Eliminar Reserva	
	5. – Generar Boleto	
	6. – Solicitar forma de pago	
7. – Confirmar pago (Tarjeta de débito)		
	8. – <u>Validar Tarjeta de débito</u>	
		8.1. – La tarjeta se encuentra dañada → RECHAZAR EL COBRO Y DAR AVISO.
	9. – Efectuar cobro del boleto	
		9.1. – La tarjeta No posee saldo disponible → RECHAZAR EL COBRO Y DAR AVISO.
	10. – Imprimir Boleto de Entrada	
	11. – Entregar Boleto impreso y el medio de pago (Tarjeta de débito)	

## Curso de Error:

1.E1 - No funciona el sistema de gestión de entradas

→ *se realizará la operación en forma manual, registrando el asiento vendido.*

8.E1 - No se puede establecer conexión con la red de tarjeta de débito

→ *dar aviso al cliente e informar sobre otras formas de pago.*

10.E1 - No funciona la impresora de boletos

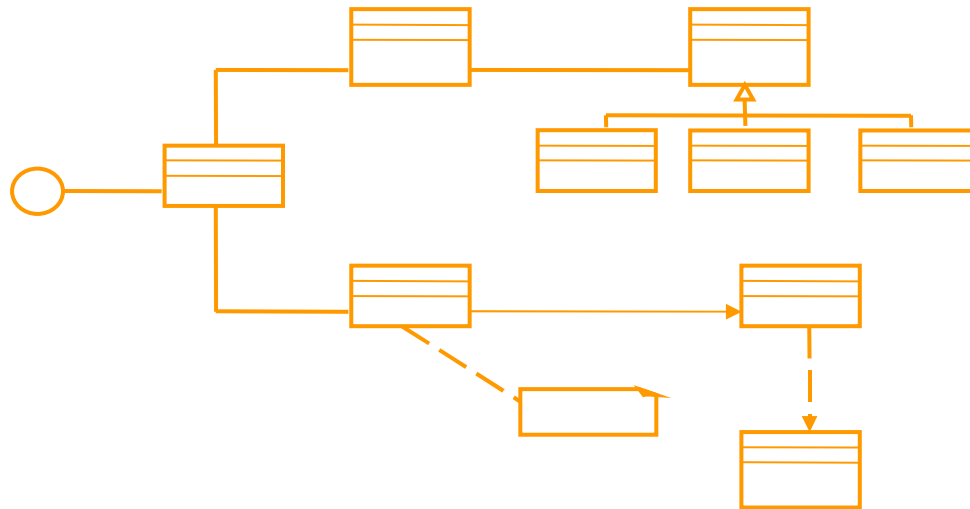
→ *dar aviso al cliente y entregar el boleto a cliente, llenando el formulario correspondiente en forma manual.*

# 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:

NombreClase
atributos
operaciones
responsabilidades

La visibilidad puede ser:

- Público (+): objetos de cualquier clase.
- Protegido (#): objetos de las subclases.
- De paquete (~): objetos de cualquier clase del mismo paquete.
- Privado (-): objetos donde está definido el atributo u operación.

¿Qué pasa en Smtalltalk y otros lenguajes?

## Sintaxis

[visibilidad] nombre [multiplicidad][:tipo][=valor inicial][{lista\_propiedades}]

## Propiedades:

- modifyable
- readOnly
- frozen

## Ejemplos

Representación
id
nombre
responsable
género

Representación
-id: Integer {frozen}
+nombre:String
responsable [1..2]:String
#genero:String=musical



- **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.



## Propiedades

- isQuery
- sequential
- guarded
- concurrent

## Ejemplos

Representación
nuevaRep() nombreRep() estaVigente()

Representación
+nuevaRep( <b>in</b> nombre:String) #nombreRep() estaVigente(): Boolean {isQuery}

## ¿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 existen?

- Asociación
- Generalización
- Dependencia
- Realización

# Diagrama de clases: relaciones

- **Notación:**

- Asociación



- Agregación



- Composición



- Generalización



- Dependencia



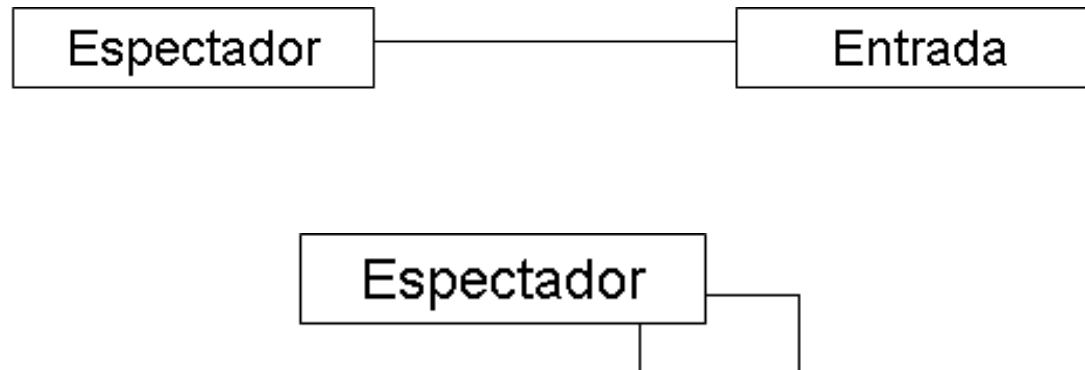
- Realización



## ¿Qué es una Asociación?

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

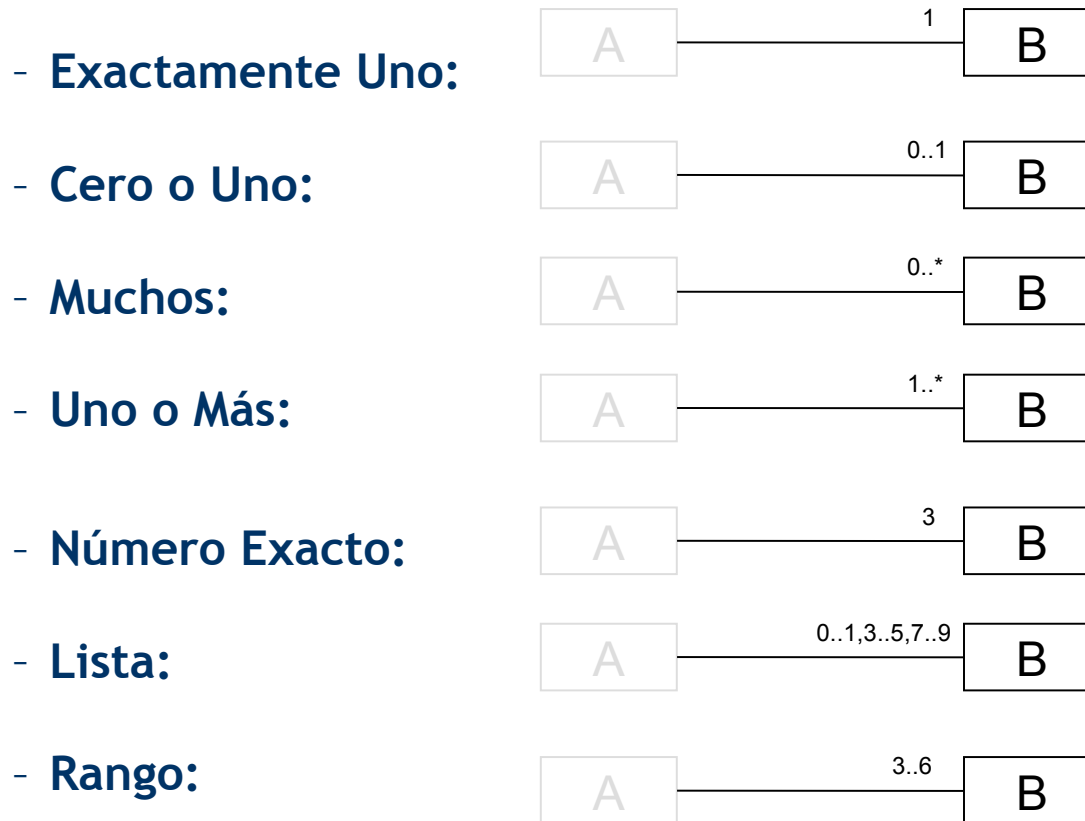
## Ejemplo



# Asociación: propiedades (cont.)

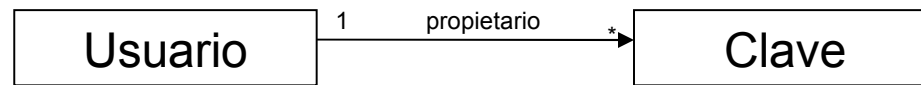
**Multiplicidad:** indica “cuántos” objetos pueden conectarse a través de una instancia de una asociación.

- Se puede indicar una multiplicidad de:



# Asociación: propiedades

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

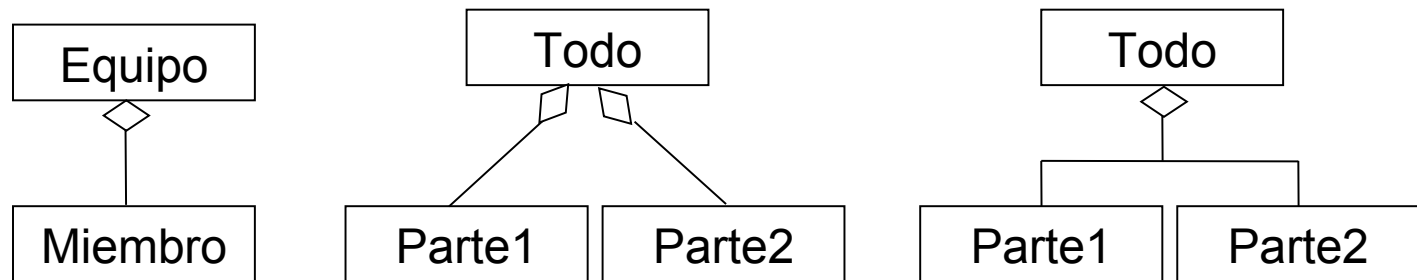


- **Visibilidad:** sirve para limitar la visibilidad a través de esta asociación relativa a los objetos externos a ella.
  - Pública (+)
  - Protegida (#)
  - Privada (-)

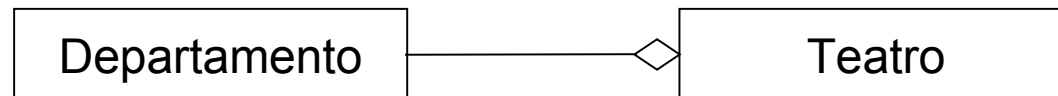


# Asociación: propiedades (cont.)

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



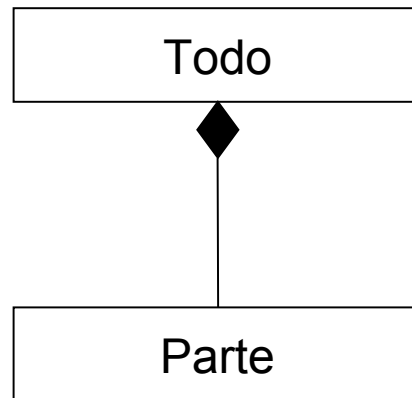
**Ejemplo:**



# Asociación: propiedades (cont.)

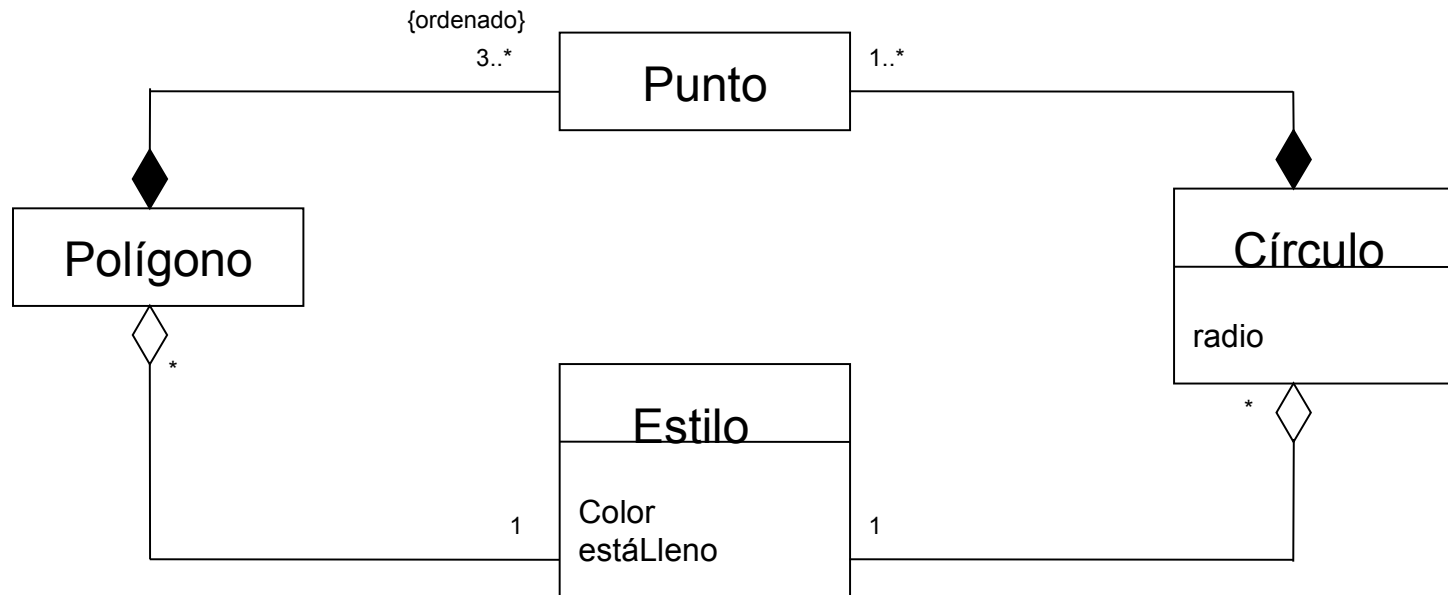
**Composición:** es una forma de agregación, con fuerte sentido de posesión y tiempo de vida coincidentes de las partes con el conjunto.

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





## Ejemplos de Agregación y Composición

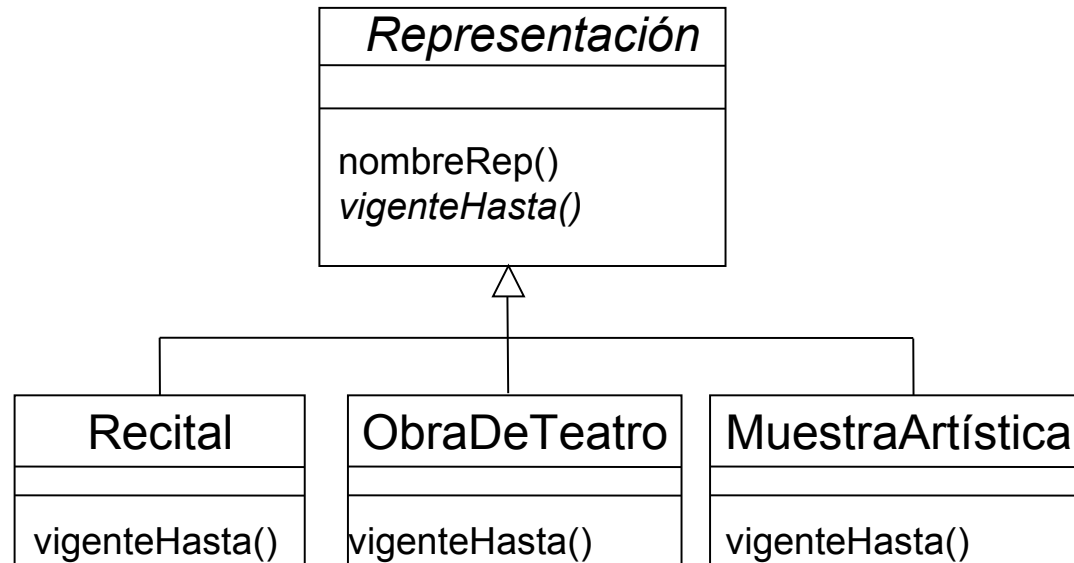


# Relaciones: Generalización

## ¿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

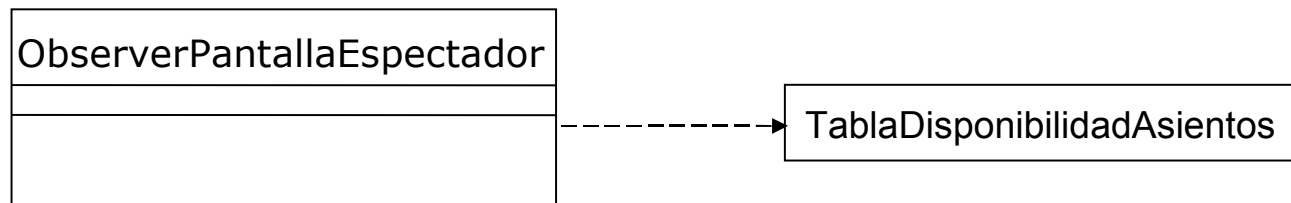


- En UML puede representarse herencia múltiple

## ¿Qué es una Dependencia?

Es una relación de uso que declara que un cambio en la especificación de un elemento puede afectar a otro elemento que la utiliza, pero no necesariamente a la inversa.

## Ejemplo



## ¿Qué es una Interfaz?

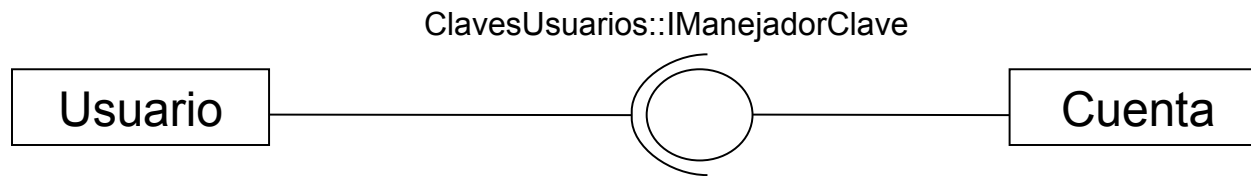
Es una colección de operaciones que se utiliza para especificar un servicio de una clase o componente.

## Notación

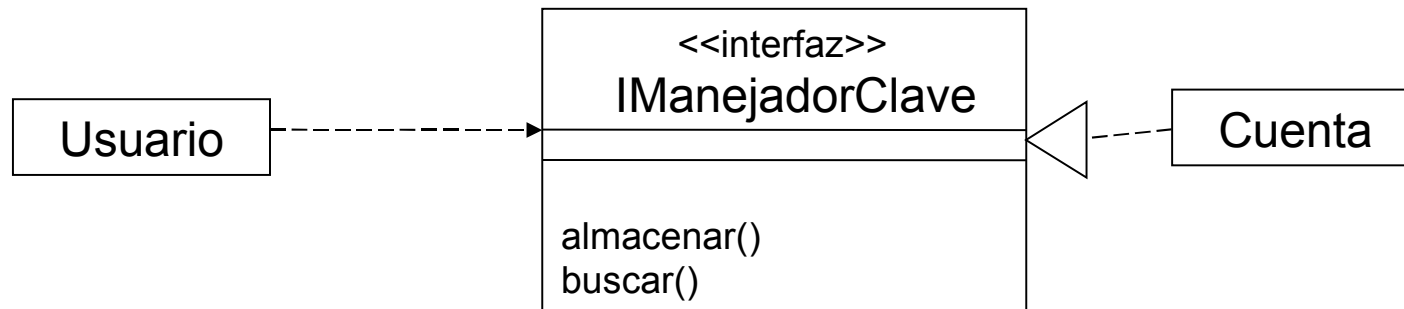
- **Nombre:** sirve para distinguirla del resto de las interfaces.
- **Operaciones:** se utilizan para especificar un servicio.
- **Relaciones:** participan en las relaciones de asociación, generalización y dependencia.

# Interfaces: utilización

- Notación simple:



- Notación expandida:



# 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: contenido

- Objetos



objeto:Clase

- Enlaces



- Mensajes



- Pueden contener:

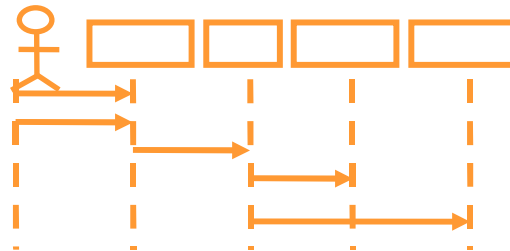
- notas y restricciones.



# Diagrama de Secuencia

## Definición:

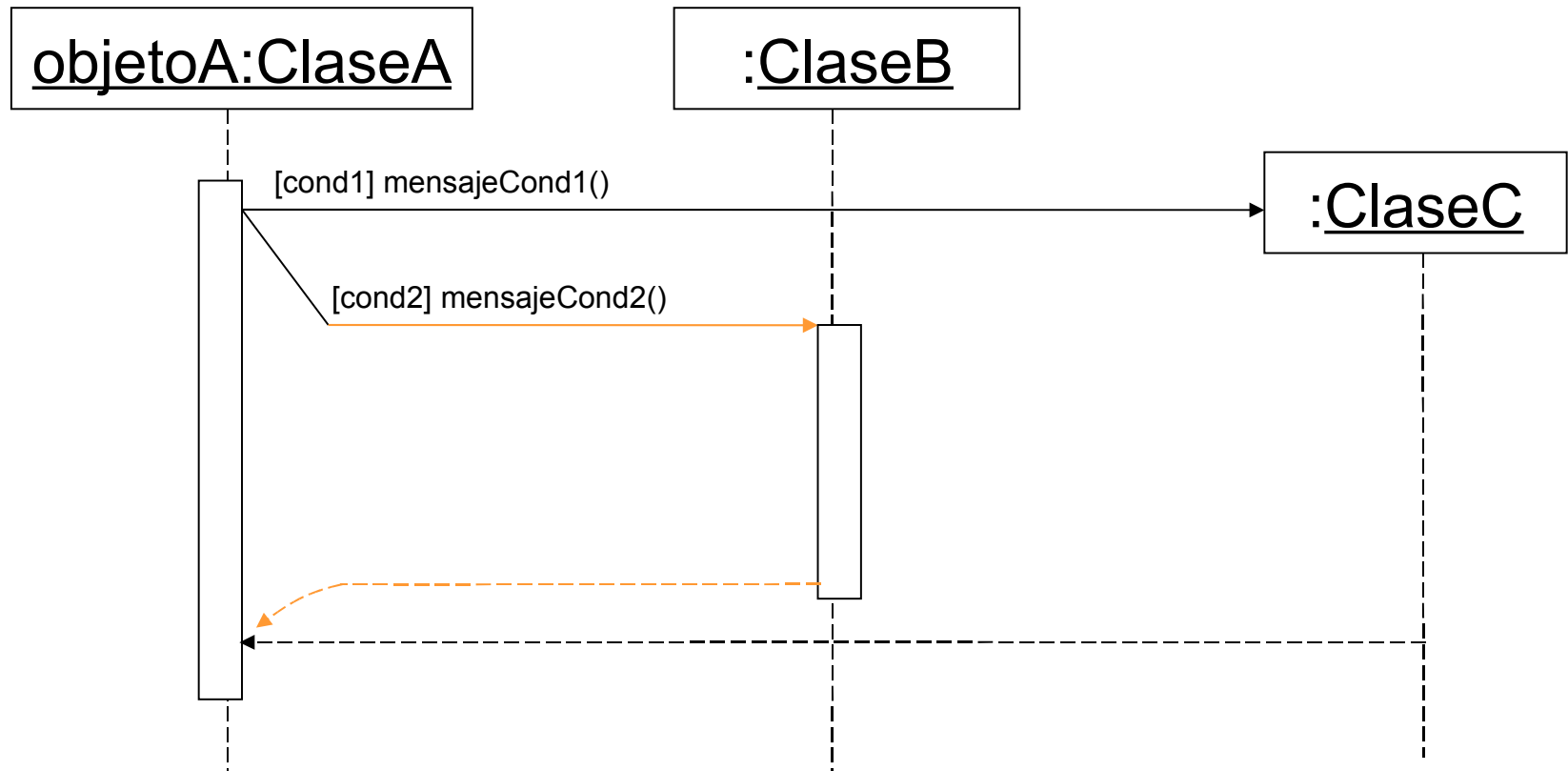
Un diagrama de secuencia destaca el orden temporal de los mensajes.



## Sintaxis:

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

# Bifurcaciones

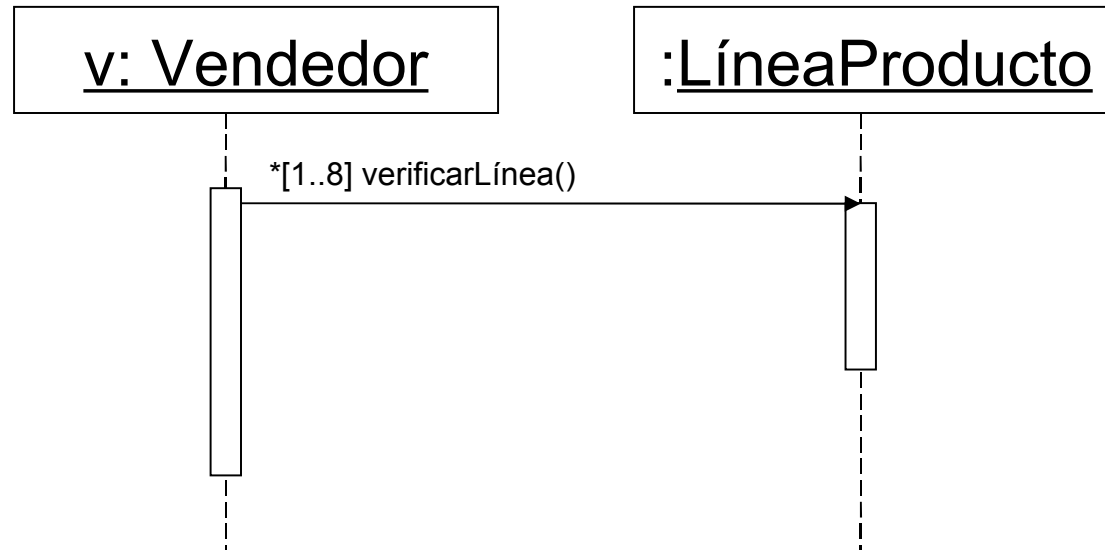


# Iteración o bucle

- Sintaxis:

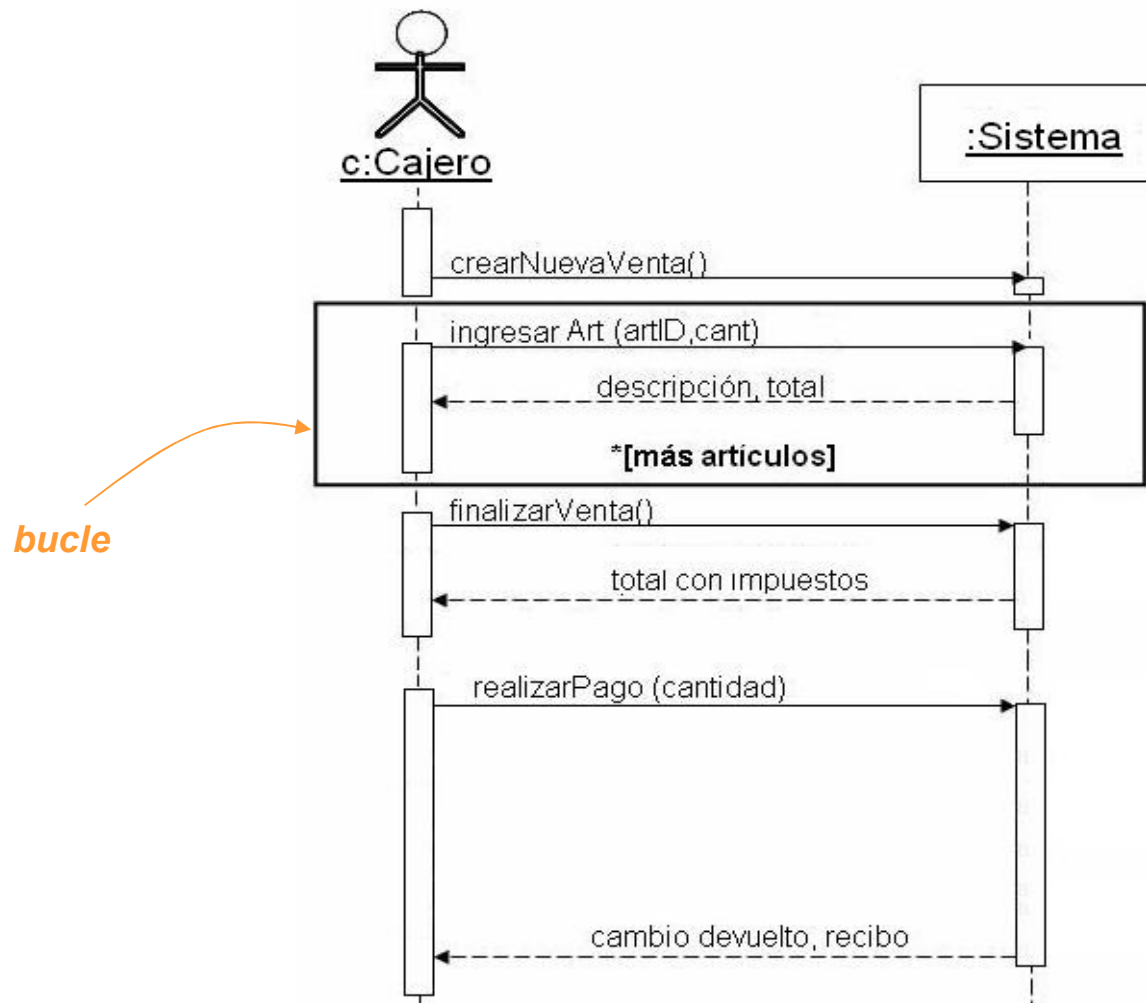
\* [expresión-iteración ] mensaje

- Ejemplo:

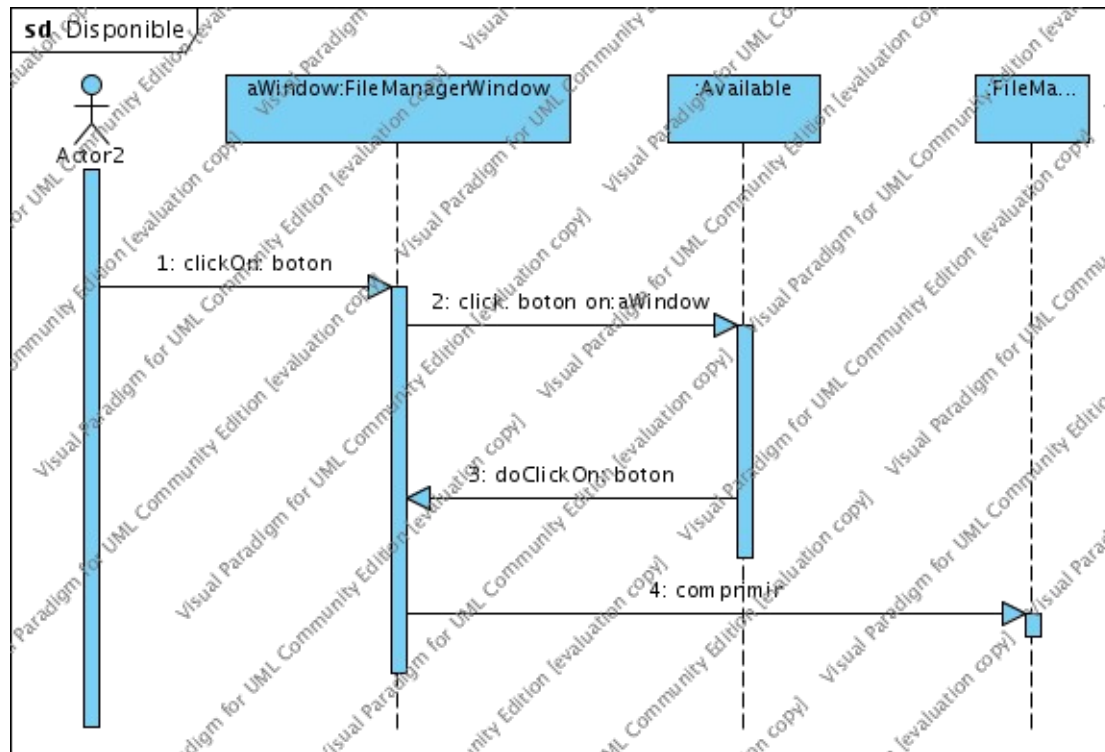


# Iteración de una serie de mensajes

- Ejemplo:



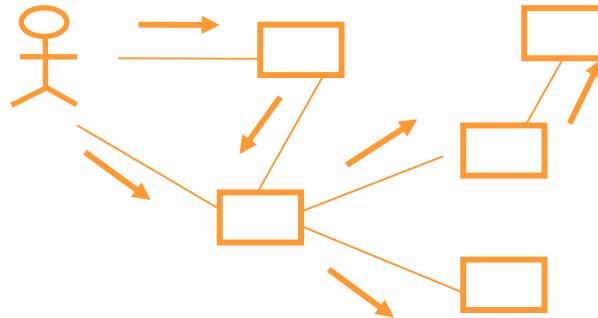
# Ejemplo



# Diagrama de Colaboración

## Definición:

Un Diagrama de Colaboración destaca la organización estructural de los objetos participantes y el envío de mensajes.



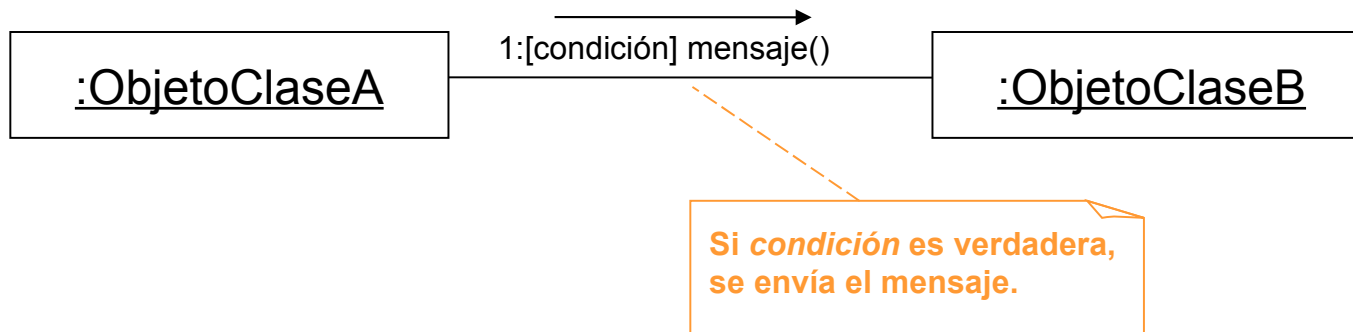
## Sintaxis:

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

# Mensajes condicionales

Un mensaje condicional es aquel que se envía si la evaluación de la cláusula es *verdadera*.

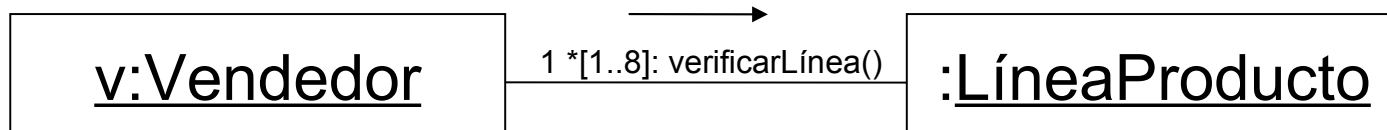
**Notación:**



- Sintaxis:

\* [expresión-iteración ] mensaje

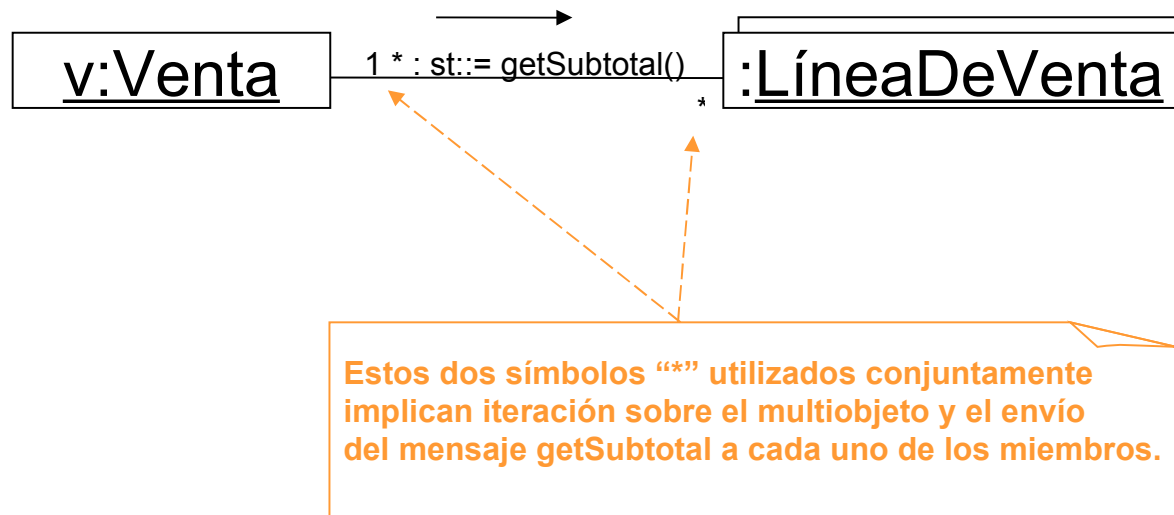
- Ejemplo:





# Iteración sobre una colección

- Los multiobjetos se utilizan para denotar un conjunto de instancias -colección-.
- Ejemplo:



# Diagrama de Máquina de Estados

- **Los Diagramas de Máquina de Estados**

- Son útiles para modelar la vida de un objeto.
- Describen los estados por los que puede pasar un objeto durante su ciclo de vida y el comportamiento en esos estados junto con los eventos que causan los cambios de estado.

- **Los Diagramas de Máquina de Estados pueden asociarse a**

- Clases
- Casos de Uso
- Sistemas Completos

para visualizar, especificar, construir y documentar la dinámica de un **objeto individual**.

# Diagrama de Máquina de Estados

Las Máquinas de Estados pueden componerse de:

- **estados**

- estados simples.



Nombre\_del\_estado

- estados compuestos.



Nombre\_del\_estado

NombreSubestado

NombreSubestado

- **transiciones**

- eventos.
- acciones.

# Diagrama de Máquina de Estados

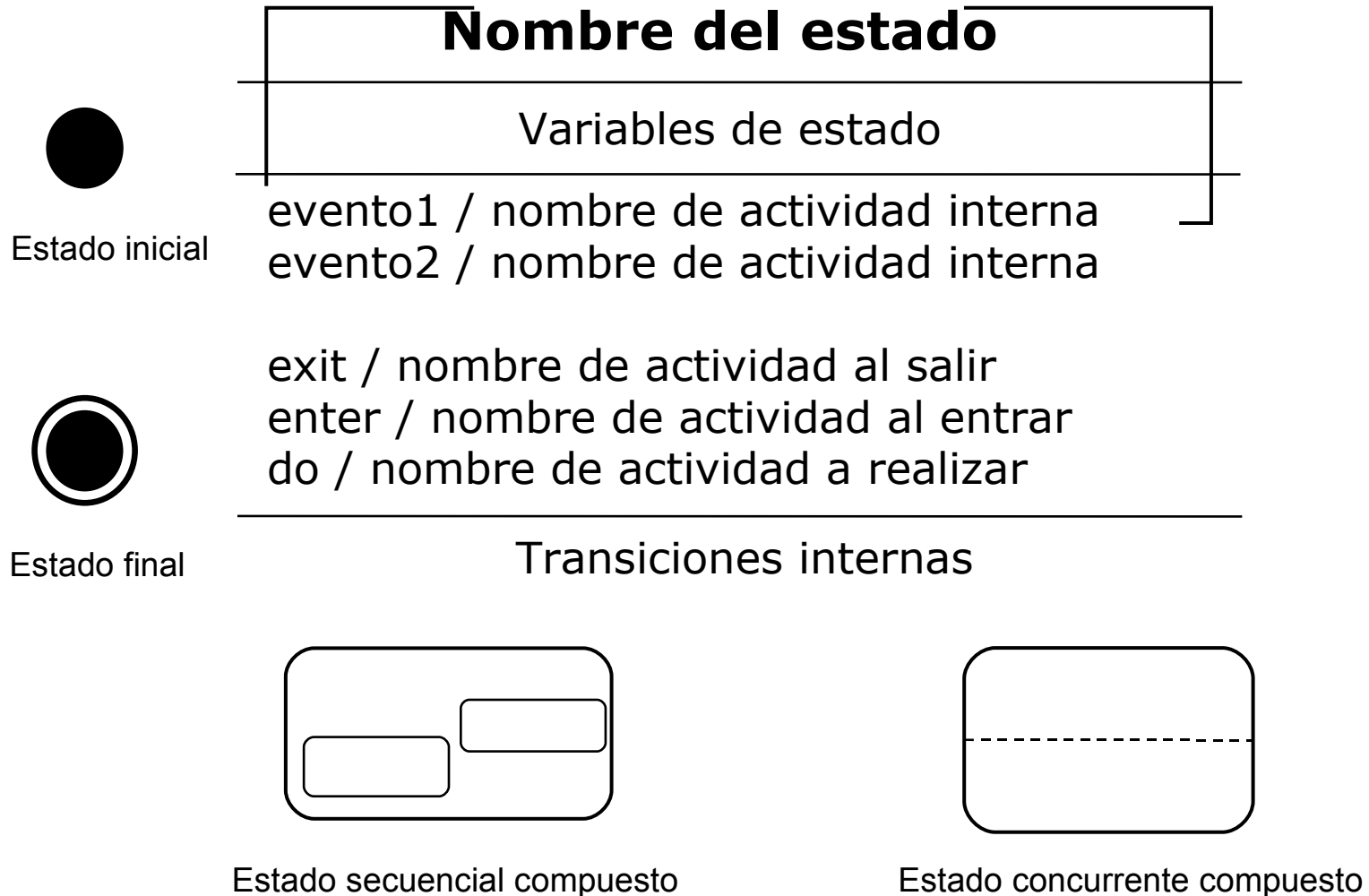
## Estado:

es una condición en la que puede estar un objeto en algún momento de su ciclo de vida, durante un cierto tiempo.

Mientras está en un determinado estado, el objeto puede llevar a cabo algunas (o todas) de las siguientes acciones:

- realizar una actividad.
- esperar un evento.
- satisfacer una o más condiciones.

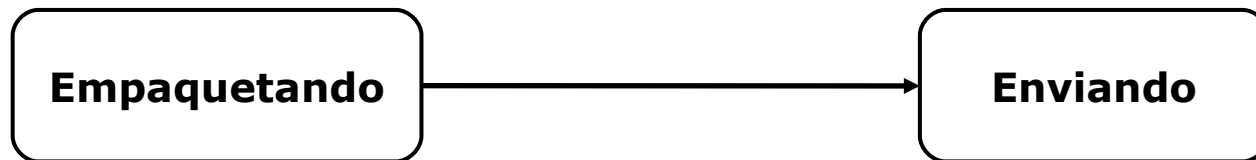
# Diagrama de Máquina de Estados



# Diagrama de Máquina de Estados

- **Transiciones:**

una transición es un cambio del objeto desde un estado (*estado fuente/origen*) a otro (*estado destino*).

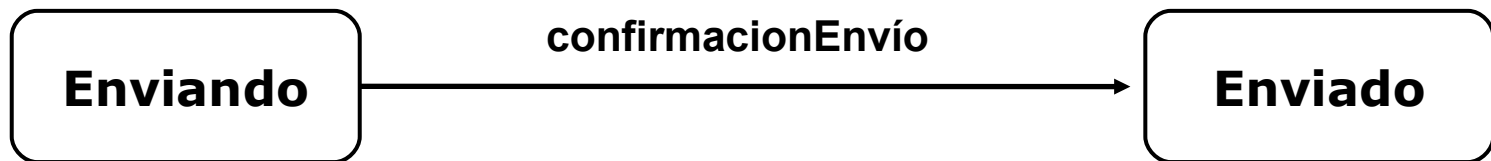


También puede haber una auto-transición cuando ambos estados coinciden.

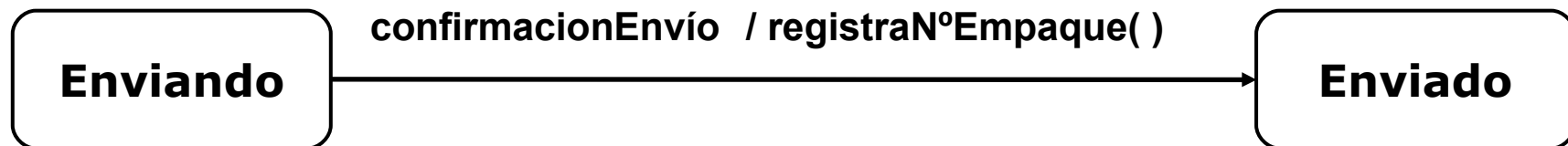


# Diagrama de Máquina de Estados

**Evento:** es la especificación de un acontecimiento significativo.



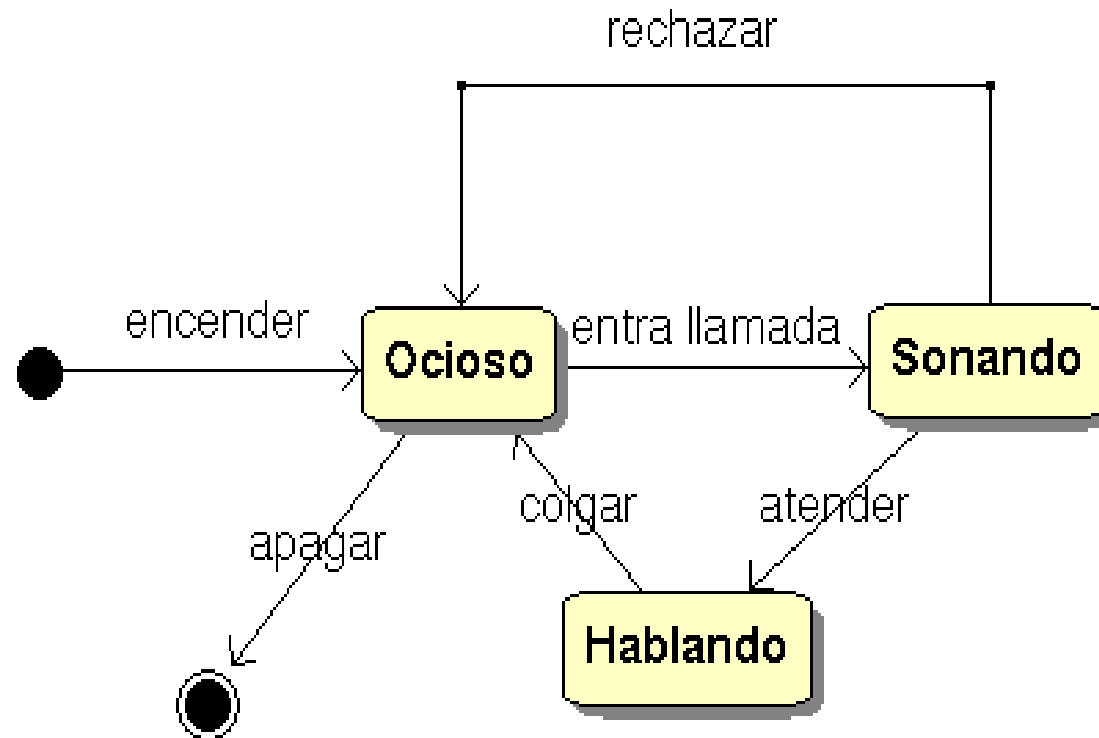
**Acción:** es una computación que produce un cambio de estado en el modelo o la devolución de un valor.





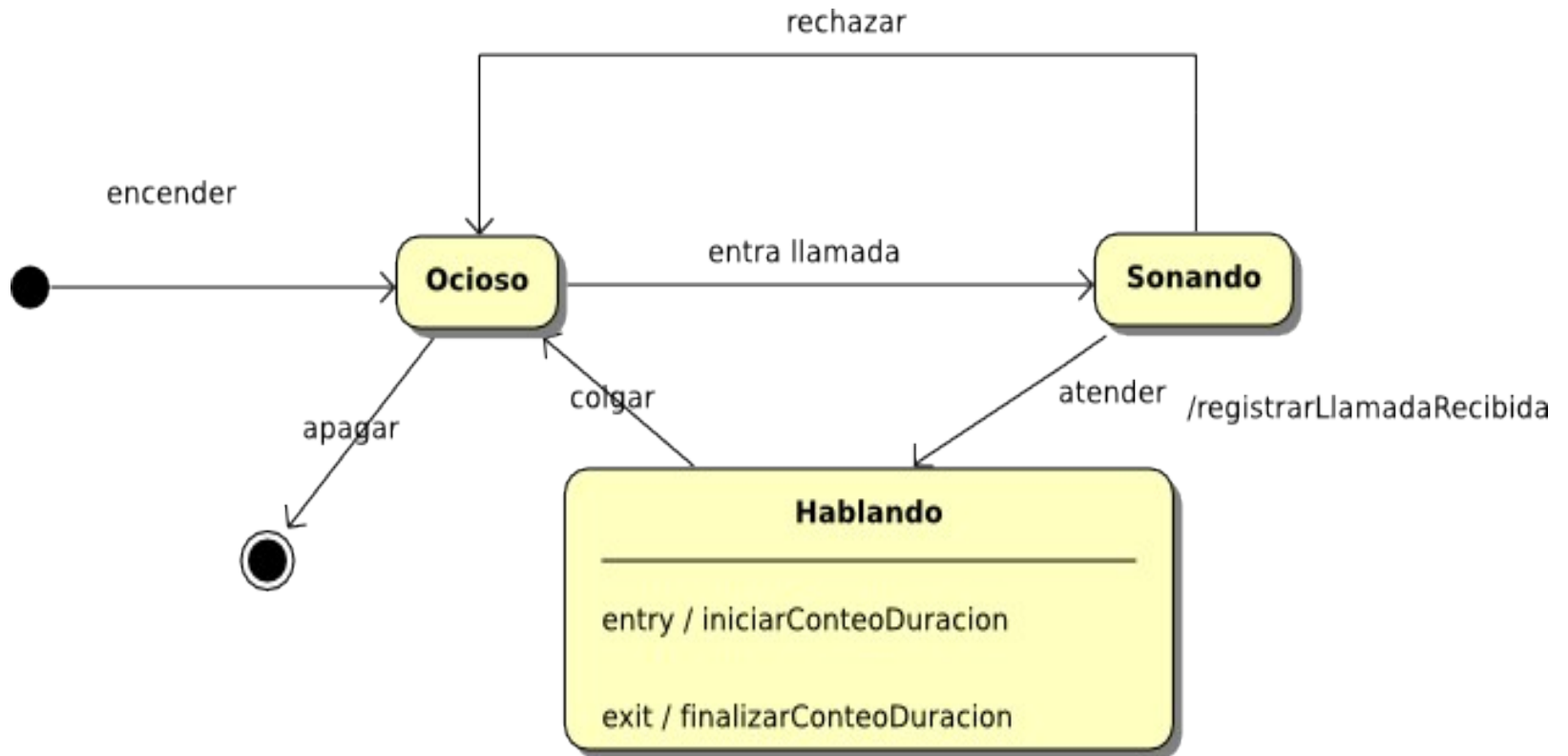
# Diagrama de Máquina de Estados

Ejemplo: teléfono



# Diagrama de Máquina de Estados

Ejemplo: teléfono



- Partes que no se vieron:
  - Packages
  - Colaborationa
  - Activity
  - Deployment/Component
  - Object Constraint Language
  - Metamodel (MOF), Profiles, XMI

- UML produjo un gran impacto en la comunidad de objetos a finales de los 90's.
- UML es la notación estándar defacto.
- Muchos de los “metodologistas” de los 80's se volcaron a UML.
- UML sigue creciendo en detalles y complejidad.
- Es difícil expresar comportamiento (y threads).
- Es la base de otras áreas como Model Driven Architecture (MDA).