INFORMÁTICA INDUSTRIAL

Modelado. El lenguaje Unificado de Modelado.

M. Abderrahim, A. Castro, J. C. Castillo
Departamento de Ingeniería de Sistemas y Automática

uc3m | Universidad Carlos III de Madrid

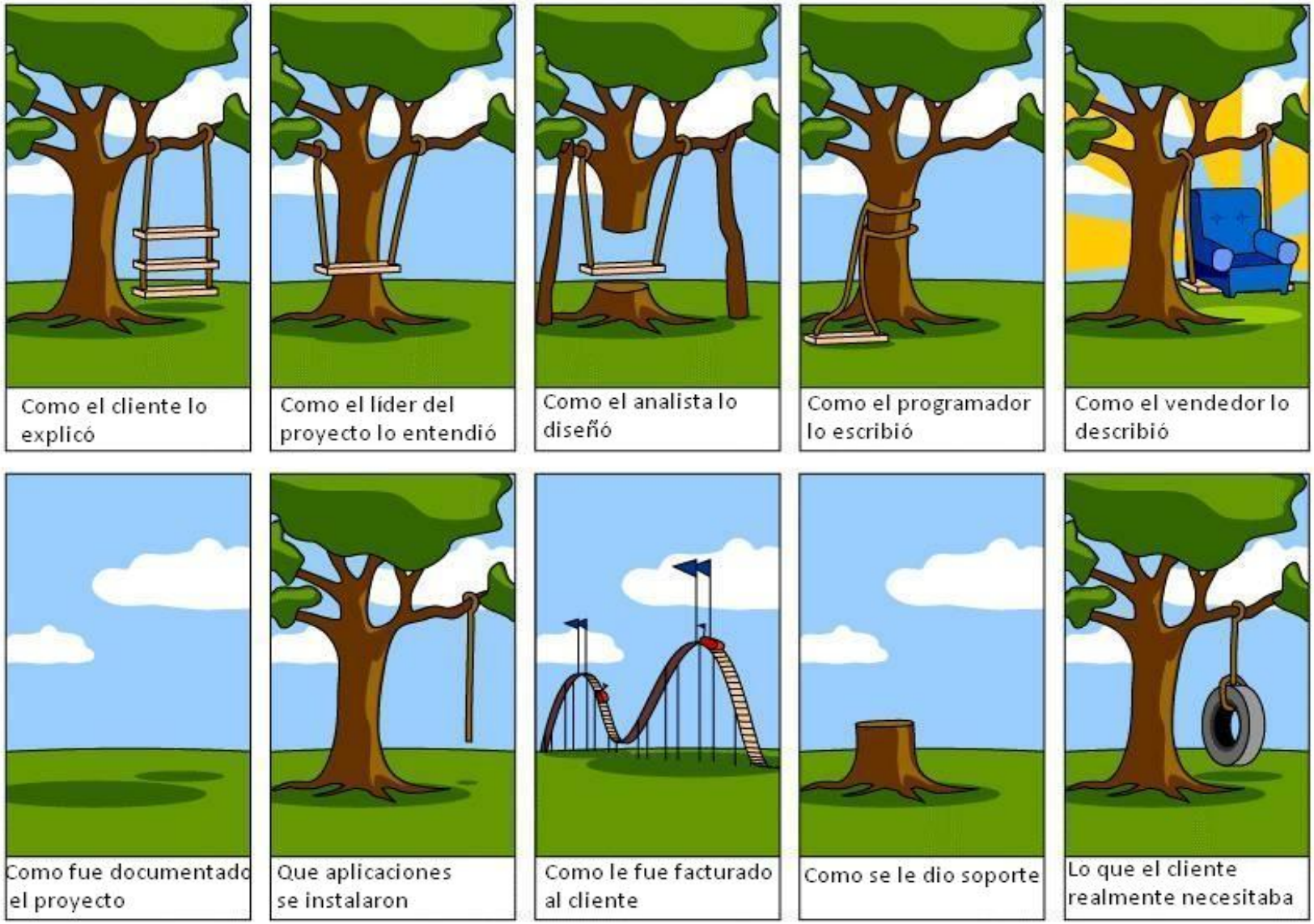
El Lenguaje Unificado de Modelado

- **¿Qué es UML?**
- Diagrama de **Clases**
- Notas
- Diagrama de **Secuencia**
- Diagrama de **Actividad**
- Diagrama de **Estados**
- Diagrama de **Casos de Uso**

¿Qué es UML?

- Consiste en un conjunto integrado de diagramas definidos para ayudar a los desarrolladores de software y de sistemas a realizar las tareas de:
  - Especificación
  - Visualización
  - Diseño Arquitectónico
  - Construcción
  - Simulación y pruebas
  - Documentación

¿Por qué es necesario?



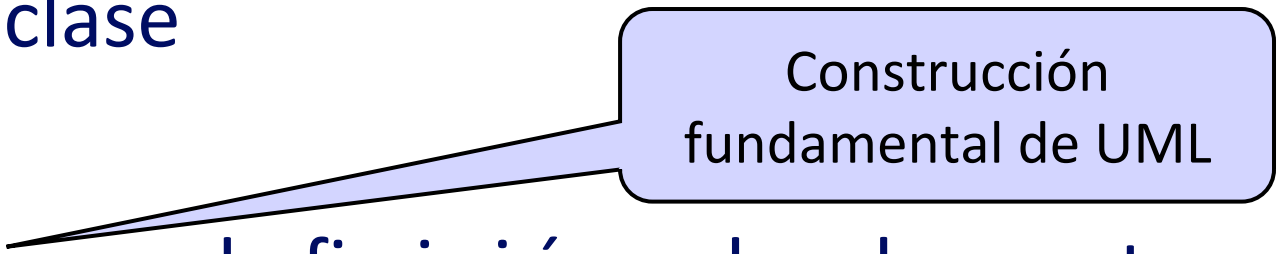
https://www.tamingdata.com/2010/07/08/the-project-management-tree-swing-cartoon-past-and-present/

Conceptos de Modelado Estructural

- **Objeto:** representación de algo que se describe mediante un *identificador*, una *estructura (propiedades)* y un *comportamiento*
- **Propiedades de los objetos:**
  - **Atributos:** propiedades relevantes de un objeto que representa su estructura (diferentes objetos tienen algunos valores diferentes)
    - Simples: un solo valor
    - Compuestos: múltiples valores
  - **Relaciones:** asociaciones entre los objetos

Conceptos fundamentales

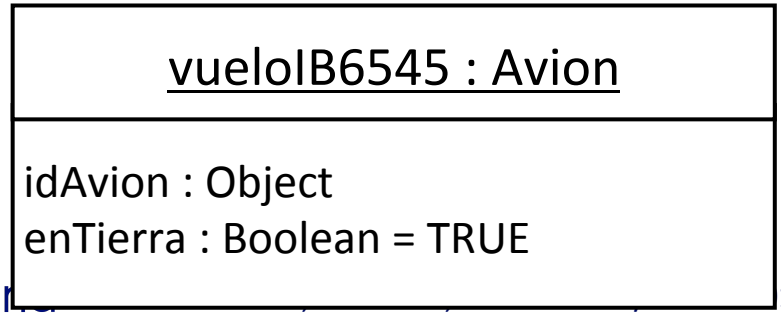
- **Clasificación:** agrupación de objetos con propiedades y comportamiento similares dentro de una clase



- **Clase:** definición de la estructura y el comportamiento de un conjunto de objetos que tienen el mismo patrón estructural y de comportamiento
- **Instancia:** Cada objeto que pertenece a una clase

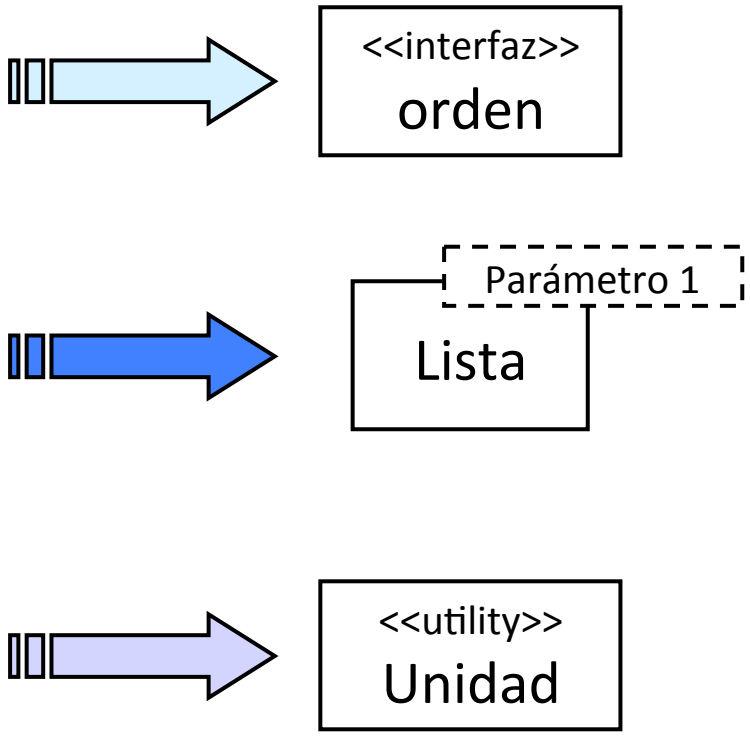
Conceptos fundamentales

- **Extensión de clase:** conjunto de todas las instancias de una clase
- **Generación de objetos**
  - **Instanciación:** proceso de generación o creación de las instancias de una clase
- **Objetos de una clase**
  - **nombre del objeto: nombre de la clase** a la que pertenece el objeto, ambos subrayados
  - Cada atributo definido en la clase tendrá su valor específico



Conceptos fundamentales

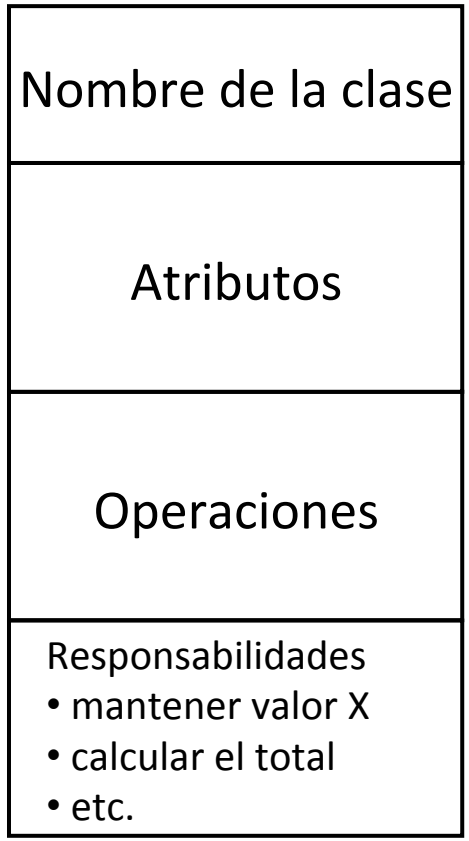
- **Clases: definen el vocabulario básico de un modelo y fundamentan el modelado de datos**
  - **Interfaz:** clase asociada que describe su comportamiento visible.
  - **Clases abstractas o parametrizables:** modelos de clases que se corresponden con clases genéricas. No son instanciables.
  - **Clases utilitarias:** librerías de funciones



Conceptos fundamentales

- **Responsabilidades de una clase**
  - Representa las obligaciones que una clase tiene en relación al resto de las clases
  - Normalmente están soportadas por las operaciones de la clase

- Se pueden agregar compartimentos al final de las clases para representar cualquier información necesaria en el modelo



Abstracción

- La técnica de hacer un modelo de tus ideas del mundo es el uso de la **abstracción**
  - Por ejemplo, un mapa es un modelo del mundo, no el mundo en miniatura
- En los diagramas UML se muestra una abstracción del sistema, no todo el sistema, con el objetivo de que sea fácil de entender

Puntos de vista

- UML permite crear diagramas que reflejan diferentes **puntos de vista** del mismo sistema.
  - Por ejemplo, hay mapas físicos, mapas políticos, mapas históricos … todos sobre el mismo mundo

- Esto permite mostrar ciertos aspectos y ocultar otros para que sean más fáciles de comprender

Diagramas del Lenguaje Unificado de Modelado

- Diagrama de **Clases**
- Notas
- Diagrama de Paquetes
- Diagrama de Objetos
- Diagrama Estructural Compuesto
- Diagrama de **Secuencia**
- Diagrama de **Actividad**
- Diagrama de Interacción en Visión General
- Diagrama de Comunicación
- Diagrama de **Estados**
- Diagrama de Componentes
- Diagrama de Despliegue
- **Diagrama de Casos de Uso**
- …

Tipos de diagramas

- **Diagramas Estructurales:** Muestran los elementos de construcción del sistema. Características que no cambian con el tiempo
- **Diagramas de Comportamiento:** Muestra como el sistema responde a las peticiones o evoluciona con el tiempo.
- **Diagramas de Interacción:** Engloba a ciertos diagramas de comportamiento que muestran el intercambio de mensajes dentro de un grupo de objetos que cooperan (colaboración) para obtener un objetivo

Diagramas y Modelo

- UML define el formato de un conjunto de diagramas
- Un modelo representa los elementos del sistema
  - Clases con sus métodos y atributos
  - Objetos con sus relaciones
- Cuando existe un conjunto “consistente” de diagramas se forma un modelo
- Es posible que elementos del modelo no aparezcan en ningún diagrama

El Lenguaje Unificado de Modelado

- ¿Qué es UML?
- Diagrama de Clases
- Notas
- Diagrama de Secuencia
- Diagrama de Actividad
- Diagrama de Estados
- Diagrama de Casos de Uso

Diagrama de clases

- Diagrama estructural que muestra entidades del mundo real, elementos de análisis y diseño o clases de implementación y sus relaciones
- Los diagramas de clases están formados por
  - Clases
  - Relaciones
    - Asociación
    - Herencia
    - Agregación
    - Composición
    - Uso (Dependencia)
    - Realización

Diagrama de clases

- Formato gráfico de una clase
  - Atributos (Propiedades)
  - Métodos (Operaciones)

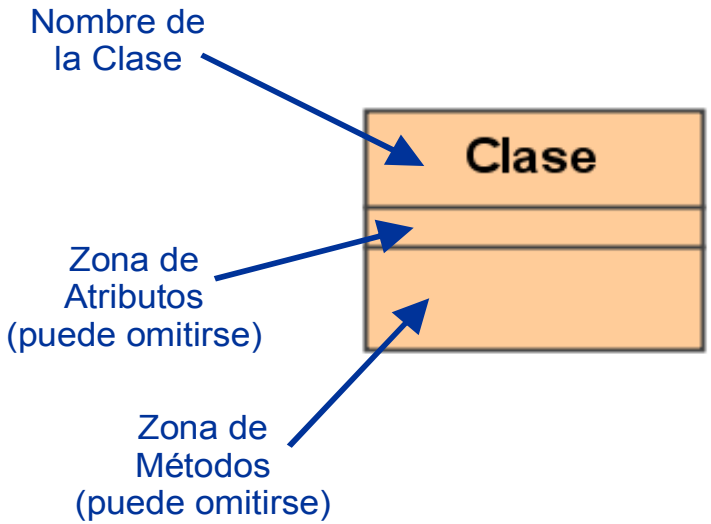


Diagrama de clases

Atributos (Propiedades)

- Sintaxis de los atributos

nombreAtributo : tipoAtributo

- Tipos
  - Tipos UML: Integer, Boolean, String
  - Tipos de cualquier lenguaje de programación
  - Clases del modelo
  - Se pueden omitir en bocetos

- Ejemplo:

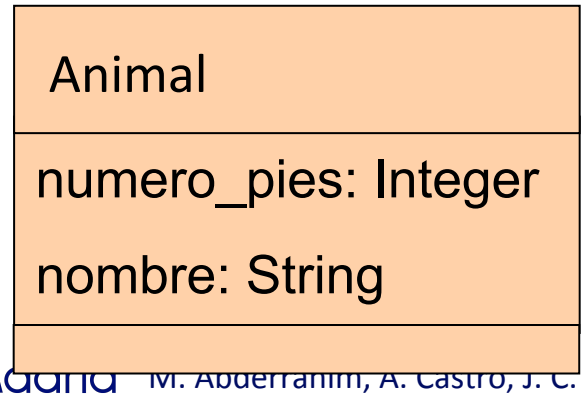


Diagrama de clases

Atributos (Propiedades)

- Valores por defecto

```
nombreAtributo : tipoAtributo = valorPorDefecto
```

```
numero_pies: Integer = 5
nombre: String = 'Pepe'
```

- Multiplicidad (Arrays)

```
nombreAtributo : tipoAtributo [Multiplicidad]
```

```
numero_pies: Integer [0..1]
nombres: String [1..*] = ('Jose', 'Miguel', 'Antonio')
fechas: Date [2,3,4]
```

Diagrama de clases

Métodos (Operaciones)

- Sintaxis Métodos

nombreOperacion(parametros) : tipoDevuelto

- Ejemplos

Animal
numero_pies: Integer nombre: String
cazar (otro_animal : Animal): Resultado tiene_crias(): Boolean comer (una_comida: TipoComida, float: cantidad) comer (TipoComida, float)

Diagrama de clases

Visibilidad

Símbolo	Visibilidad
+	Público
-	Privado
#	Protegido
~	Paquete

- Ejemplos

Animal
numero_pies: Integer - nombre: String
+ cazar (otro_animal : Animal): Resultado + tiene_crias(): Boolean + comer (una_comida: TipoComida, float: cantidad) + comer (TipoComida, float)

Diagrama de clases

Miembros Estáticos

- Los miembros (atributos o métodos) estáticos se subrayan

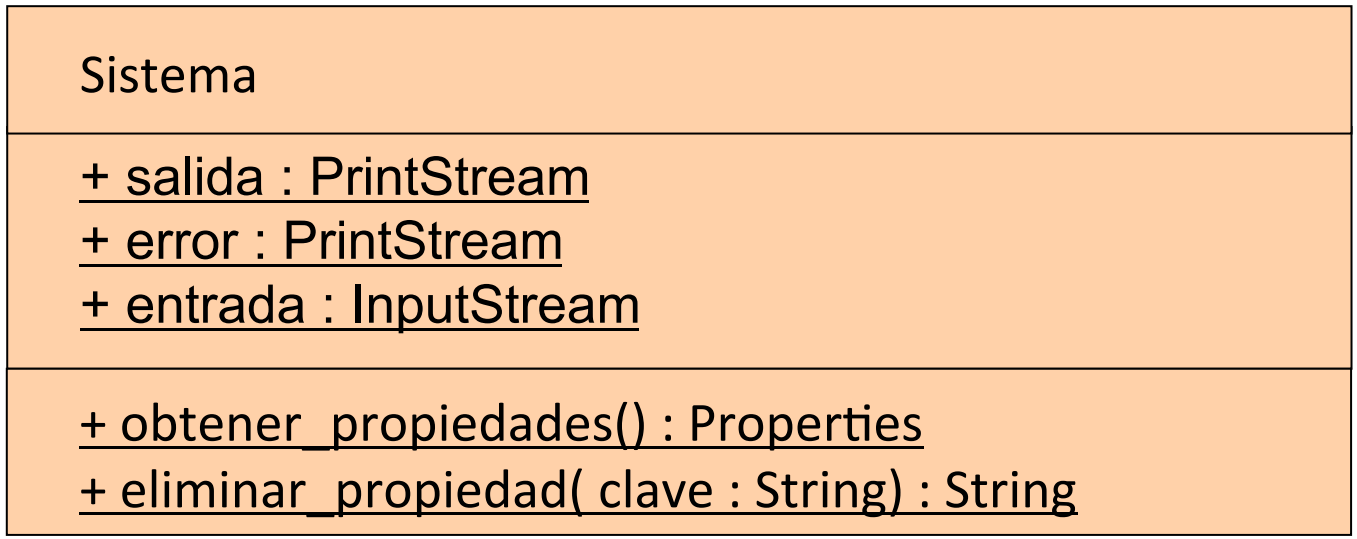


Diagrama de clases

Elementos Abstractos

- Los elementos abstractos aparecen en cursiva (Clases o métodos)

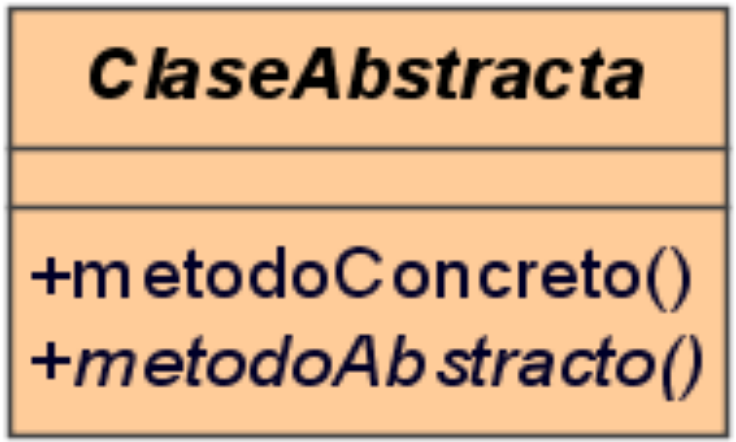


Diagrama de clases

Relación de Asociación

- Relación estructural para indicar la vinculación entre dos clases
- Una asociación se forma al unir dos clases con una línea
- Puede tener nombre y se coloca sobre la línea (se lee de izquierda a derecha y suele ser un verbo)
- Puede tener los roles que juega cada clase en la asociación

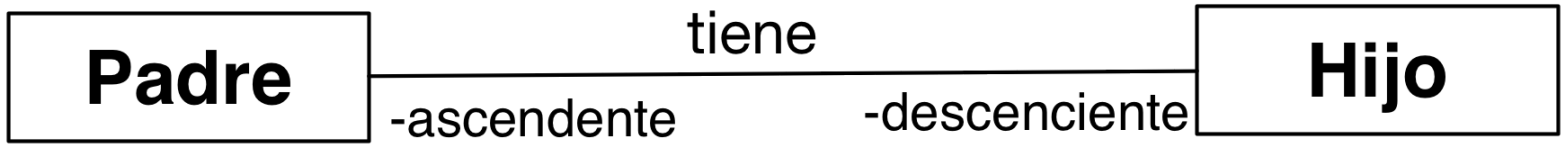


Diagrama de clases

Relación de Asociación

- Puede tener Multiplicidad (*, 0..1, 1..*)
- Puede tener restricciones entre { }
- Se puede indicar el sentido de la lectura

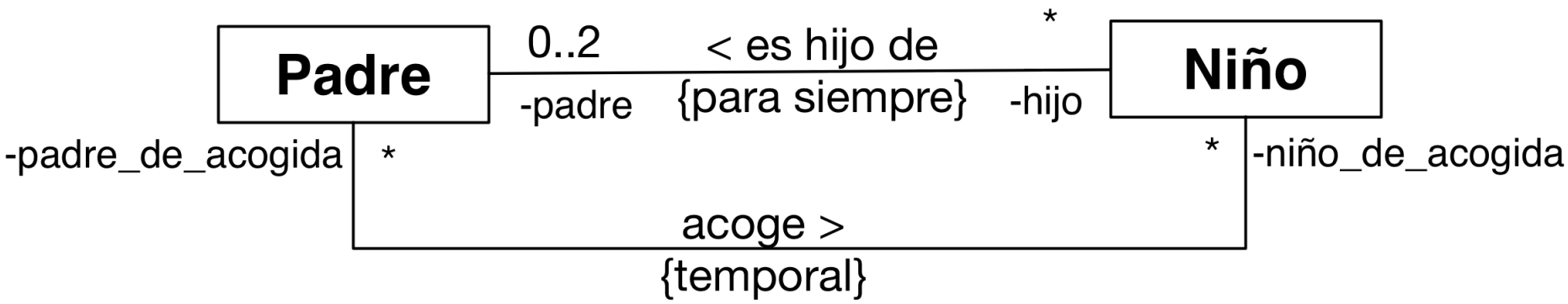


Diagrama de clases

Relación de Asociación

- Puede tener navegación direccional (sólo es posible la navegación en un sentido)
- Es posible que una clase se asocie consigo misma (asociación reflexiva)

```
public class Persona{
   ...
    List<Persona> hijos;
    ...
}
```

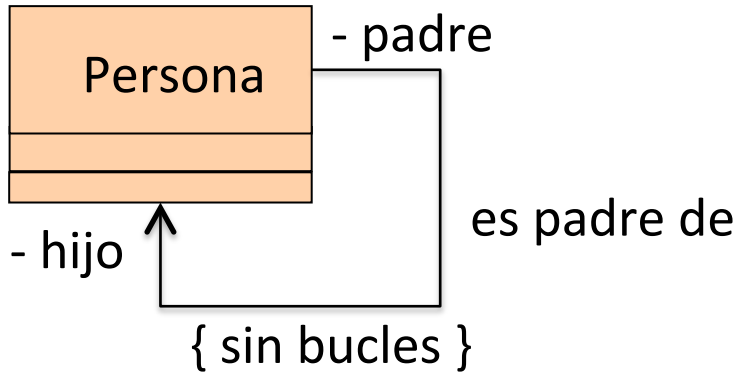


Diagrama de clases

Relación de Asociación

- Ejemplo

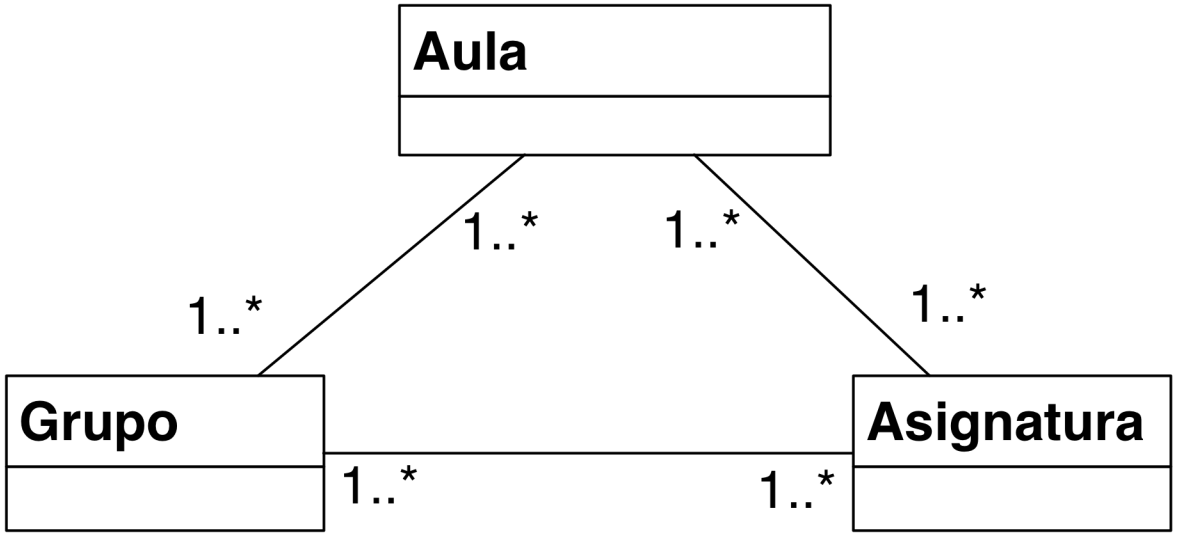


Diagrama de clases

Relación de Asociación

- Clases de Asociación
  - Similares a las relaciones atribuidas de BBDD
  - La clase de asociación debe llamarse como la asociación

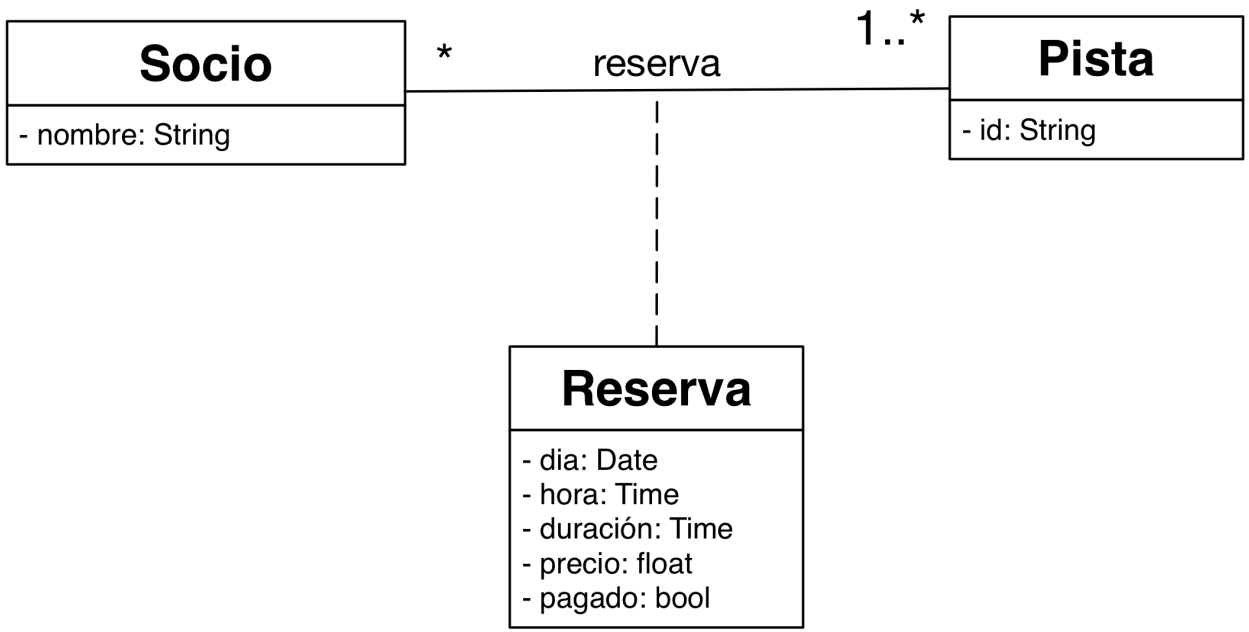


Diagrama de clases

Relación de Agregación

- Una **relación** de agregación es la que forma un **todo con sus partes**
- Son un tipo especial de relación de asociación
- Pueden tener nombre, roles, multiplicidad, …
- En las relaciones de agregación, un objeto que representa una **parte** puede estar **compartido** por varios objetos que representan el **todo** (un alumno está en un curso y también puede estar en un grupo de amigos)

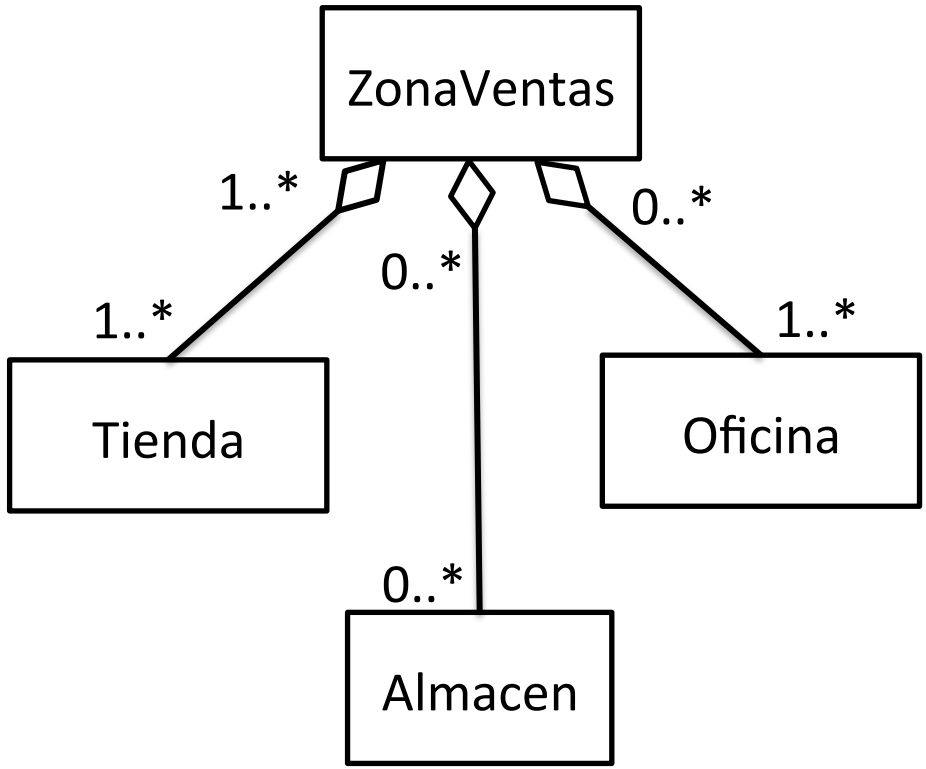
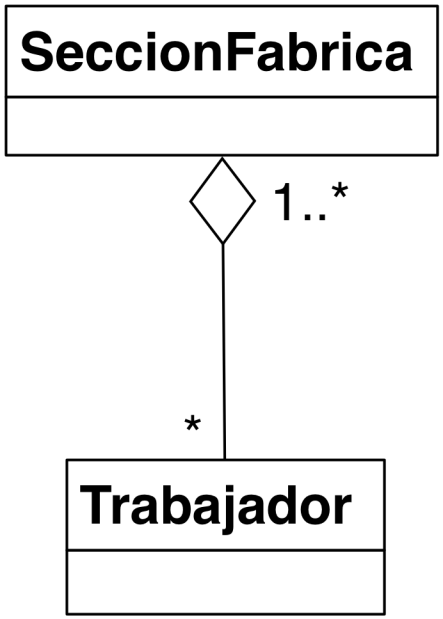


Diagrama de clases

Ejemplo de relación de agregación




```
class Trabajador {
    …
}

class SeccionFabrica {
    …
    Vector<Trabajador*> operarios;
    …
}
```

Diagrama de clases

Relación de Composición

- Las relaciones de **composición** son un tipo especial de relación de **agregación**
- Los objetos **parte** siempre están asociados a **un** objeto **todo** y sólo a uno, se crean y se destruyen con él (coche y ruedas).
- Los objetos **parte no** pueden **compartirse** entre varios objetos **todo**.

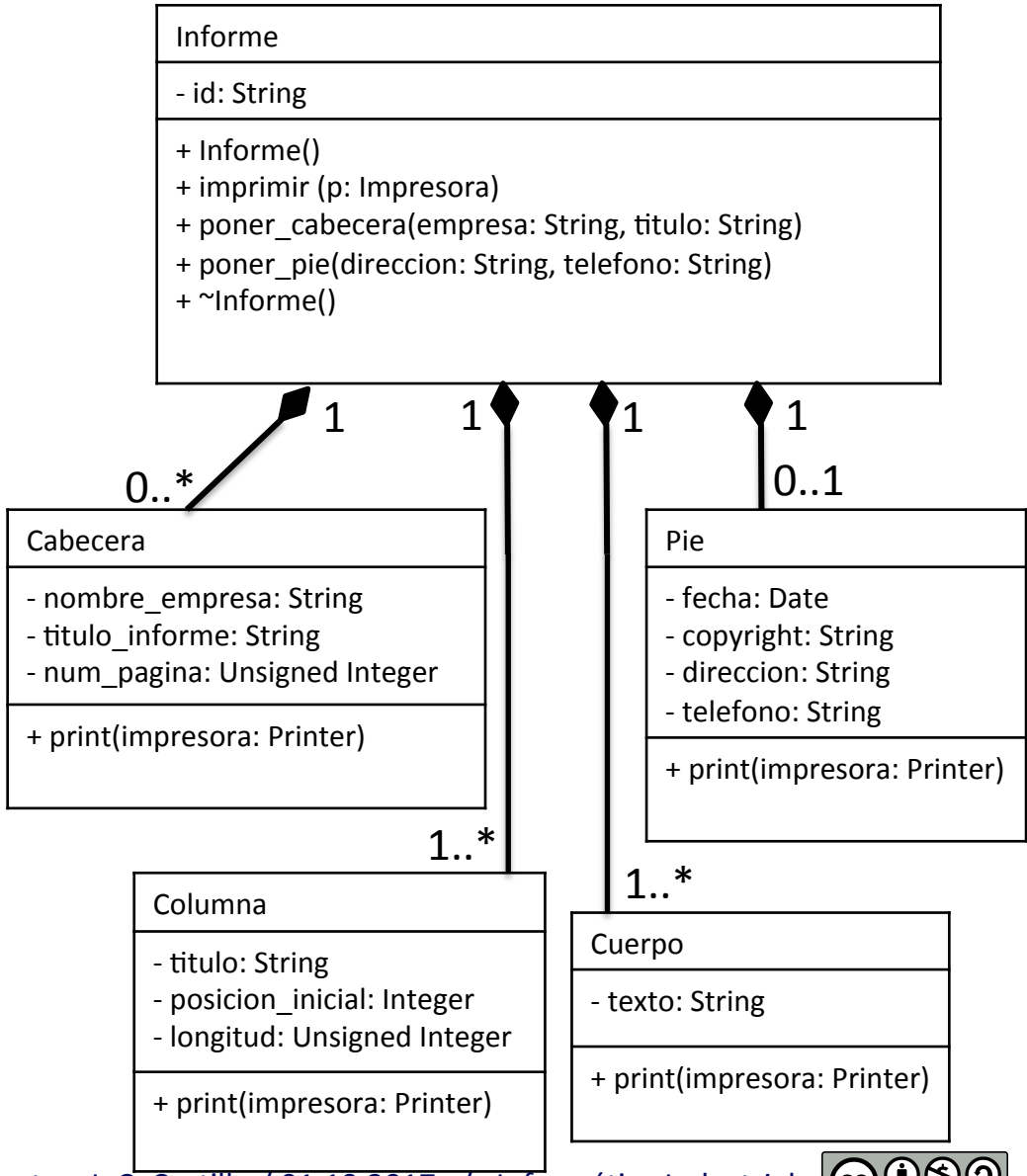
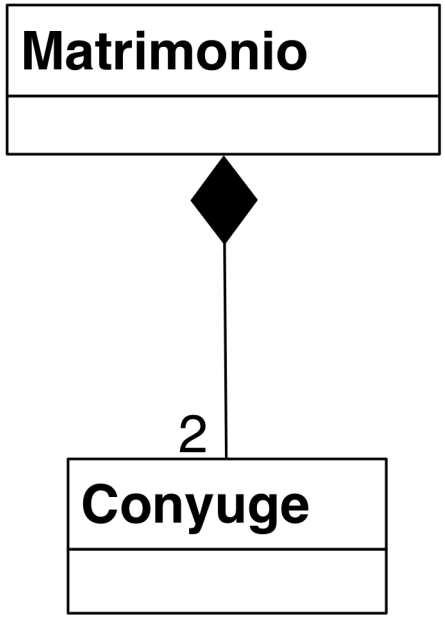


Diagrama de clases

Ejemplo de relación de composición

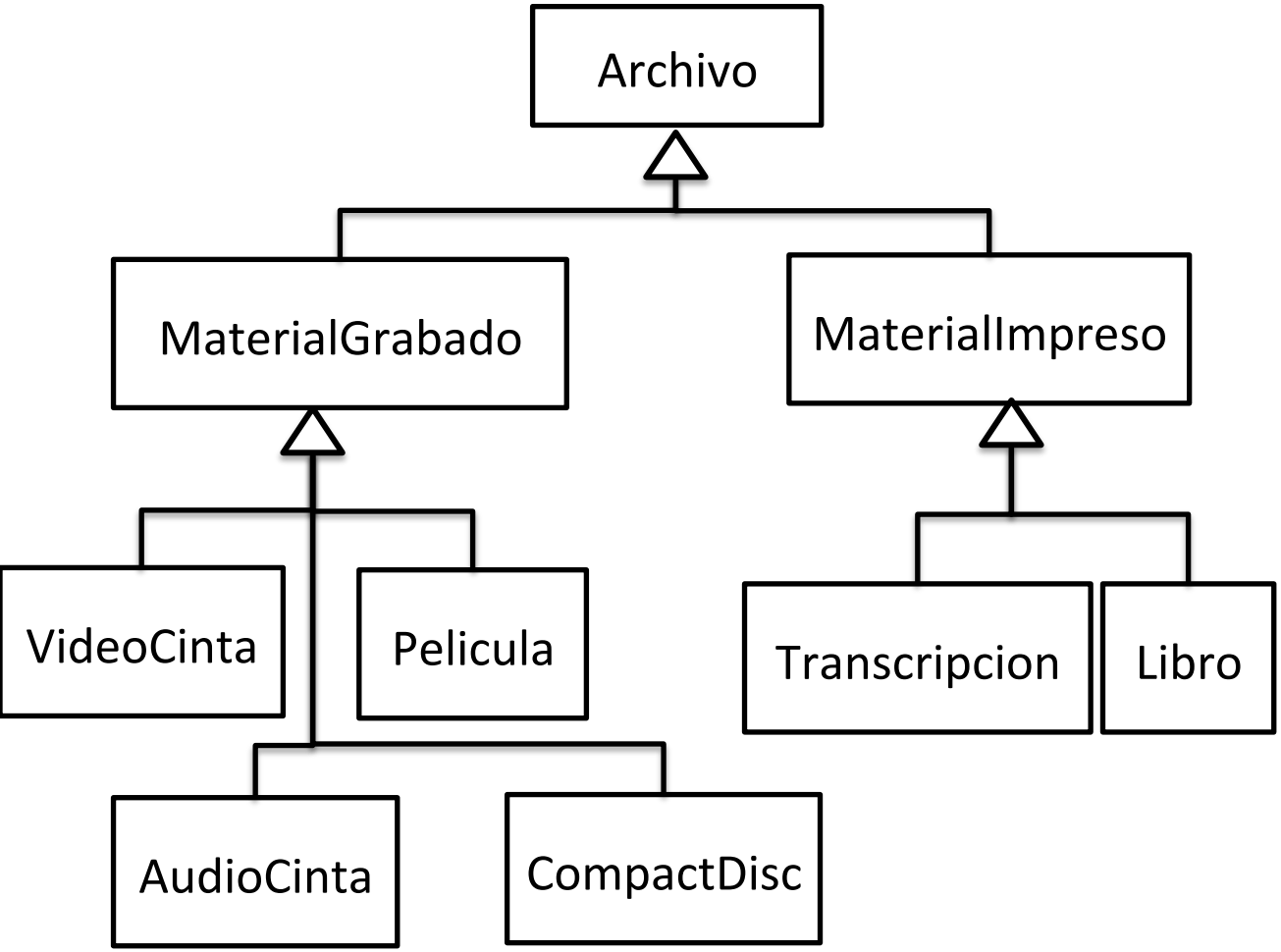



```
class Conyuge{
    …
}

class Matrimonio{
    …
    Conyuge persona1;
    Conyuge persona2;
    …
}
```

Diagrama de clases

Relación de Herencia



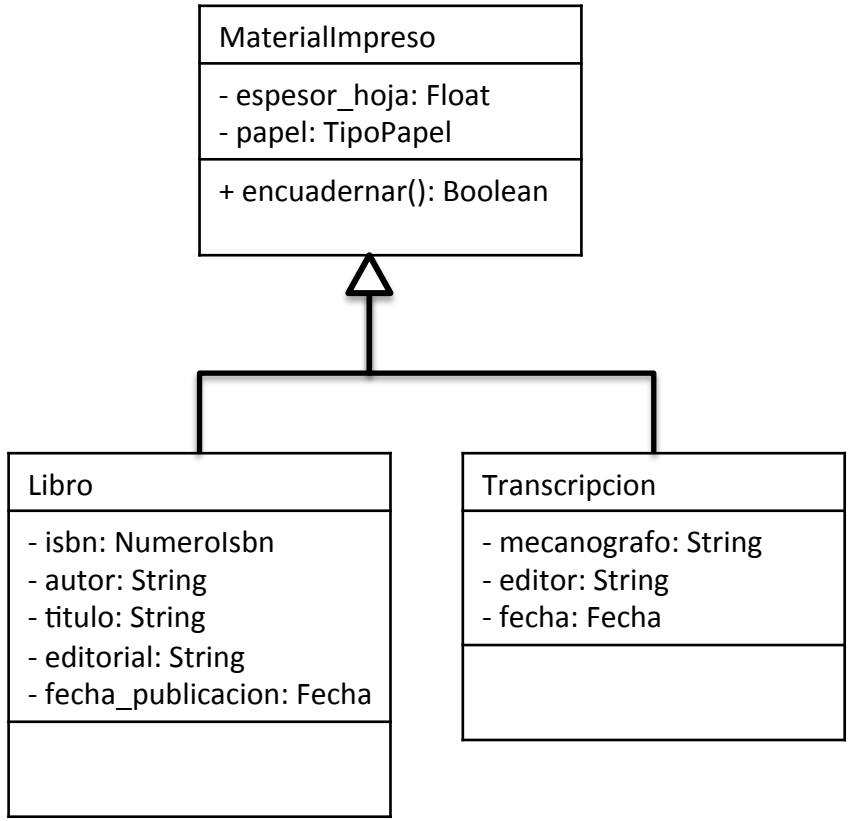


Diagrama de clases

Relación de Dependencia

- Muestra la dependencia entre una clase “cliente” y otra que ofrece un servicio usado por la primera
- Relación de uso
- Existe cuando los cambios en la clase independiente pueden afectar a la clase que depende

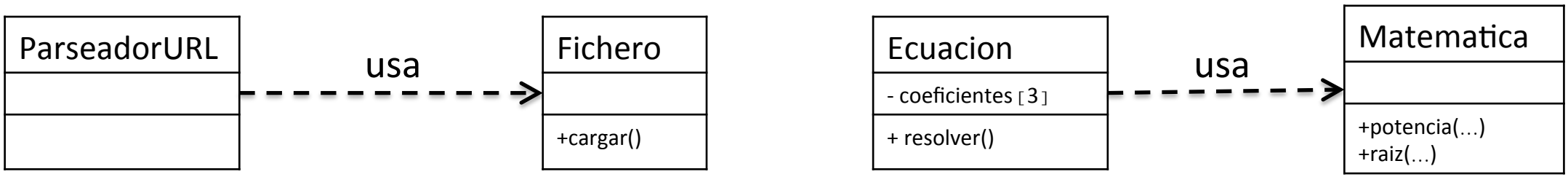


Diagrama de clases

Interfaces

- Declaración de los servicios que tienen que ofrecer las clases que los implementen.
- "Contrato" que permite a una clase realizar ciertas funcionalidades a través de métodos establecidos en el "contrato"
- Se relacionan con las clases que los implementan con una relación de **realización**

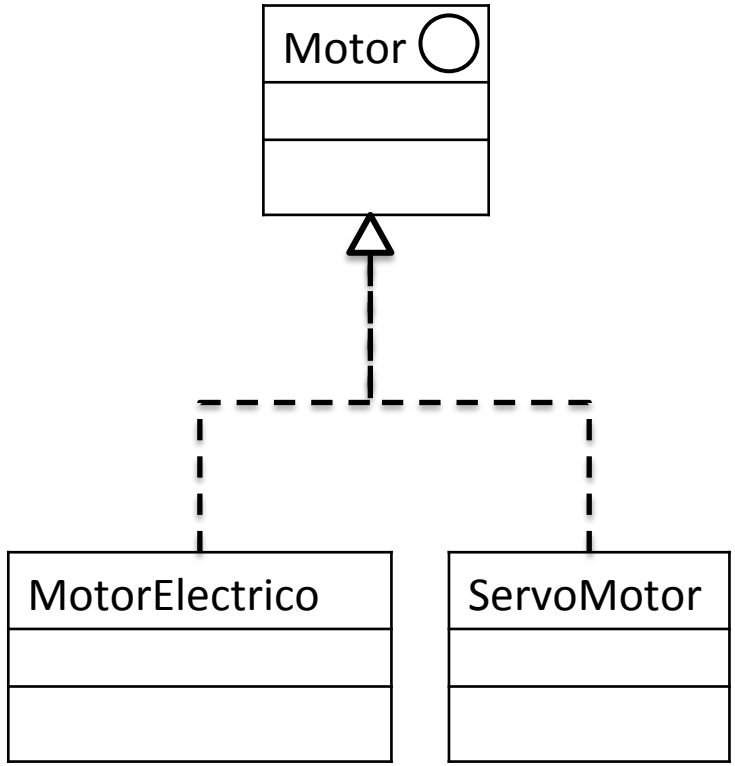
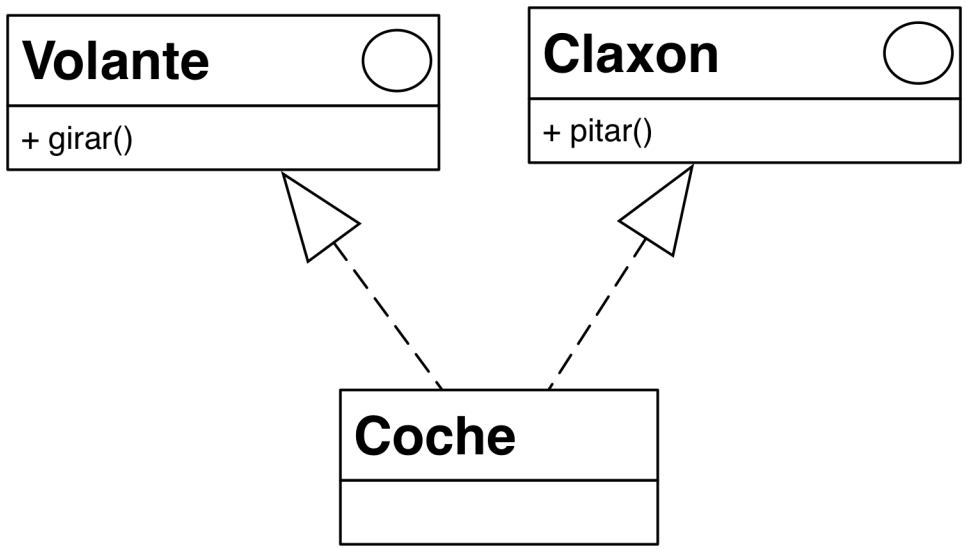
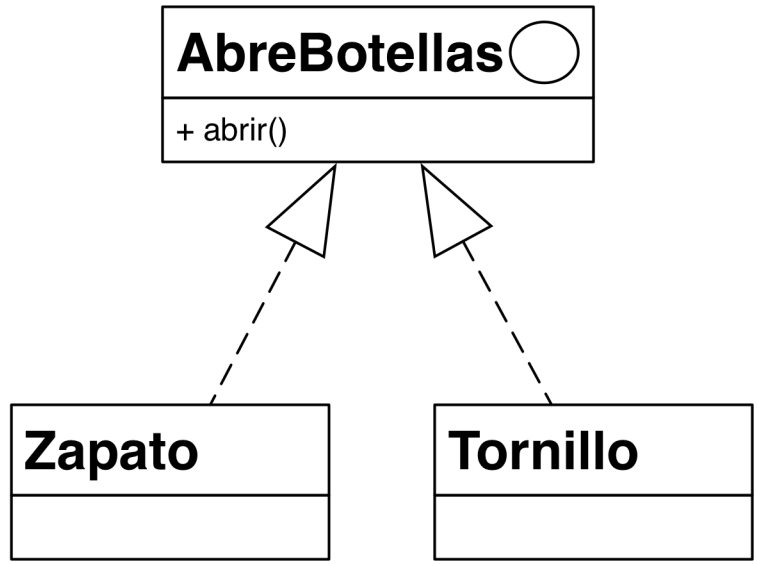


Diagrama de clases

Ejemplos de relación de realización

- Implementación de interfaces



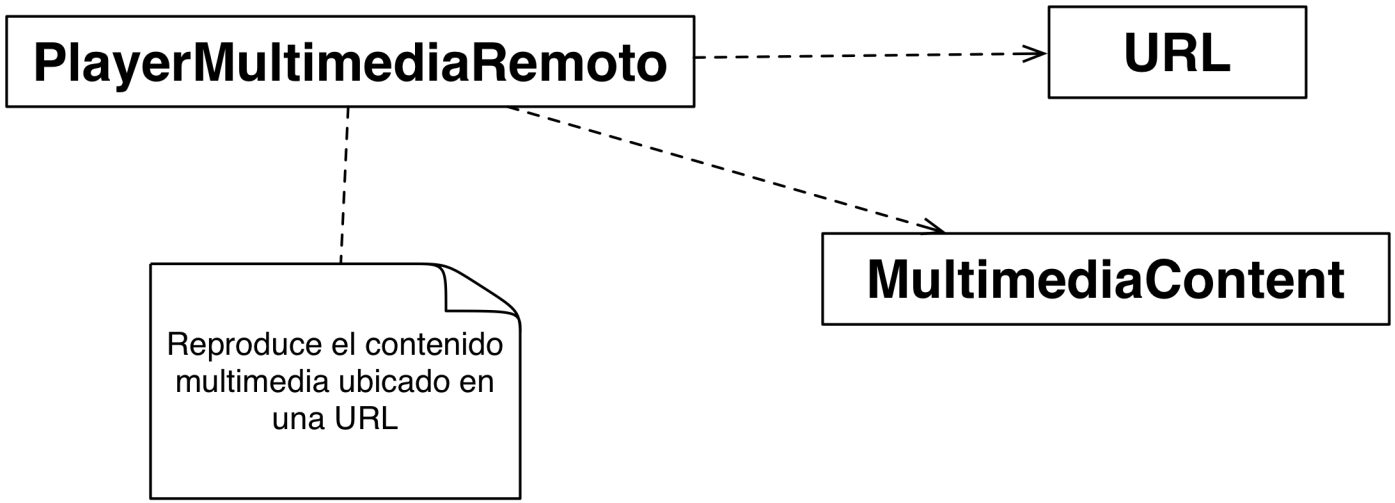


El Lenguaje Unificado de Modelado

- ¿Qué es UML?
- Diagrama de Clases
- Notas
- Diagrama de Secuencia
- Diagrama de Actividad
- Diagrama de Estados
- Diagrama de Casos de Uso

Notas

- Las notas pueden aparecer en cualquier diagrama

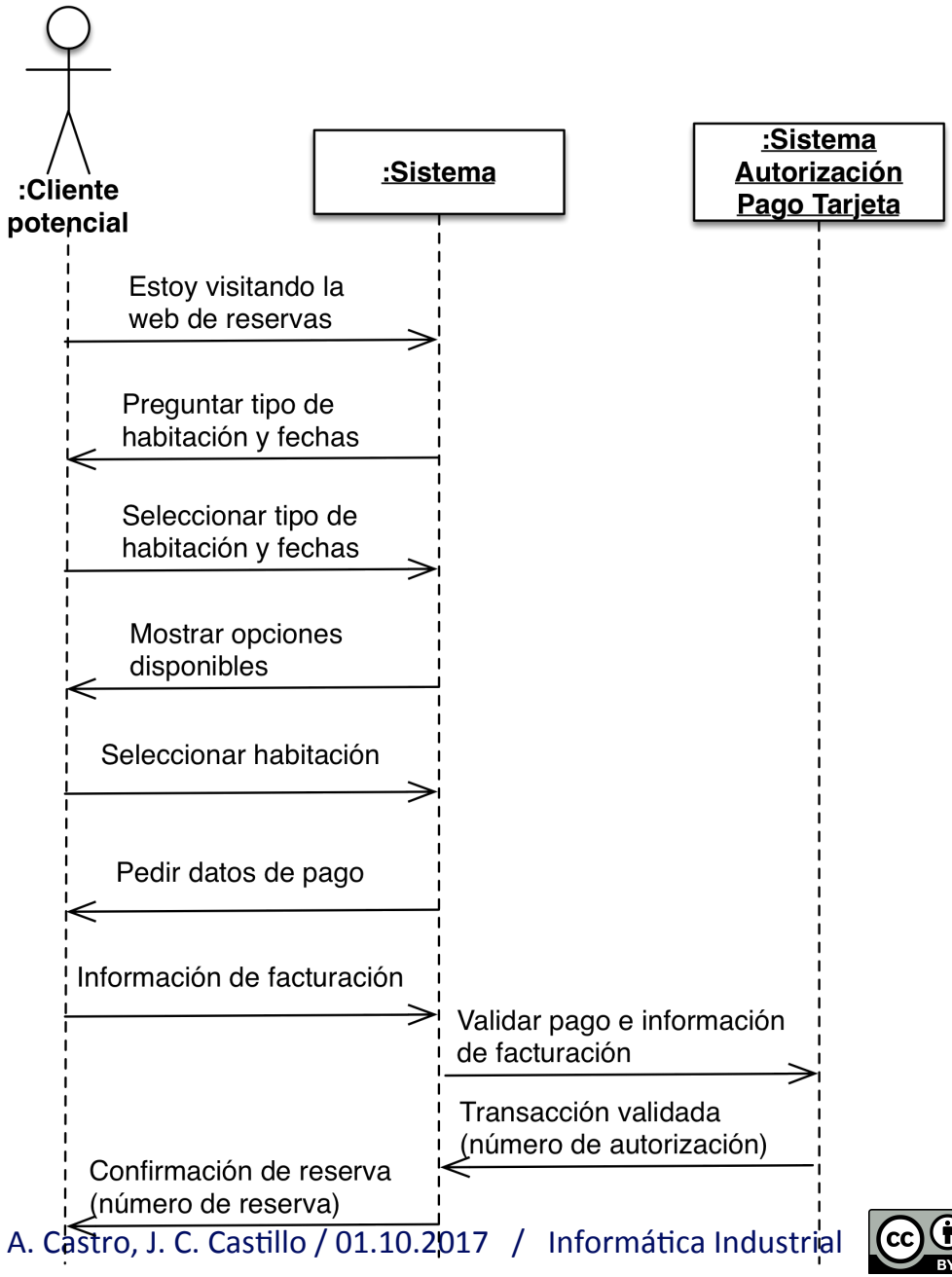


El Lenguaje Unificado de Modelado

- ¿Qué es UML?
- Diagrama de Clases
- Notas
- Diagrama de Secuencia
- Diagrama de Actividad
- Diagrama de Estados
- Diagrama de Casos de Uso

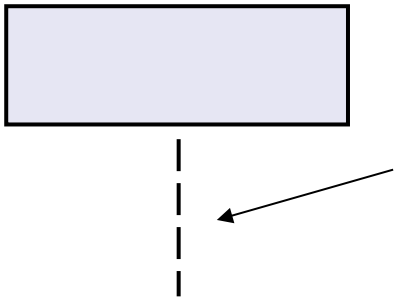
Diagrama de Secuencia

- Muestran como un grupo de objetos se intercambian mensajes a lo largo del tiempo y su orden.
- Los mensajes aparecen en orden cronológico desde la parte superior del diagrama

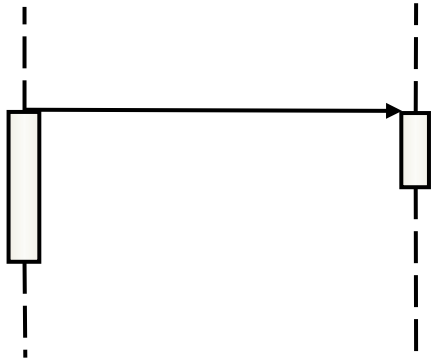


Diagramas de Secuencia

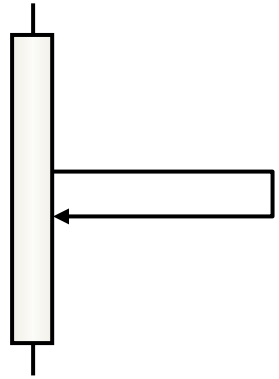
Objeto o actor



Mensaje de un objeto a otro



Mensaje a un mismo objeto



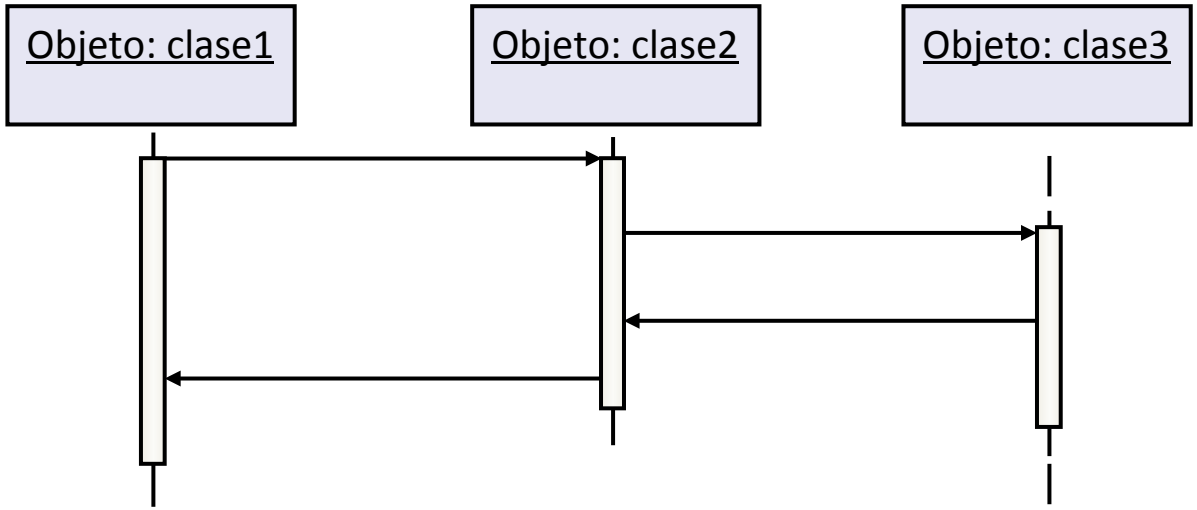


Diagrama de Secuencia

- Los mensajes pueden estar escritos en lenguaje natural o con una sintaxis más precisa y cercana a los lenguajes de programación dependiendo del nivel de abstracción

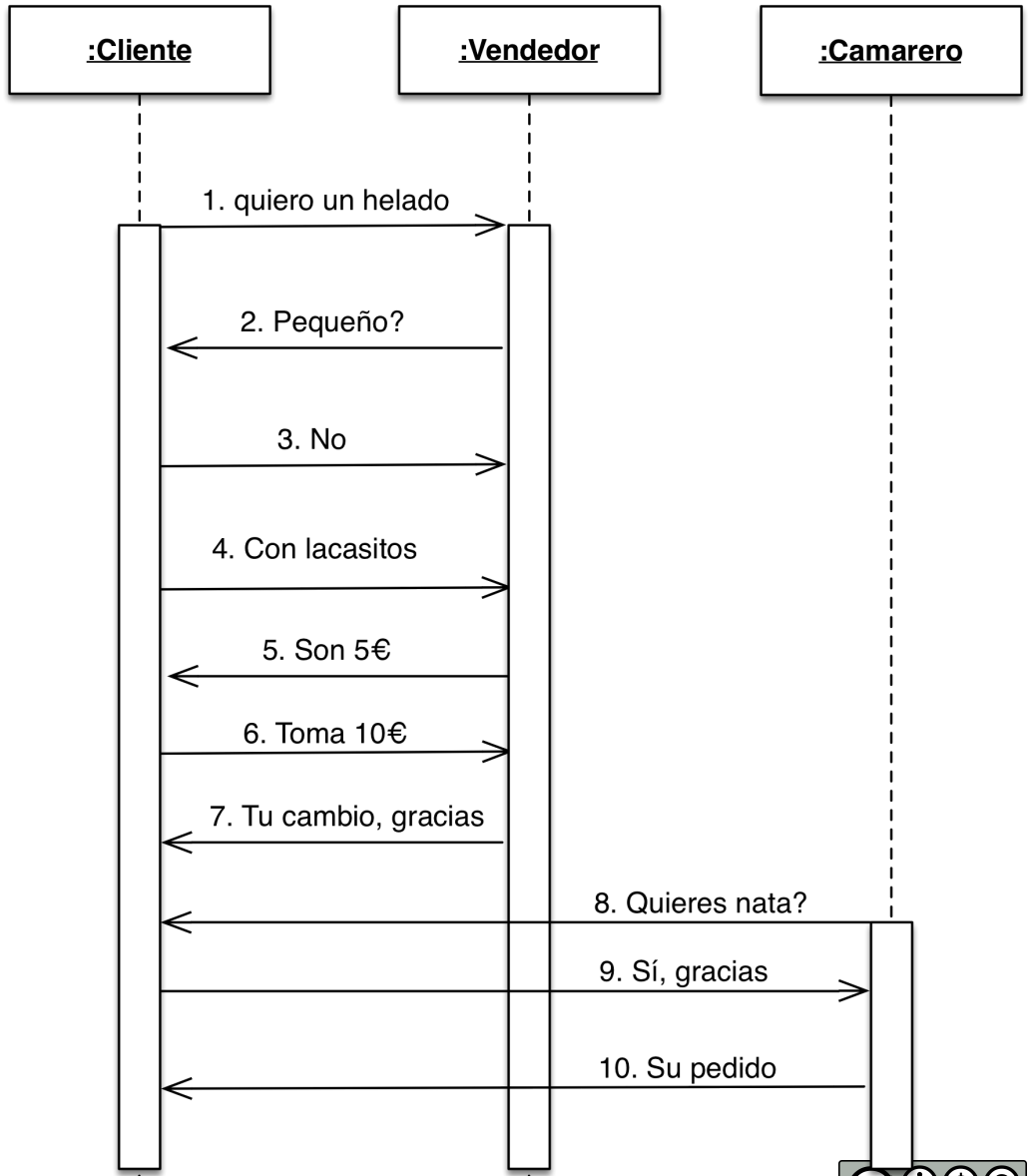


Diagrama de Secuencia

- Sintaxis mensaje

```
mensaje (params)
```

- Sintaxis parámetro

```
dirección nombreParametro: Tipo [Multiplicidad] = valorDefecto
```

  - Dirección: in, out, inout (por defecto in)
  - Para no especificar un parámetro se pone -

- Ejemplo

```
Transaction Results (status=OK, authCode)
```

Diagrama de Secuencia

- Tipos de mensajes
  - Mensajes en un diagrama alto nivel

  - Mensajes en diagramas de bajo nivel
    - Síncronas (p.e llamadas a métodos)

    - Asíncronas (p.e. sockets)

Diagrama de Secuencia

- Cuando se usan mensajes síncronos (llamadas a métodos) se puede poner explícitamente el retorno (return) o se puede omitir.

```
cout << “Introduce to nombre: ”;
String nombre;
cin << nombre;
Cout << “Te llamas “ << nombre;
```

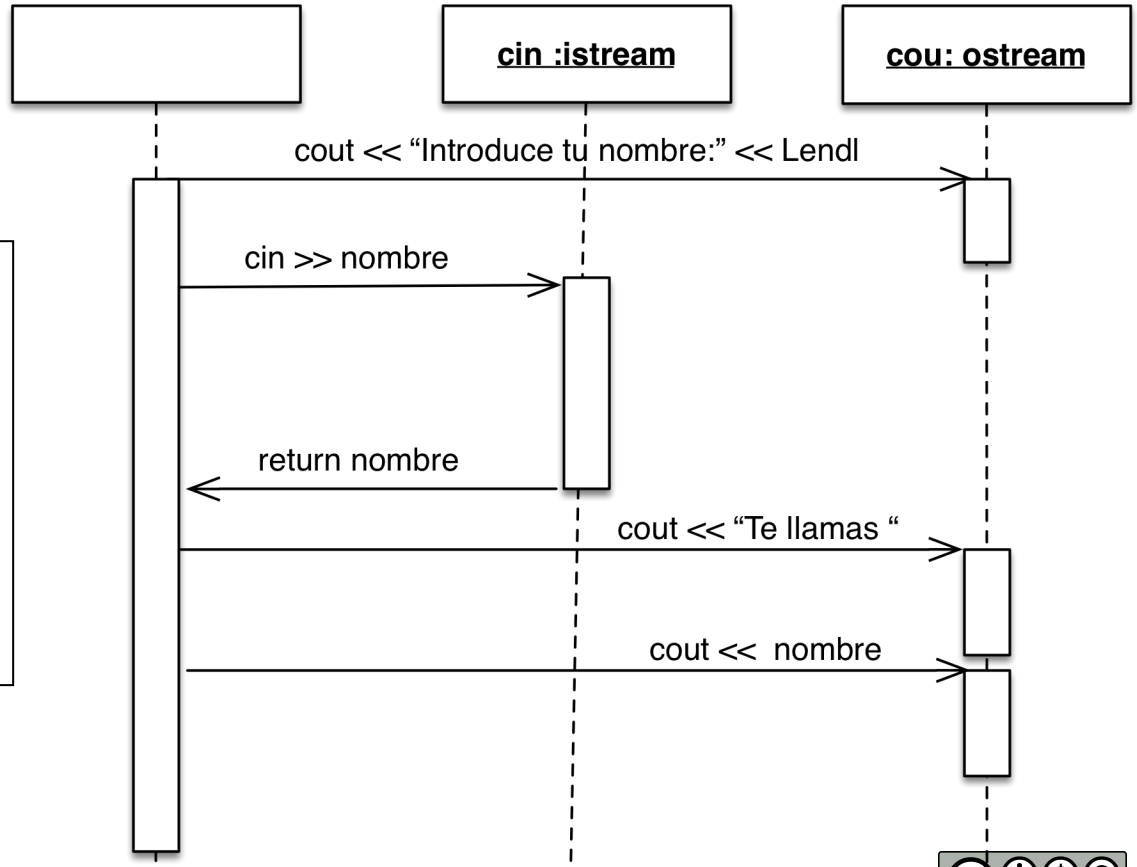


Diagrama de Secuencia

- Hay veces que los objetos no existen durante todo el tiempo
- Se usan mensajes de creación y de destrucción

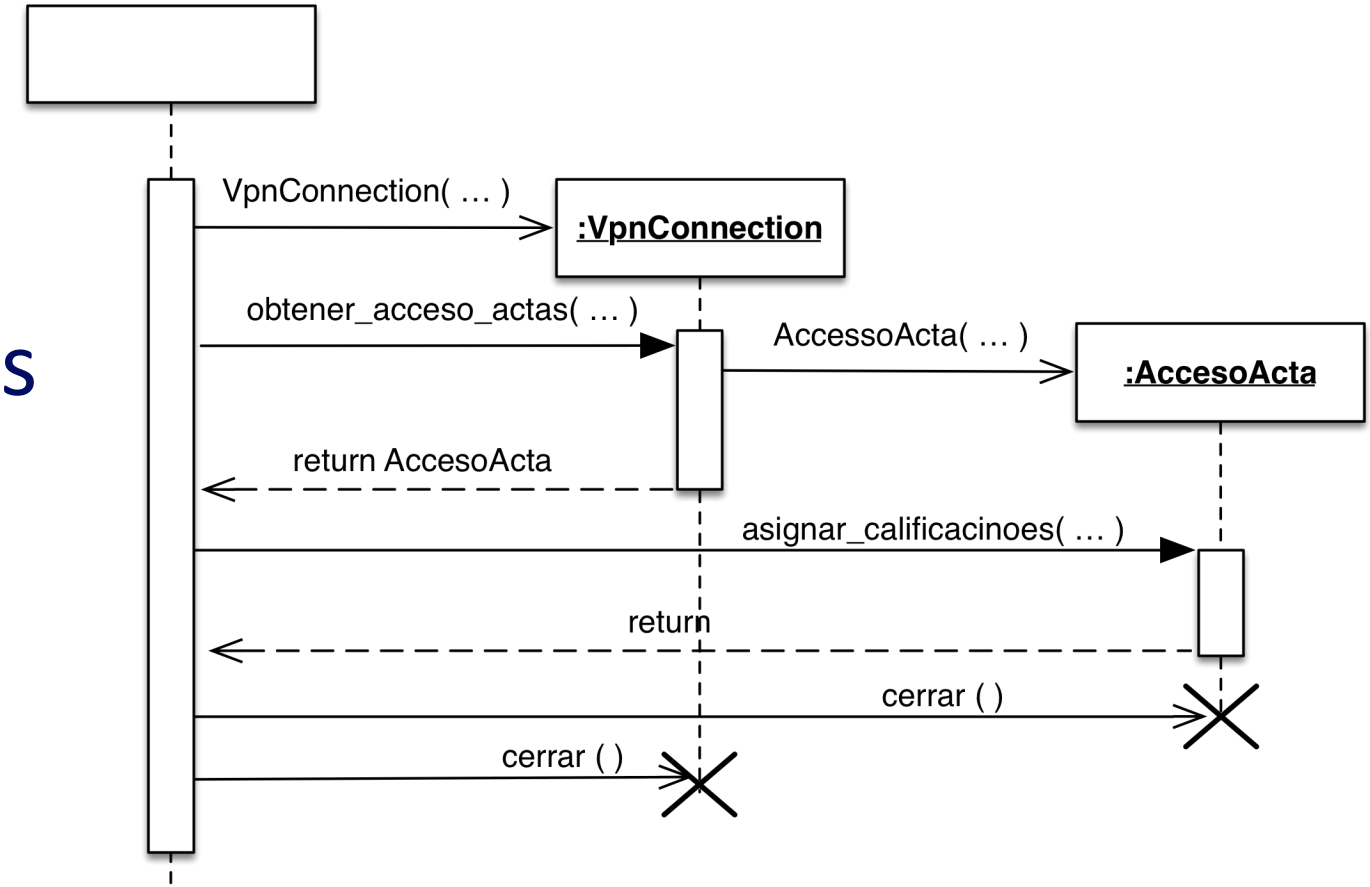


Diagrama de Secuencia

- Llamadas recursivas (autodelegación)
- Nuevo hilo de ejecución

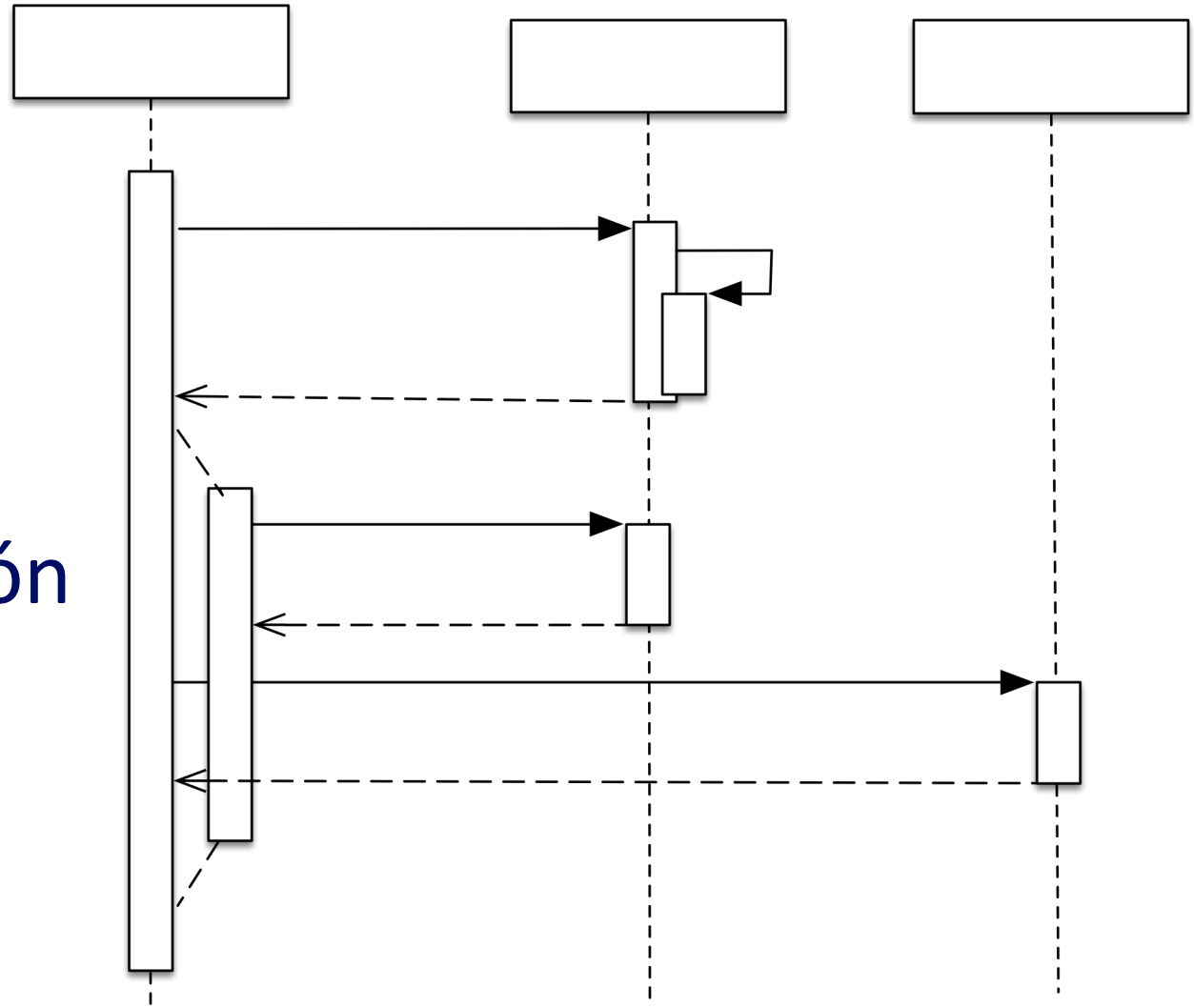


Diagrama de Secuencia

- Reutilización de diagramas de secuencia
- Funcionalidad similar a los métodos en programación
- Se pueden especificar parámetros
- Se puede especificar el valor devuelto o el nombre de una variable

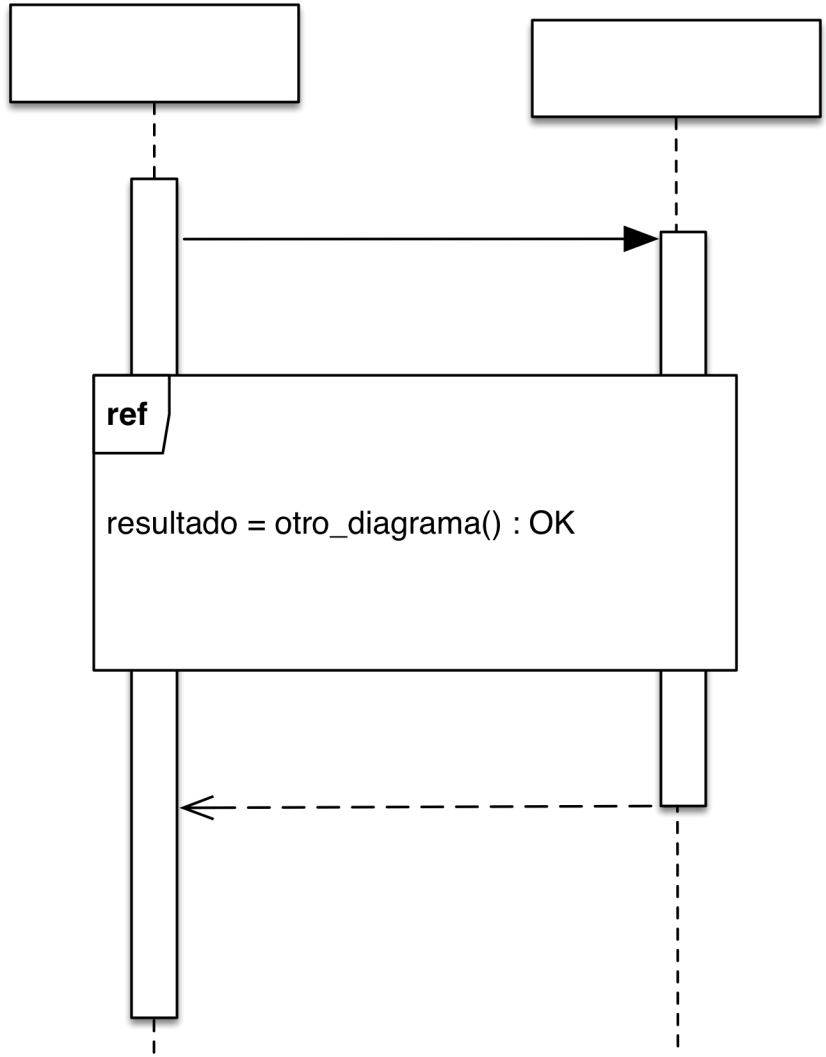
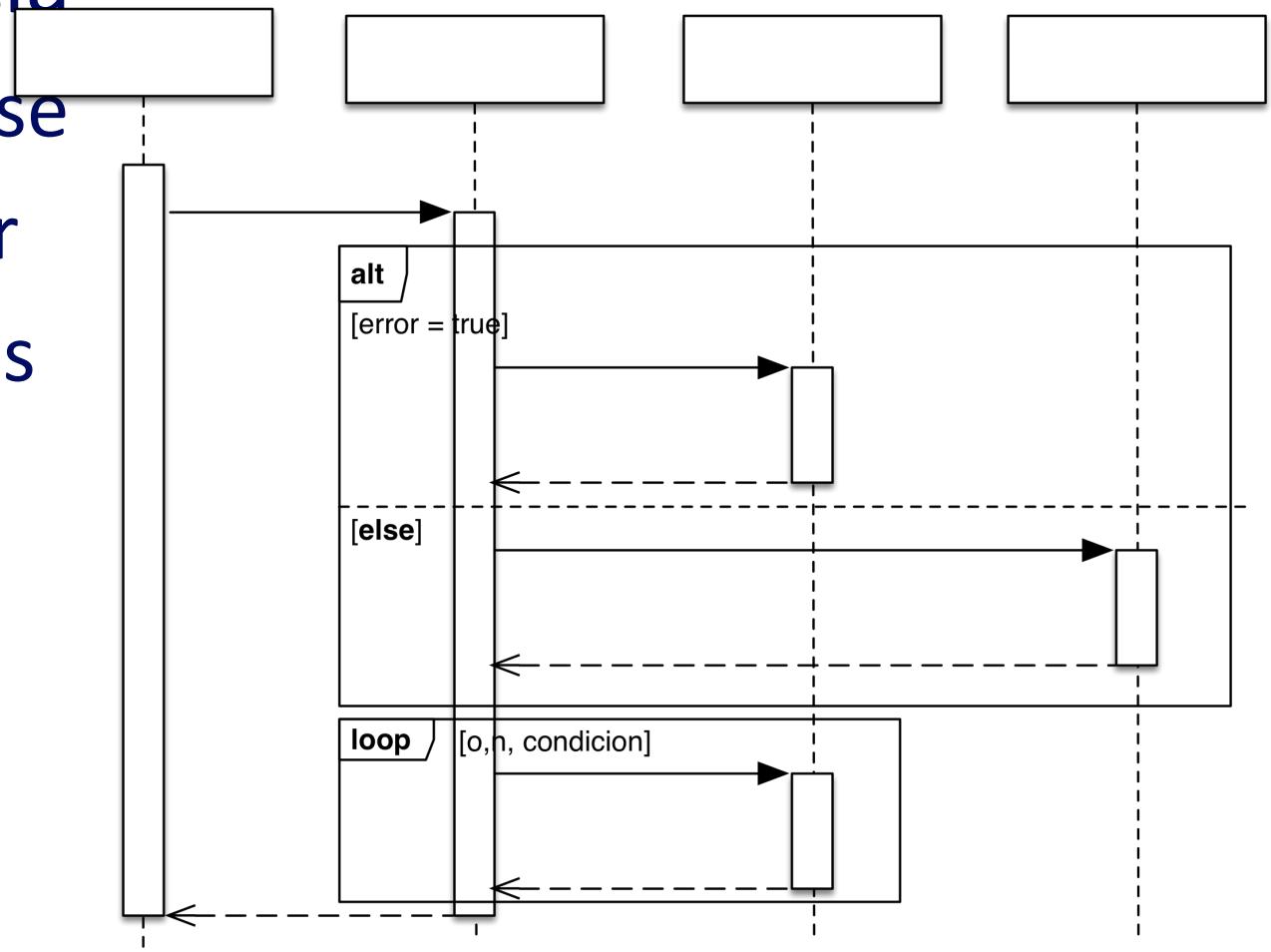


Diagrama de Secuencia

- Sentencias de control de flujo de ejecución en los diagramas de secuencia
  - alt: Equivale a un if/else
  - loop: Equivale a un for
  - Existen más sentencias



El Lenguaje Unificado de Modelado

- ¿Qué es UML?
- Diagrama de Clases
- Notas
- Diagrama de Secuencia
- Diagrama de Actividad
- Diagrama de Estados
- Diagrama de Casos de Uso

Diagrama de Actividad

- Similares a los flujogramas
- Se usan para mostrar el flujo de datos o el flujo de control
- Captura el flujo de trabajo de objetos que cooperan
- Puede representar el flujo de control en diferentes clases

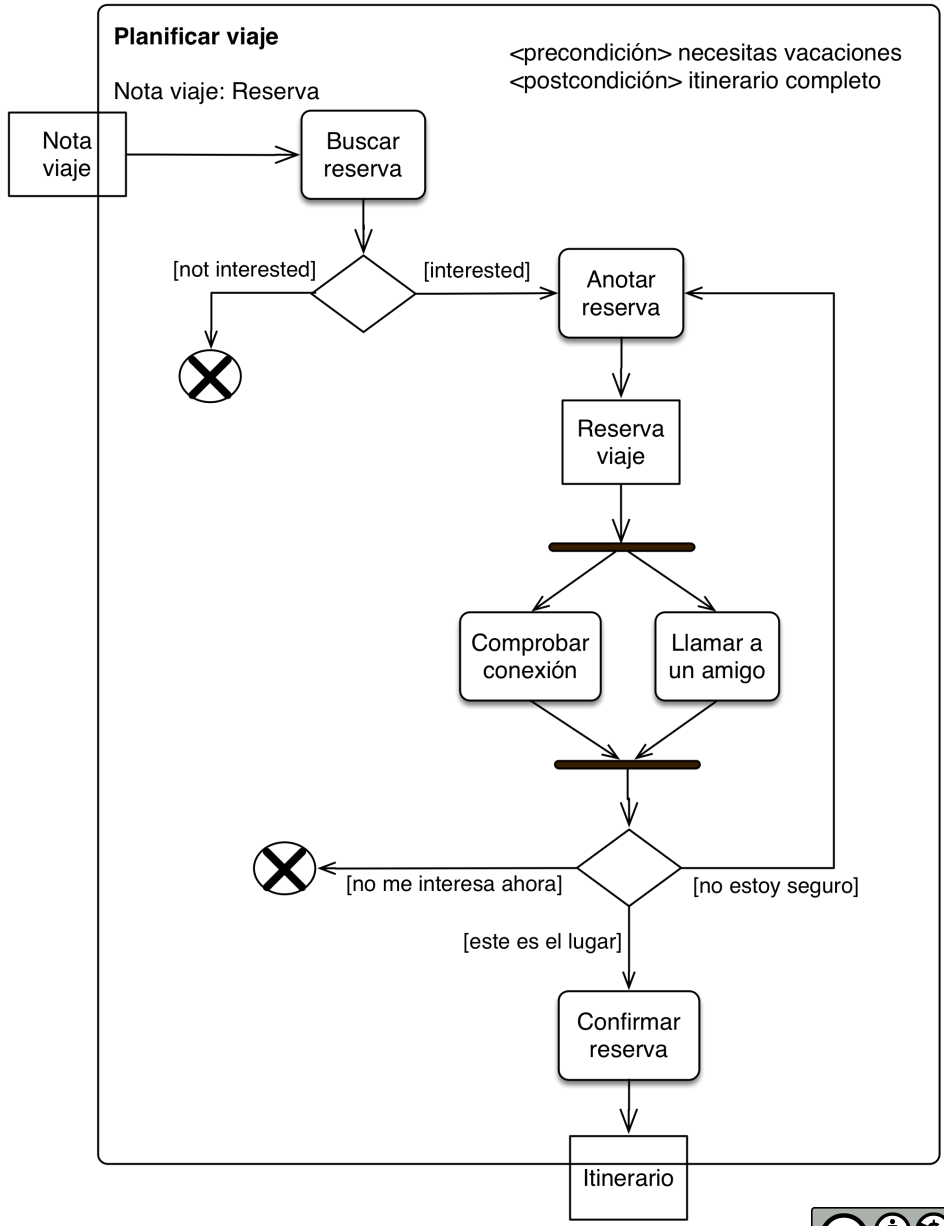
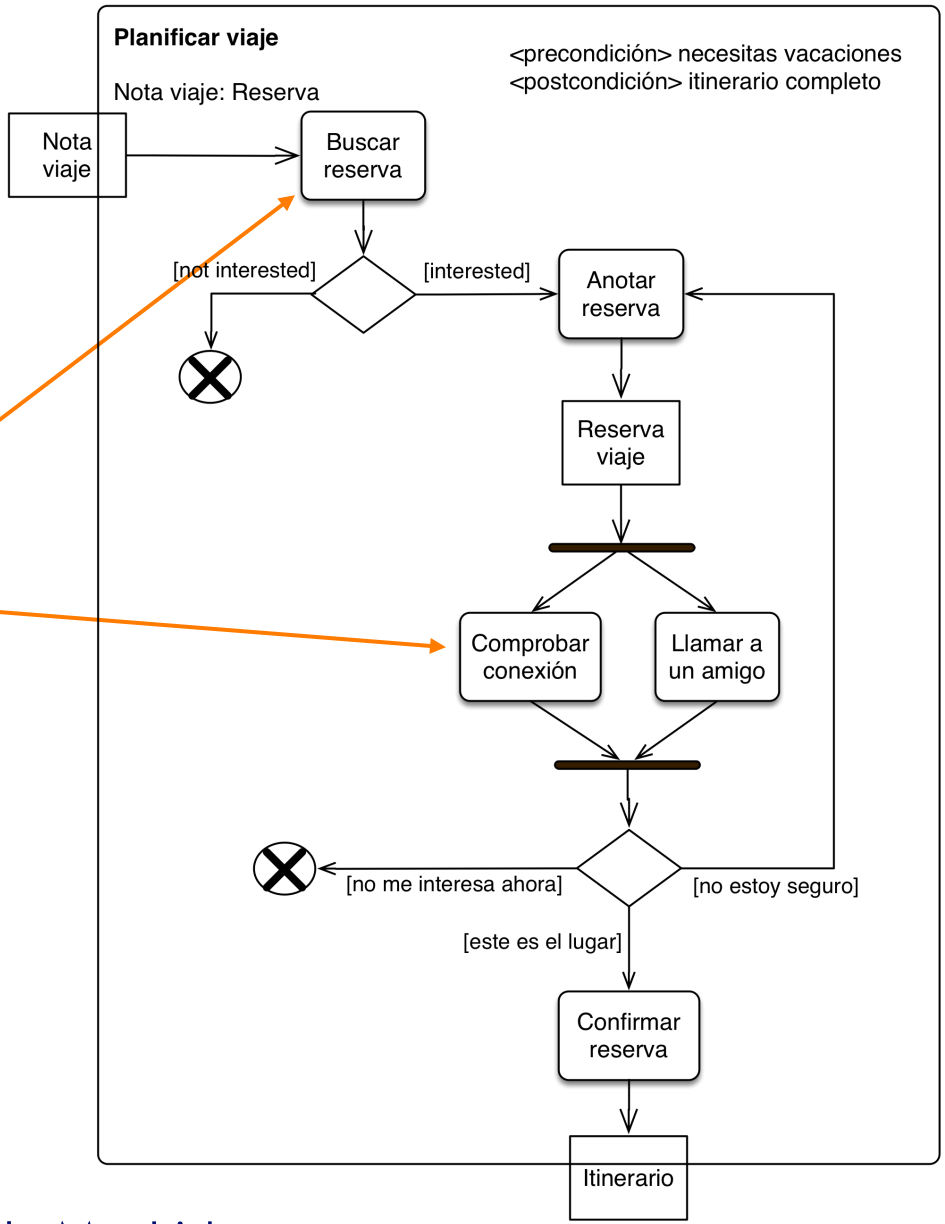


Diagrama de Actividad

- Acción / Estado
  - Obtener o establecer un valor de un atributo
  - Invocar la operación de una clase
  - Llamar a una función
  - Invocar una actividad que contiene acciones
  - Enviar una señal o notificación de un evento a un grupo de objetos
- Actividad
  - Agrupación de acciones, flujos de objetos y de control

Diagrama de Actividad



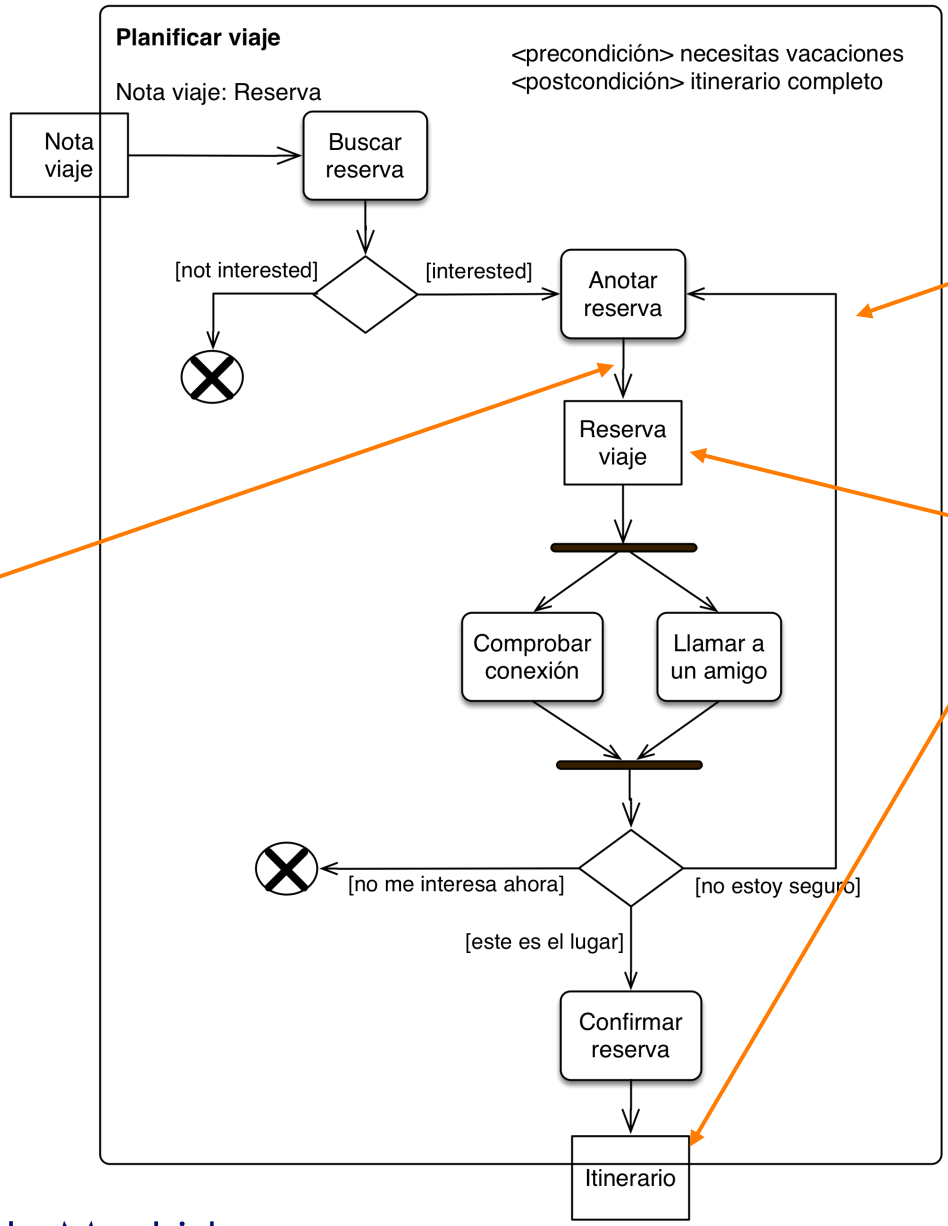
Acciones / Estados

actividad

Diagrama de Actividad

- Flujo de Control
  - Conecta las acciones por las que se ejecuta el flujo de control
- Nodo Objeto
  - Representa un objeto que sale de una acción y entra en otra acción
- Flujo de objeto
  - Los flujos que unen los objetos con las actividades

Diagrama de Actividad



Flujo de control

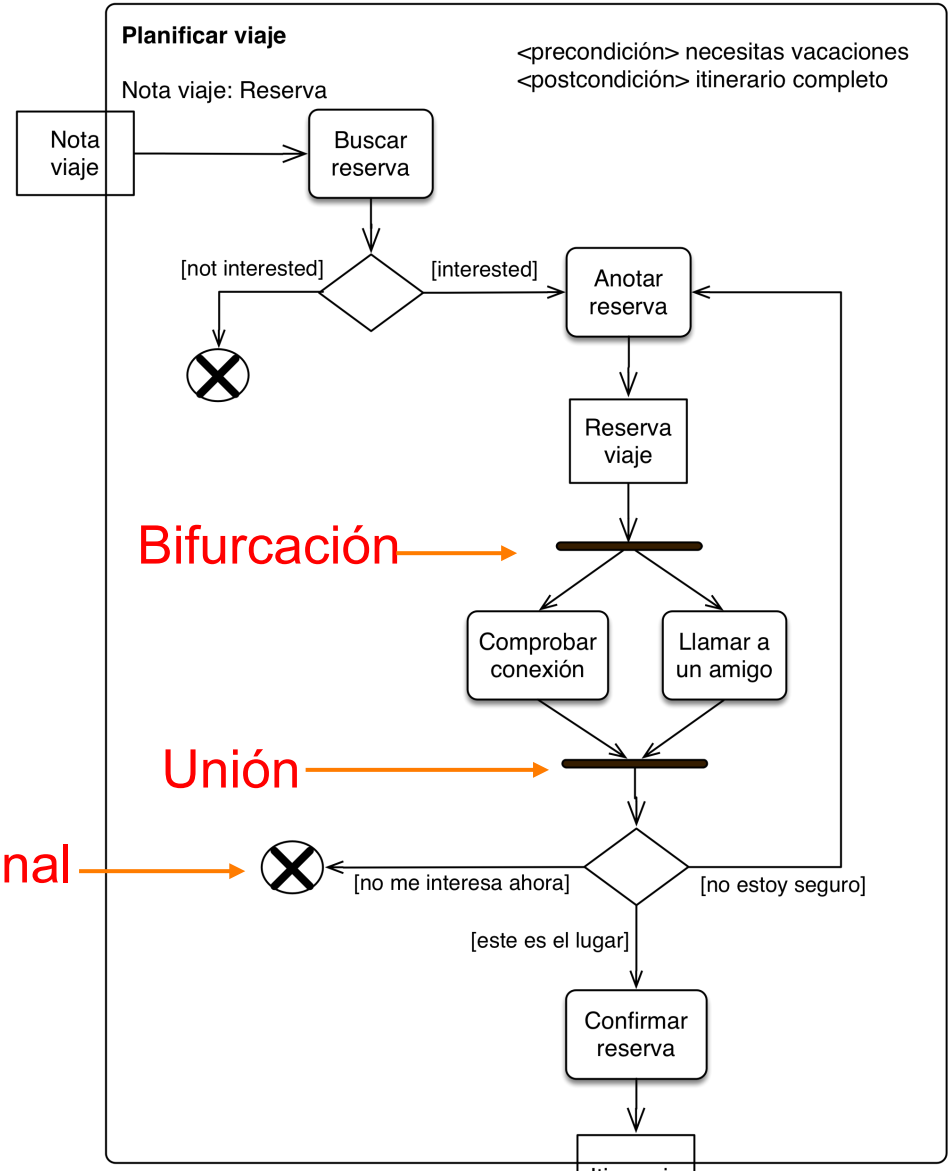
Flujo de Objeto

Nodos Objeto

Diagrama de Actividad

- Nodo de control
  - Nodos que guían el flujo de control y objetos
  - Inicial, Actividad final, Flujo final, Decisión, Fusión, Bifurcación, Unión, Conector
- Existen tipos de nodos más avanzados

Diagrama de Actividad



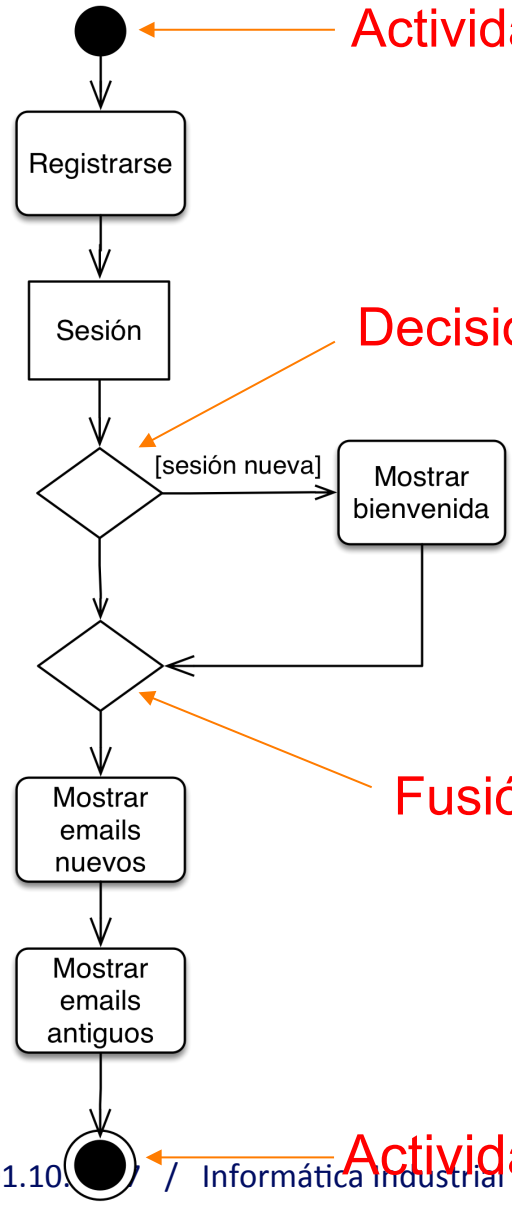
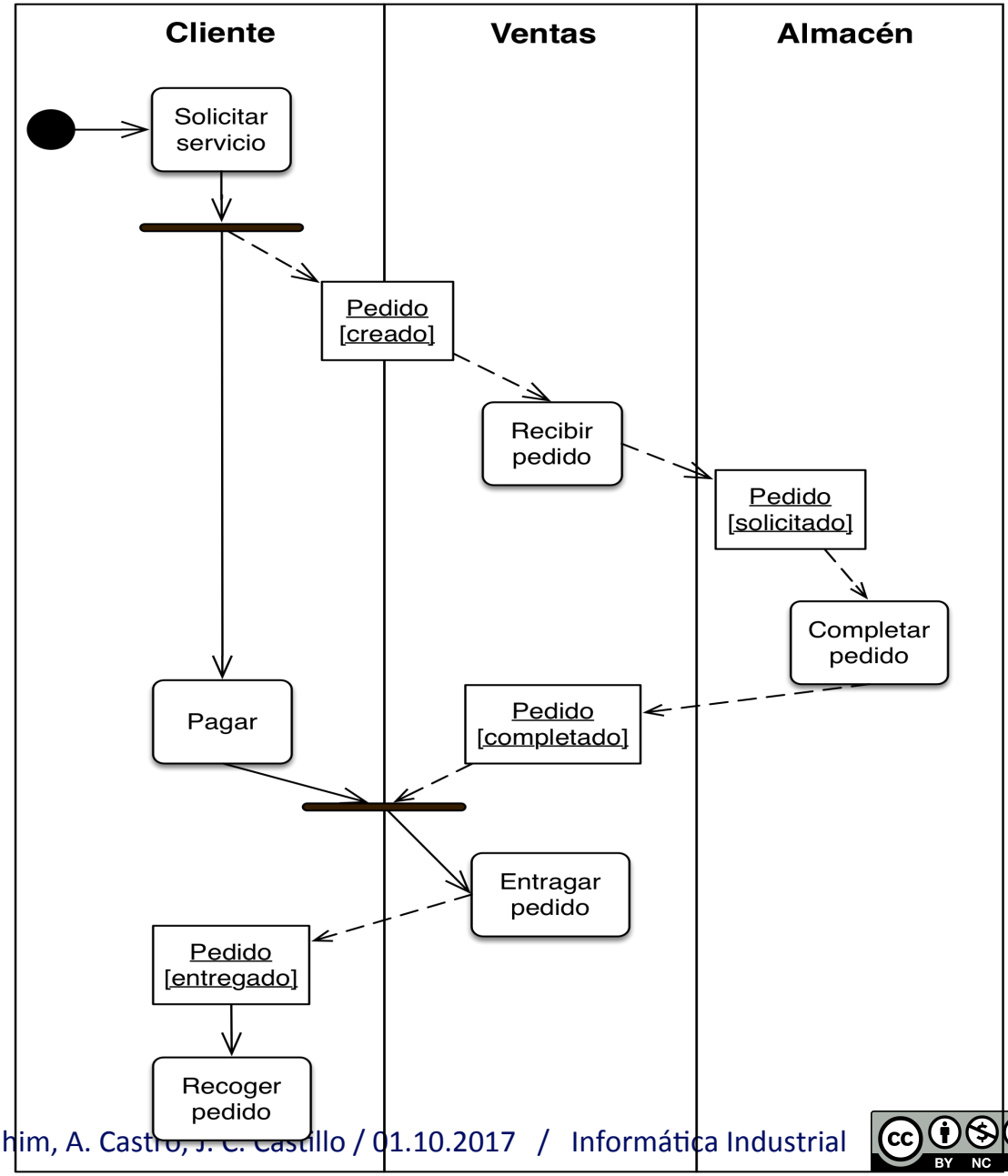


Diagrama de Actividad

- Para asignar la responsabilidad
  - Se pueden poner calles de piscinas (swim lanes)
  - Se puede poner el nombre entre paréntesis



(Cliente) Solicitar servicio

(Ventas) Entregar pedido

El Lenguaje Unificado de Modelado

- ¿Qué es UML?
- Diagrama de Clases
- Notas
- Diagrama de Secuencia
- Diagrama de Actividad
- Diagrama de Estados
- Diagrama de Casos de Uso

Diagrama de Estados

- Representa el ciclo de vida de un objeto en particular, los estados en los que se puede encontrar y como transita de uno a otro.
- Representan una máquina de estados
- Muestra los estados y las transiciones entre los estados dependiendo de los eventos que se producen

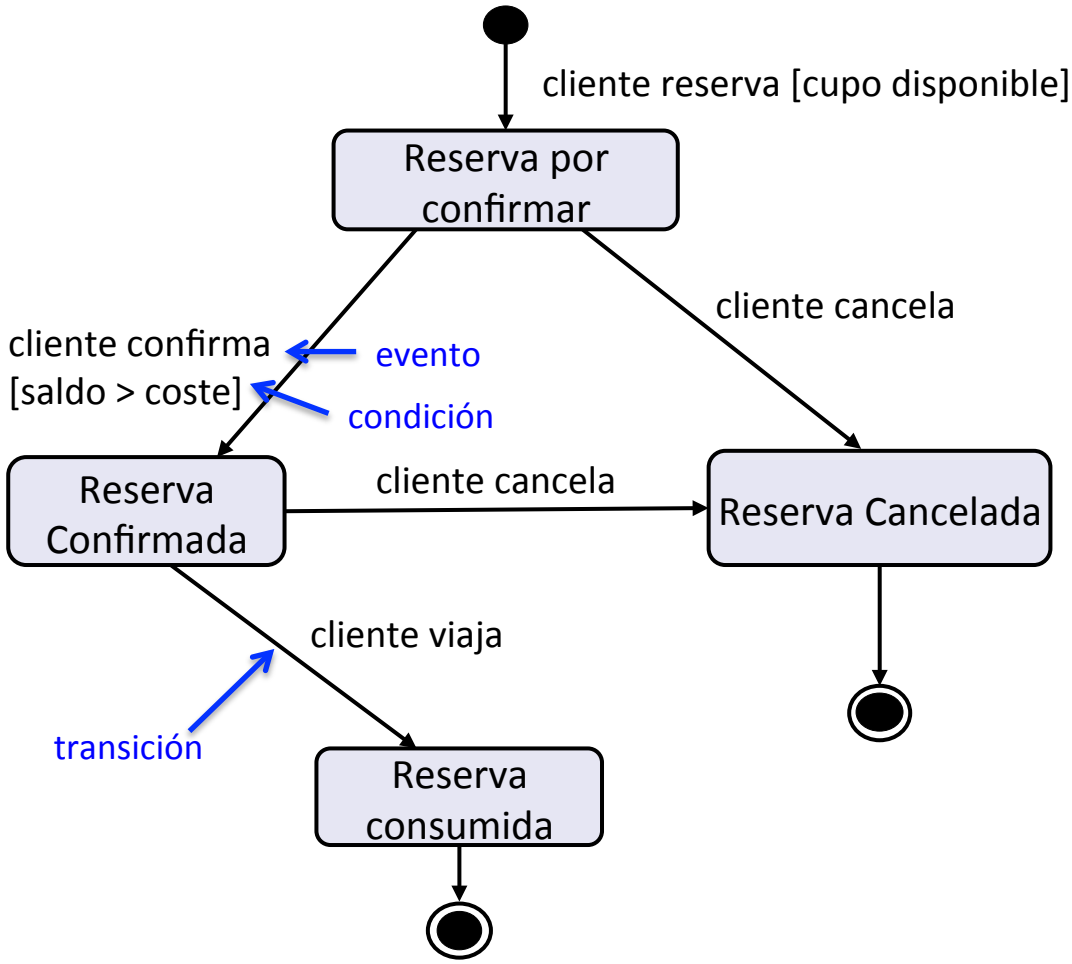
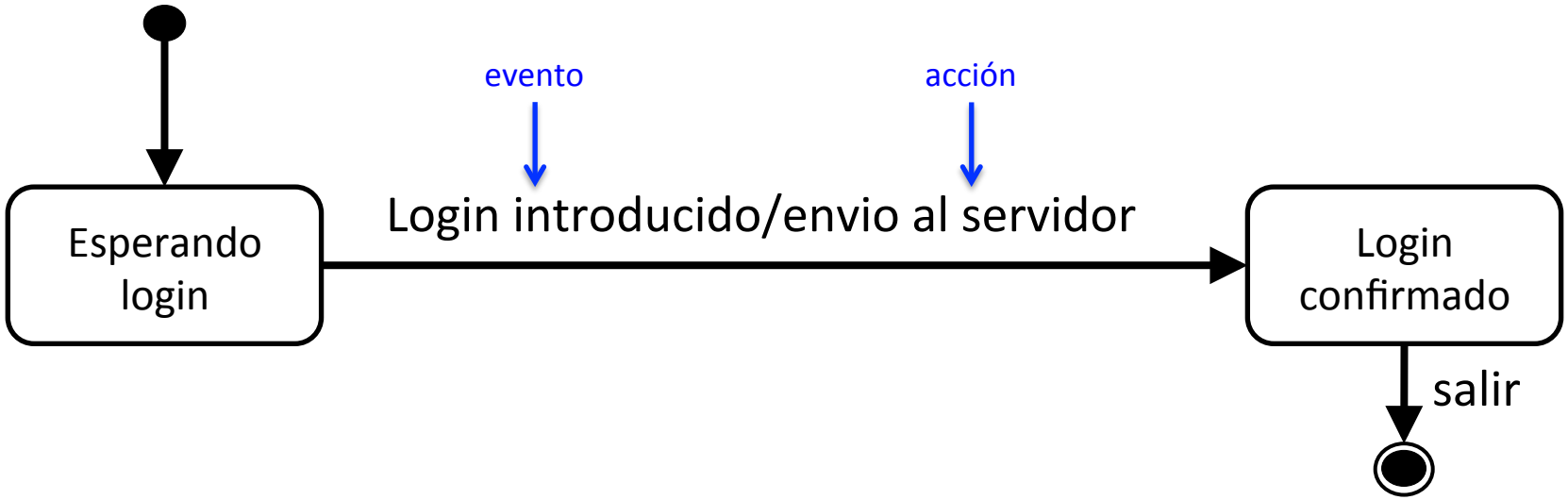


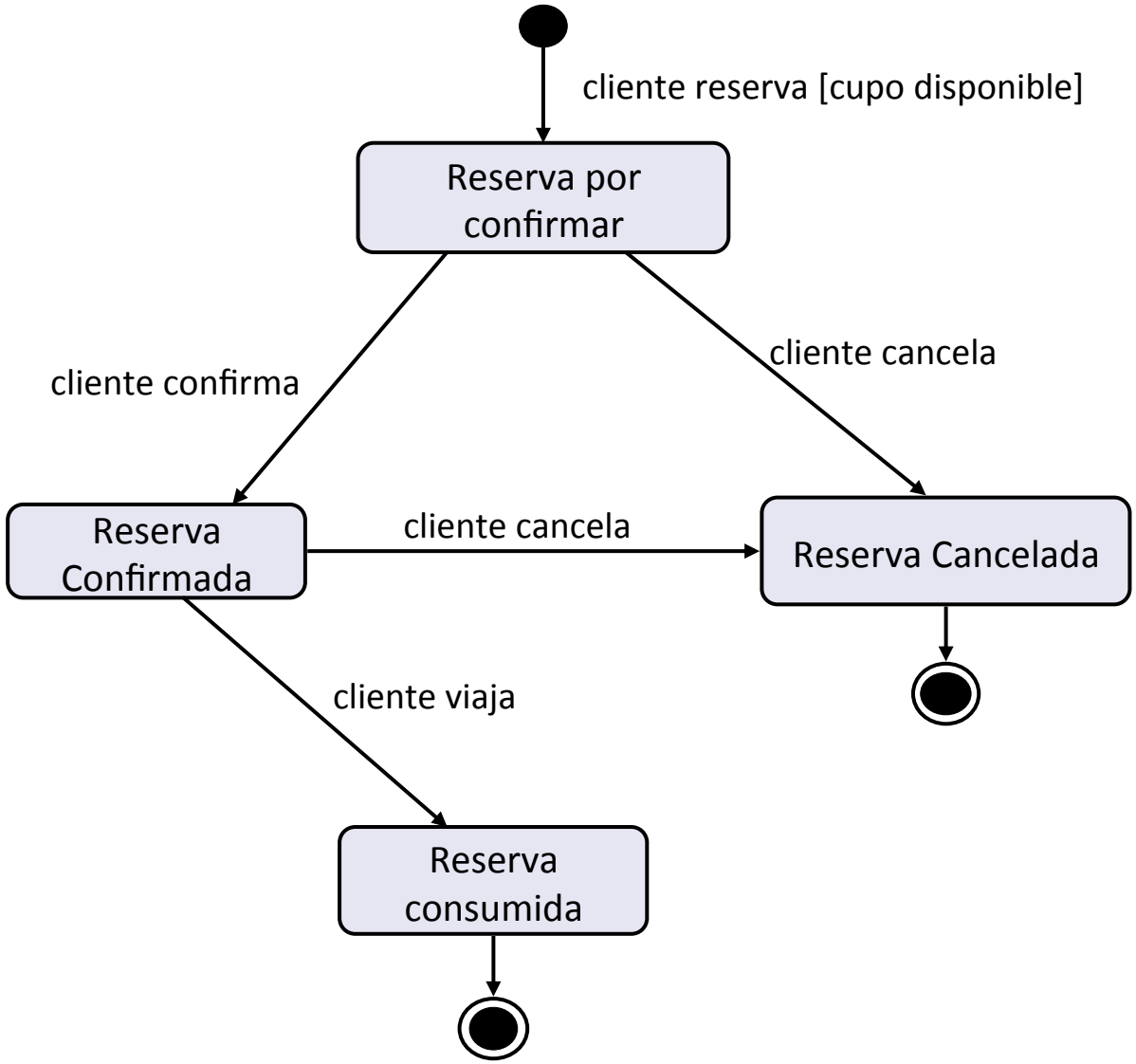
Diagrama de Estados

- Cuando llega un evento, se puede realizar una acción durante la transición a otro estado



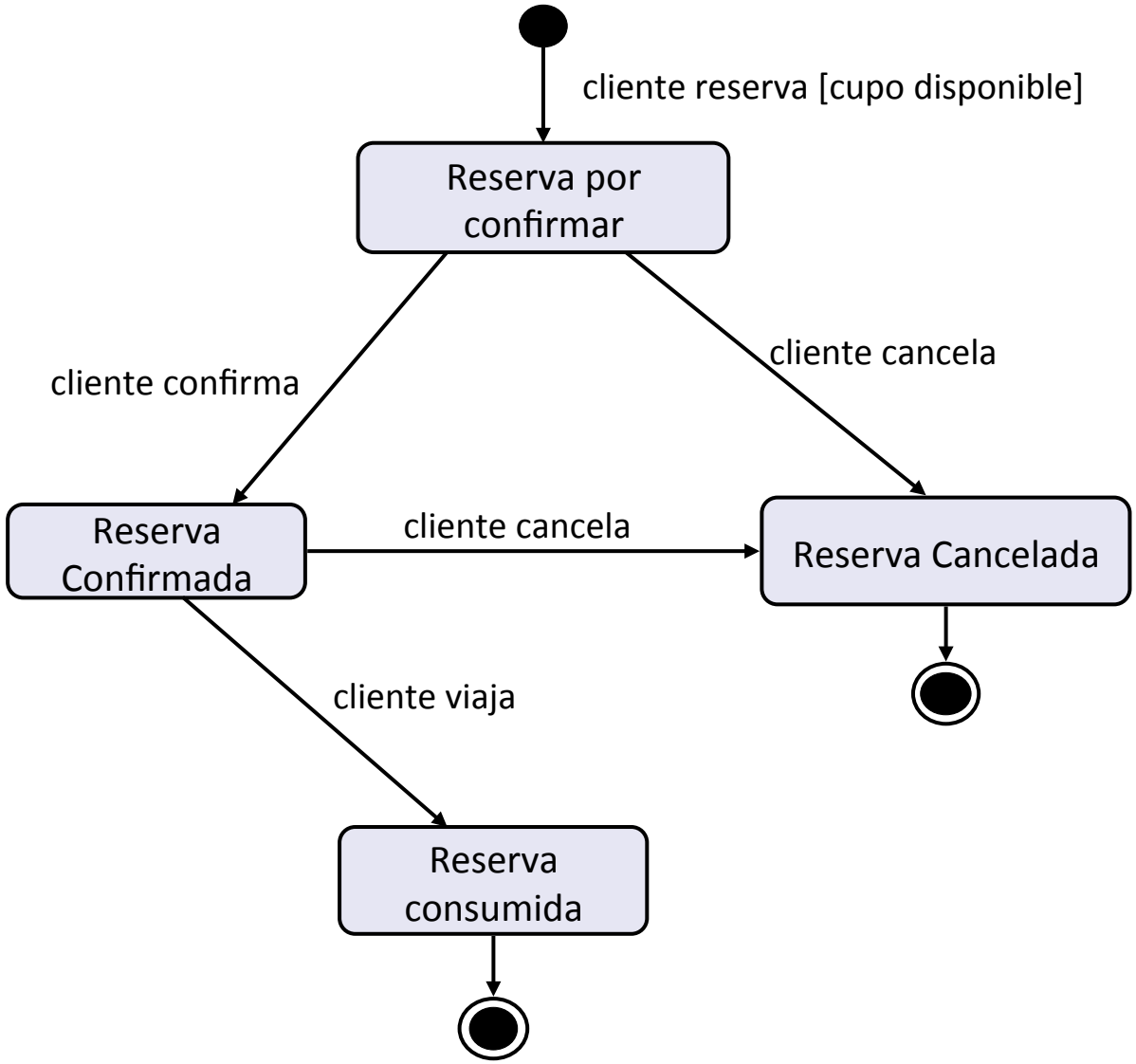
Diagramas de Estado

Ejemplo

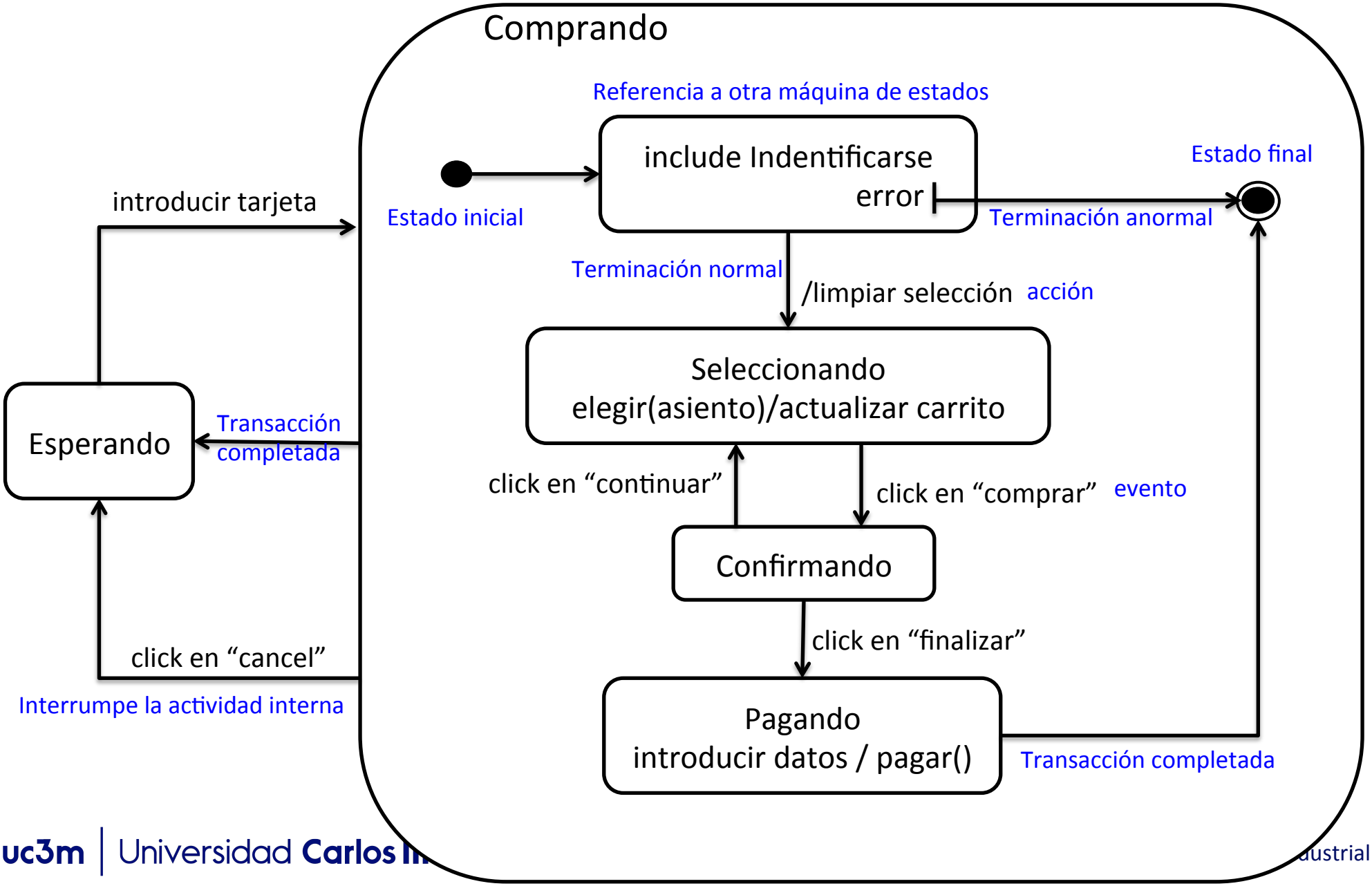


Diagramas de Estado

Ejemplo



Diagramas de Estado

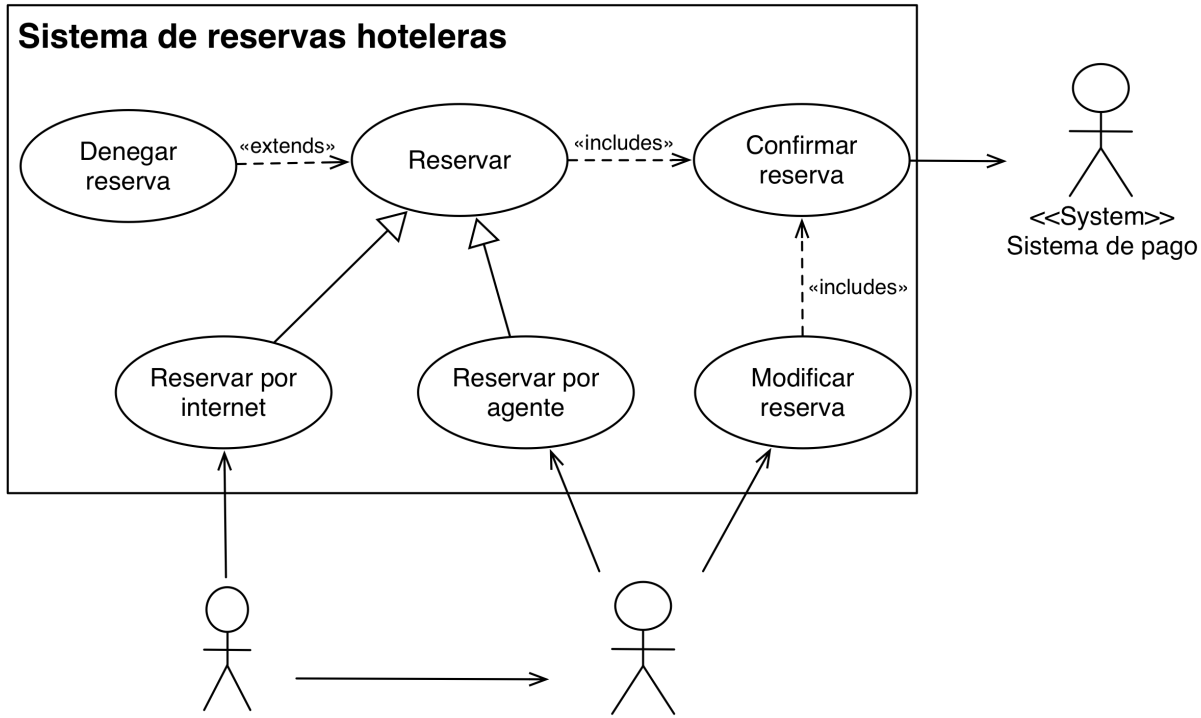


El Lenguaje Unificado de Modelado

- ¿Qué es UML?
- Diagrama de Clases
- Notas
- Diagrama de Secuencia
- Diagrama de Actividad
- Diagrama de Estados
- Diagrama de Casos de Uso

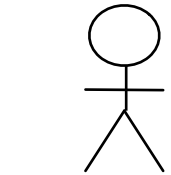
Diagrama de Casos de Uso

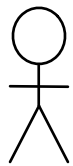
- Documentan el comportamiento de un sistema desde el punto de vista de los actores que participan.
- Los actores pueden ser usuarios u otros sistemas.
- Muestra las funciones que un sistema puede ejecutar o los servicios que ofrece
- Fácil interpretación

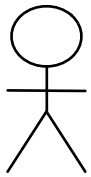


Elementos Básicos: Actores

- Usuarios del sistema: entidad externa que interactúa con el sistema
  - Humano, sistema informático, empresa…
- Independizar los actores de la forma en que se interactúa con el sistema
  - Un teclado no es un actor
- Un actor representa un rol que alguien puede estar jugando, no un individuo particular

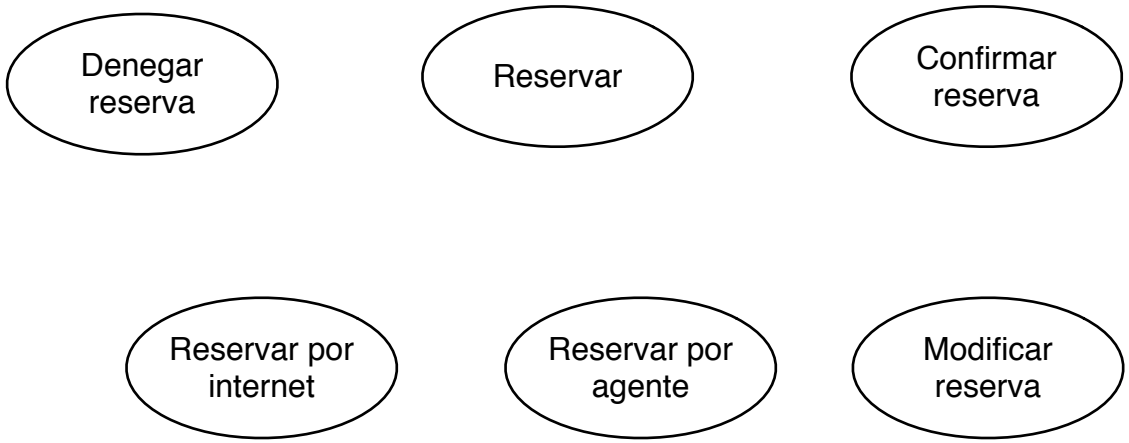



Cliente


Dependiente

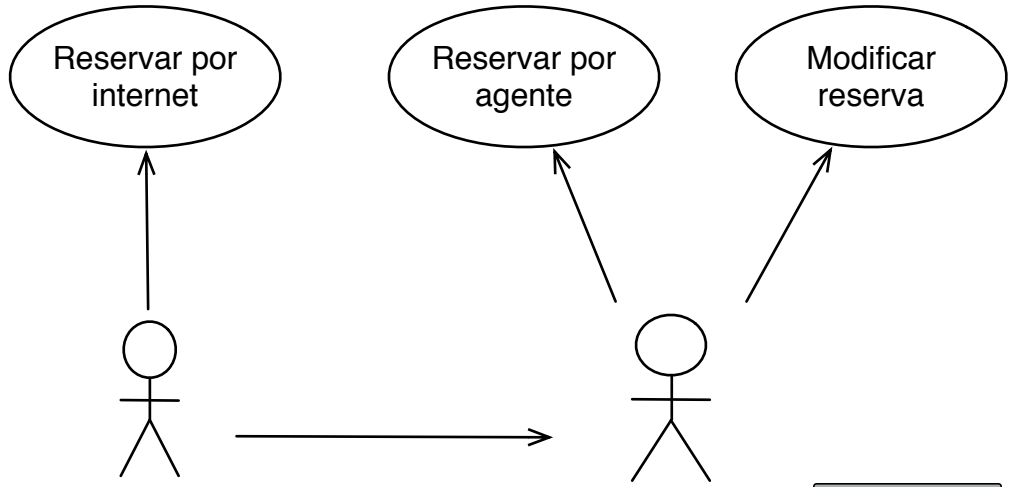
Elementos Básicos: Casos de Uso

- Es una tarea que debe poder llevarse a cabo con el apoyo del sistema que se está desarrollando.
- Representan algo destacable para los actores
- Se representan mediante un óvalo
- Cada caso de uso debe detallarse, habitualmente mediante una descripción textual



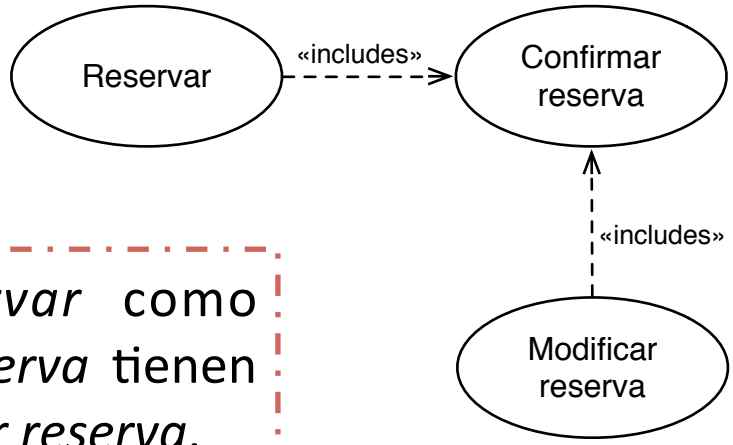
Elementos Básicos: Asociaciones

- Hay una asociación entre un actor y un caso de uso si el actor interactúa con el sistema para llevar a cabo el caso de uso.
- Relaciones son dependencias.
- Pueden existir relaciones entre un actor y un caso de uso, entre 2 actores o entre 2 casos de uso.
- Se representan con líneas y flechas (opcionales)
  - OJO: las flechas NO son el flujo de datos.
- Tipos de reutilización:
  - *include* , *extend* , generalización/especialización



Elementos Básicos: Asociaciones

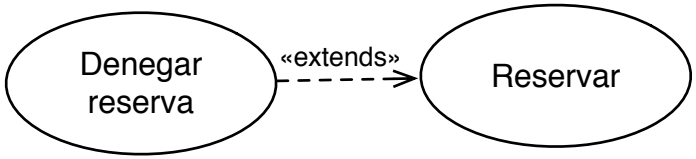
- **Include**: existe una relación de *include* entre dos casos de uso cuando el caso incluido es usado dentro del caso base.
- Equivalente a la llamada de un método.
- El caso de uso base está incompleto sin el caso de uso incluido.
- El caso de uso incluido puede ser común en dos o más casos de uso.



Tanto *Reservar* como *Modificar reserva* tienen que *Confirmar reserva*.

Elementos Básicos: Asociaciones

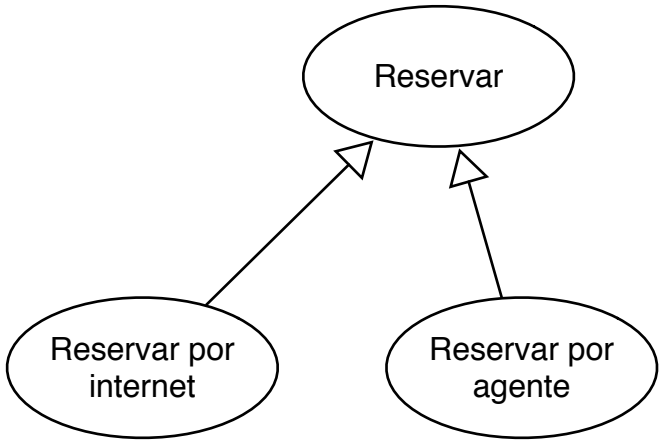
- **Extend**: se usa una relación de tipo *extend* entre dos casos de uso para especificar diferentes variantes del mismo caso de uso.
- Esta relación implica que el comportamiento de un caso de uso es diferente dependiendo de ciertas circunstancias. Representa comportamientos opcionales.
- El caso de uso base debe ser completamente funcional por si mismo, sin considerar el caso de uso que se está extendiendo.
- La flecha en el caso de las relaciones "extend" va hacia el caso de uso "original".



Reservar se comportará de forma distinta según *Denegar reserva*.

Elementos Básicos: Asociaciones

- **Generalización/Especialización**: En un diagrama de casos de uso también pueden mostrarse generalizaciones para mostrar que diferentes elementos están relacionados como tipos de otros
- Son especializaciones de un caso de uso o actor más general.
- Similar a las relaciones de herencia en los diagramas de clase.
- Son aplicables a actores o casos de uso.

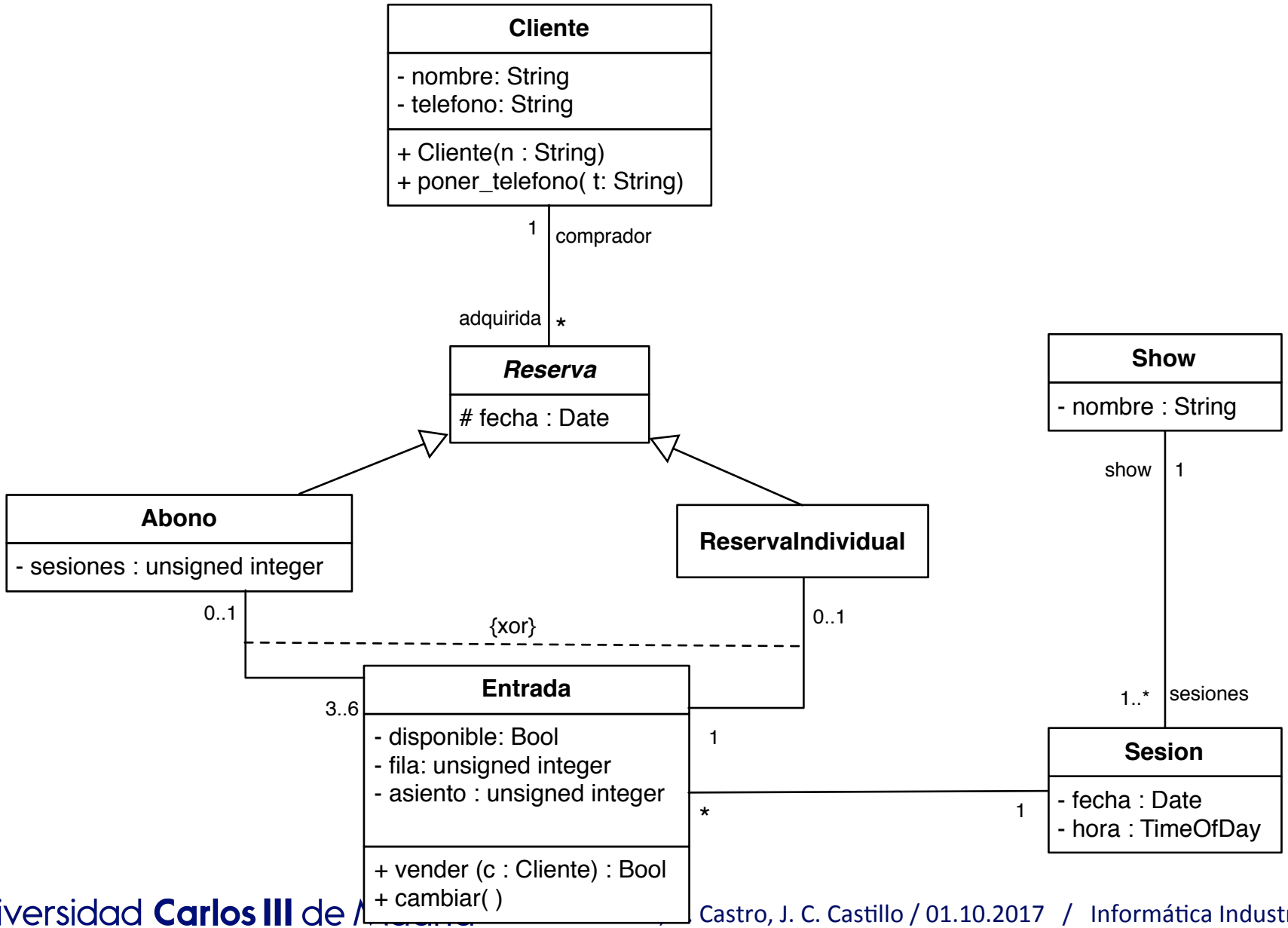


Reservar por internet y *Reservar por agente* son un tipo de *Reservar*.

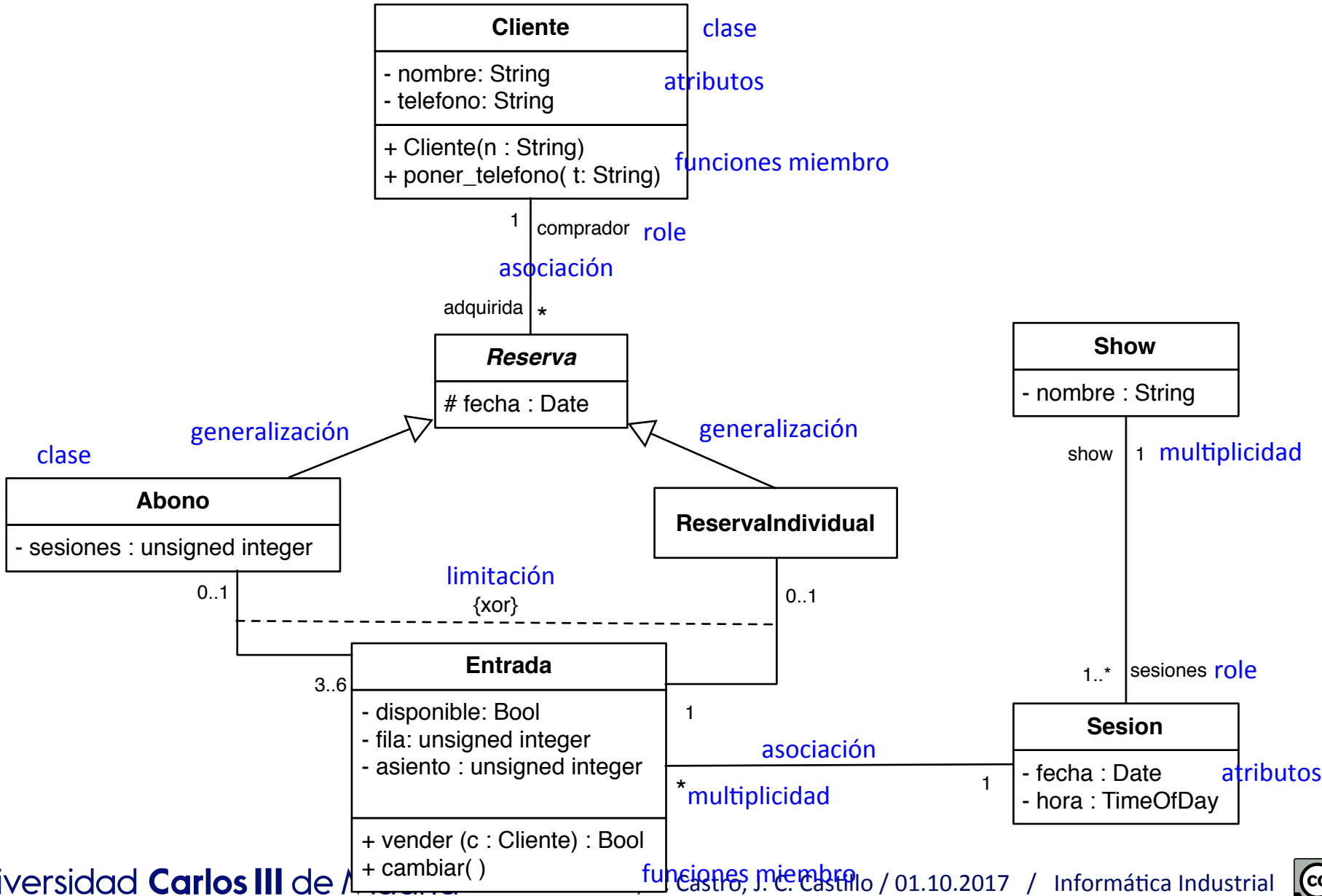
Ejemplos

- A continuación se muestran algunos ejemplos de diagramas vistos referidos a un sistema de venta de entradas

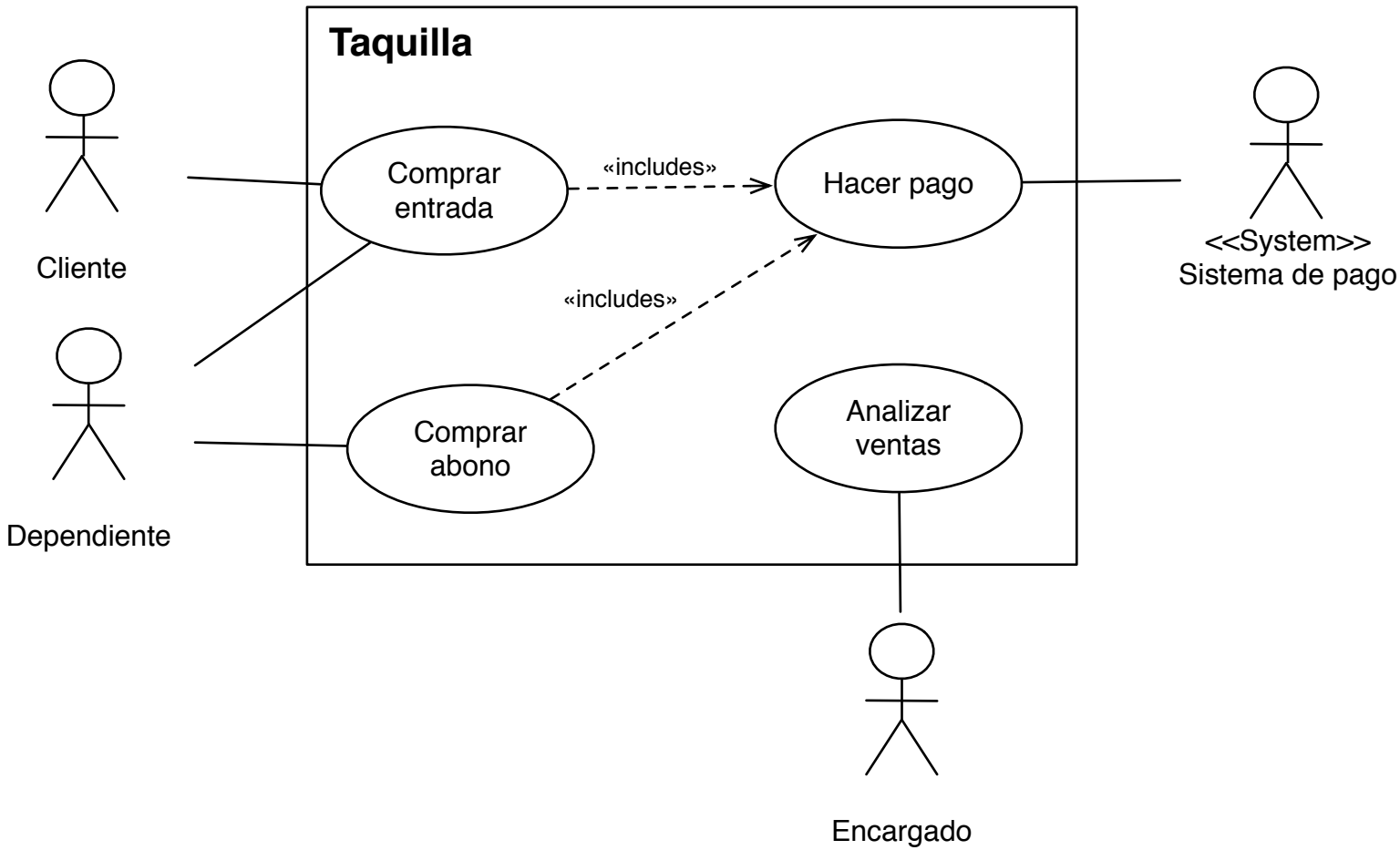
Ejemplo: Diagrama de clases



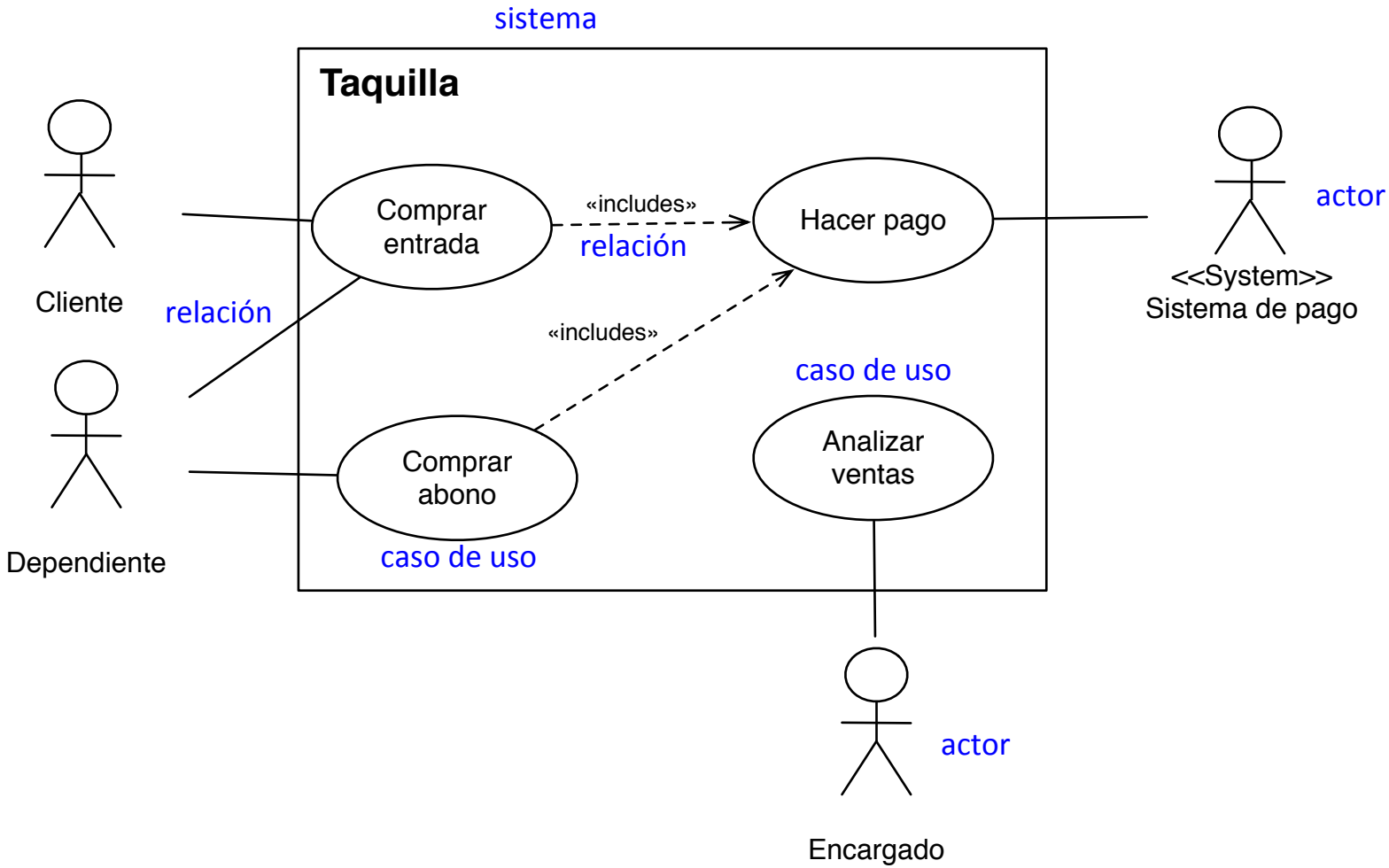
Ejemplo: Diagrama de clases



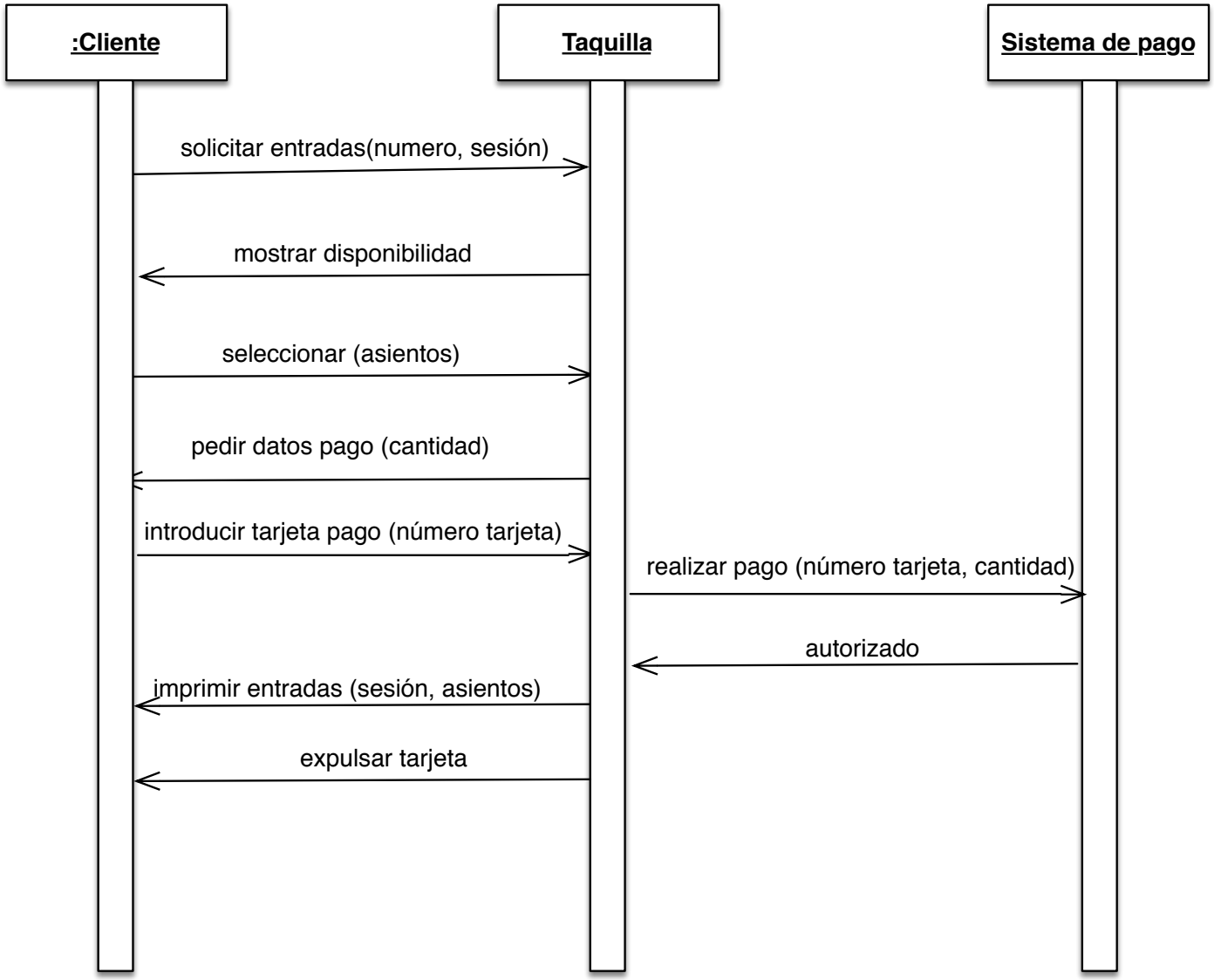
Ejemplo: Diagrama de casos de uso



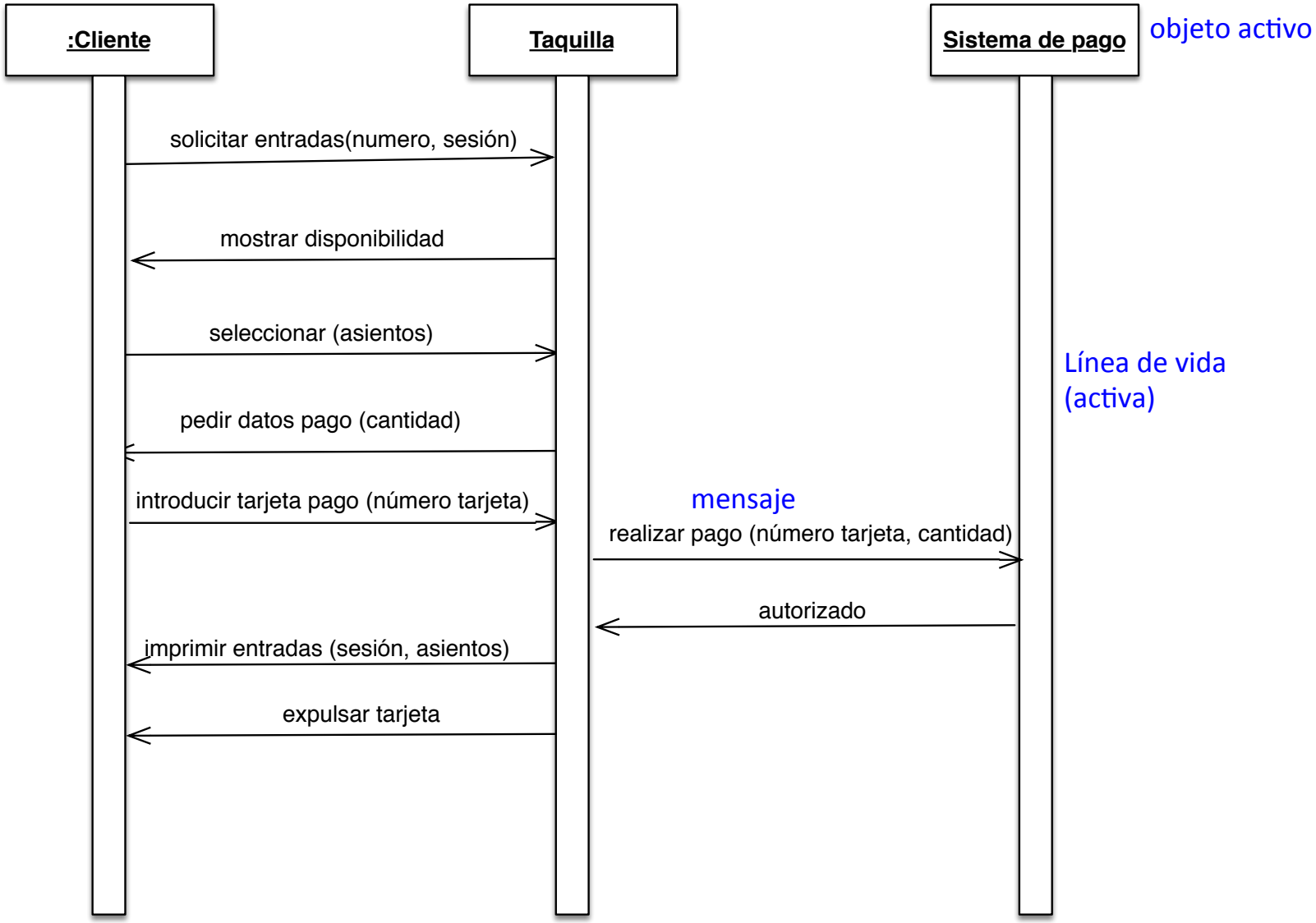
Ejemplo: Diagrama de casos de uso



Ejemplo: Diagrama de secuencia

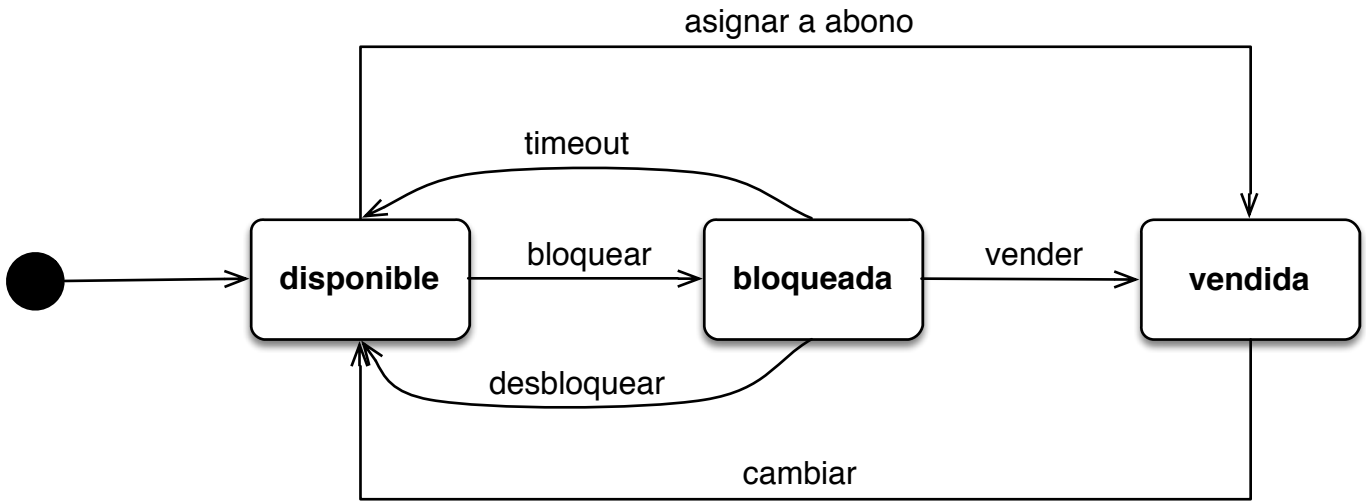


Ejemplo: Diagrama de secuencia



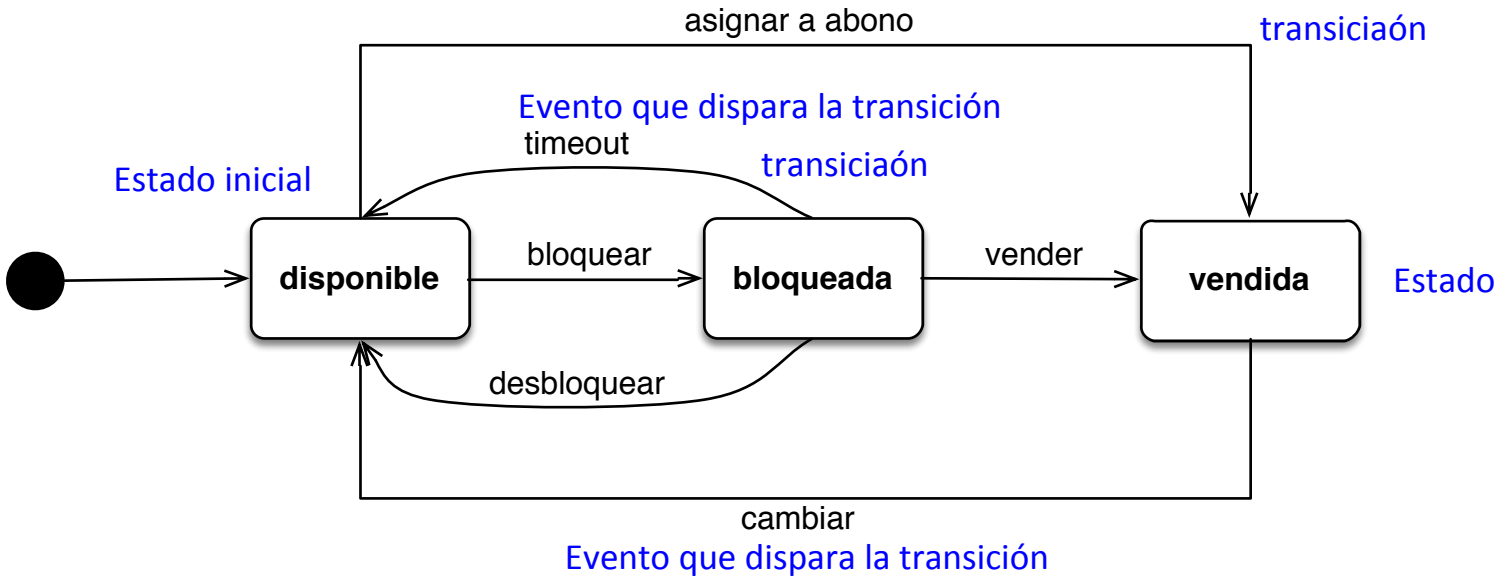
Ejemplo: Diagrama de estados

- Objeto de tipo *Entrada*

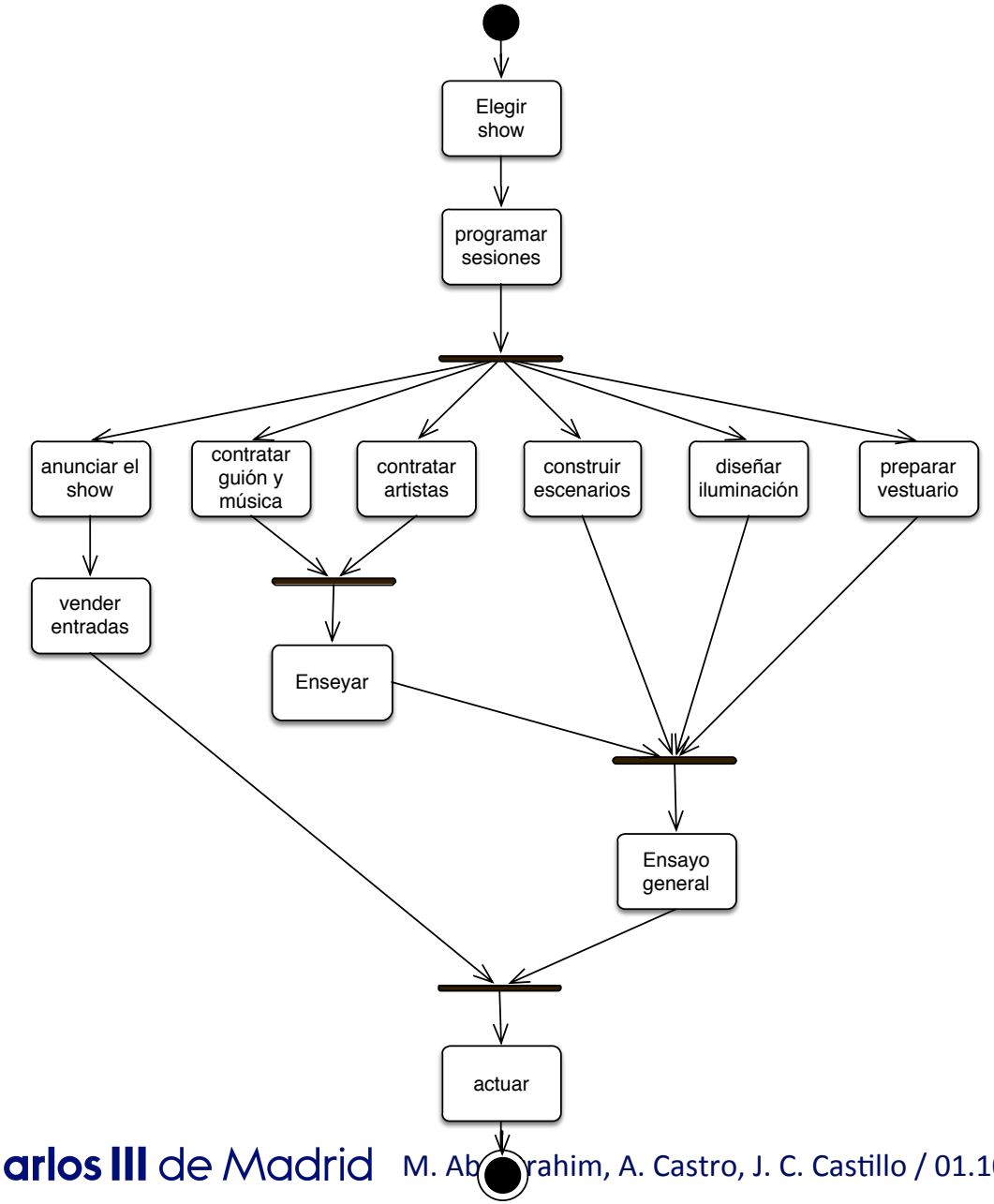


Ejemplo: Diagrama de estados

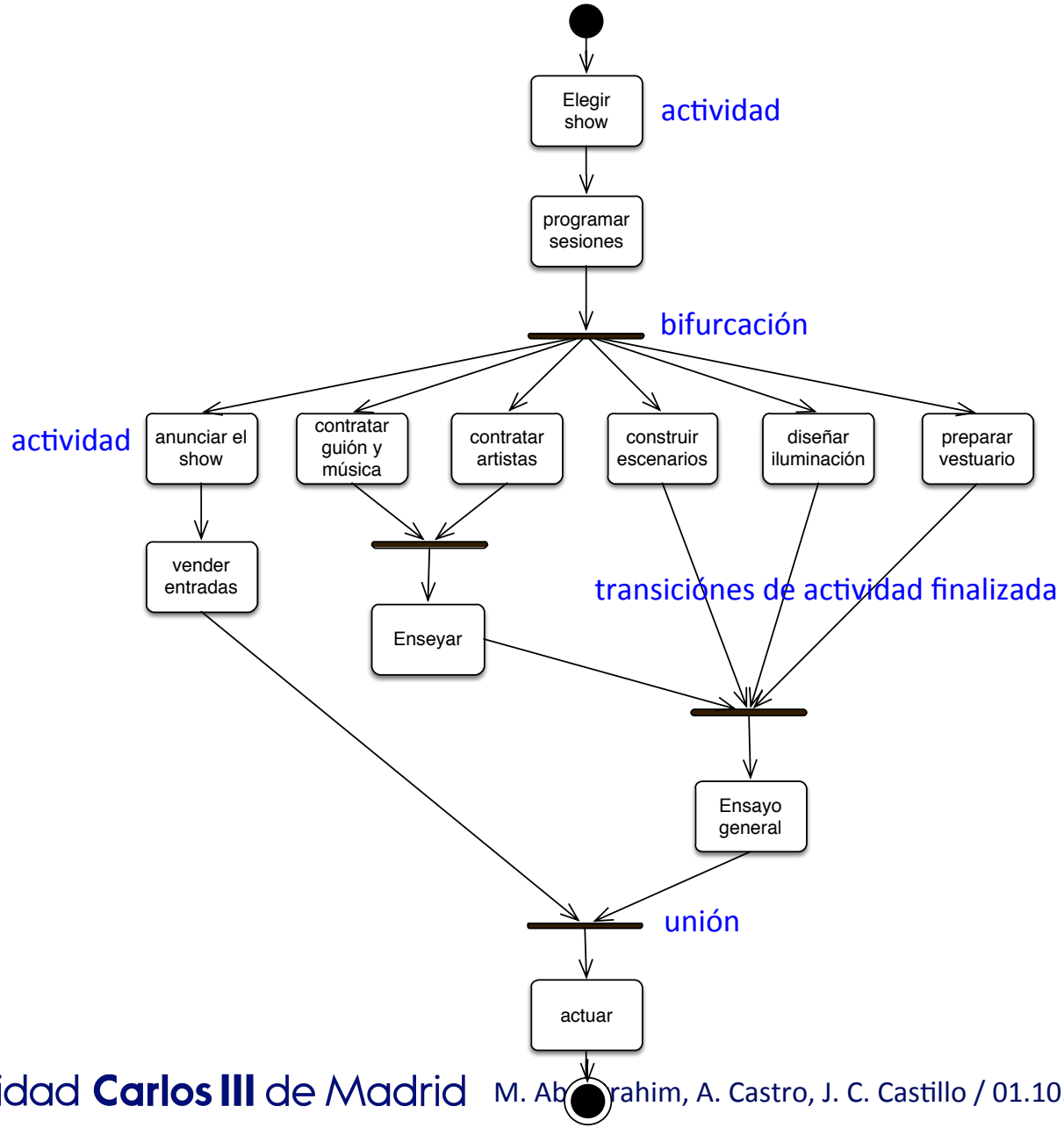
- Objeto de tipo *Entrada*



Ejemplo: Diagrama de actividad



Ejemplo: Diagrama de actividad



uc3m | Universidad Carlos III de Madrid