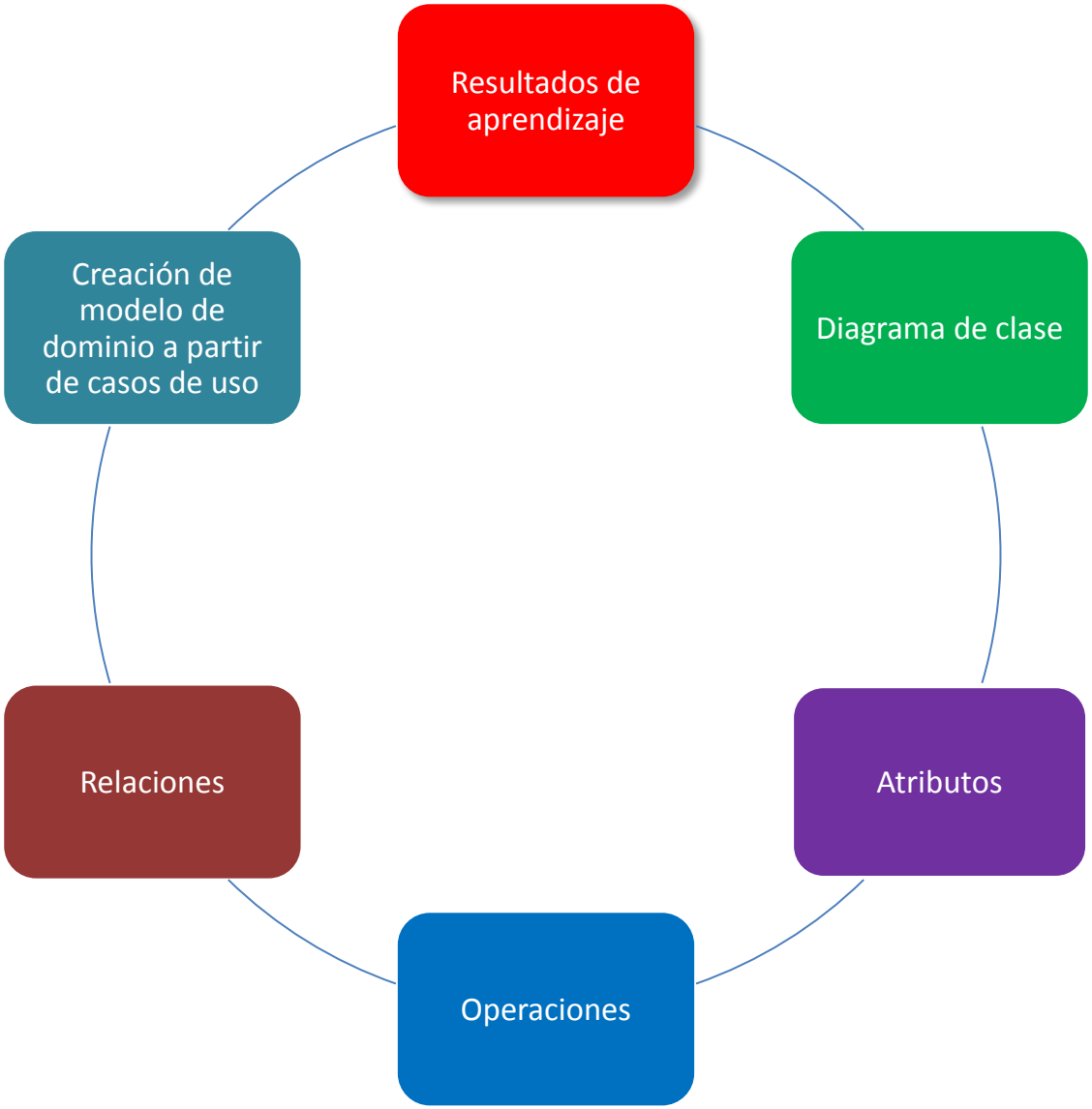
Universidad de Playa Ancha

Análisis de Sistemas
CIF 5555
2022-1


Resultados de aprendizaje
Diagrama de clase
Atributos
Operaciones
Relaciones
Creación de modelo de dominio a partir de casos de uso

Resultados de aprendizaje

- El alumno debería:
  - Aplicar el proceso de Ingeniería de Requisitos para la elicitación de las necesidades de los clientes y usuarios

DIAGRAMA DE CLASES Y OBJETOS

Resumen de la lección anterior
Esquema de la lección

- Lecciones anteriores:
  - El diagrama de casos de uso
- Esta lección y la siguiente
  - Descripción del dominio

- Específicamente en esta lección: diagrama de clases
- El diagrama más conocido y utilizado

Introducción a UML

- UML: descripción de la estructura y comportamiento de un sistema
  - los lenguajes de programación no son suficientemente abstractos para el diseño
  - UML es un estándar abierto
- ¿Cuál UML es válido?
  - un lenguaje descriptivo: sintaxis formal rígida (como lenguaje de programación)
  - un lenguaje normativo: formado por el uso y convención (como UML)
  - está bien omitir cosas de los diagramas UML si no las necesitan el equipo/supervisor/instructor

Usos de UML

- como un esbozo: para comunicar aspectos del sistema
  - diseño hacia adelante: se hace UML antes de programar
  - diseño hacia atrás: se hace UML después de programar como documentación
  - a menudo se hace en pizarra o papel
  - usado para obtener ideas escogidas aproximadas

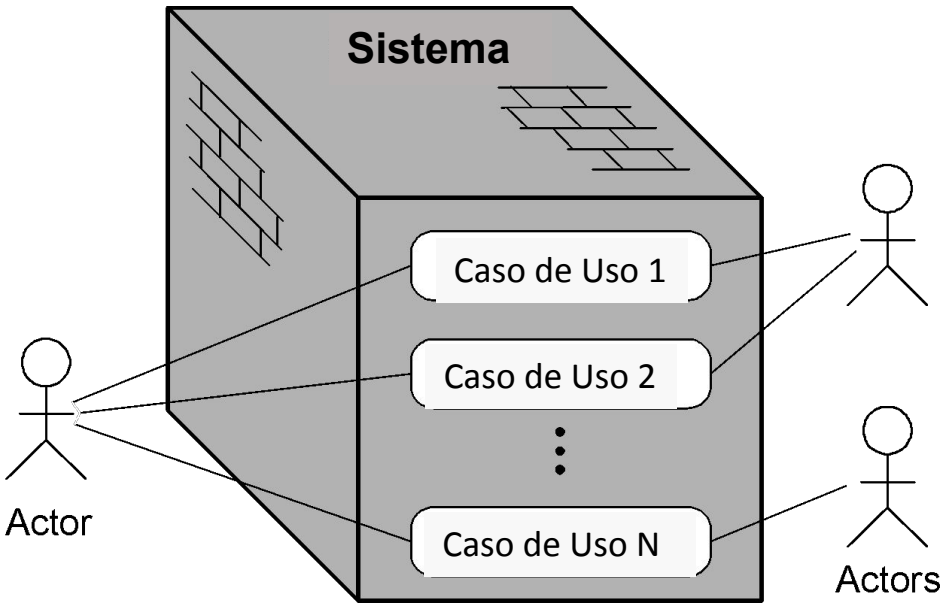
- como un anteproyecto: un diseño completo a ser implementado
  - con herramientas CASE (Computer-Aided Software Engineering)

- como un lenguaje de programación : con las herramientas adecuadas, el código puede ser auto-generado y ejecutado desde UML
  - adecuado sólo si es más rápido que programar en un lenguaje "real"

Casos de uso versus modelo de dominio

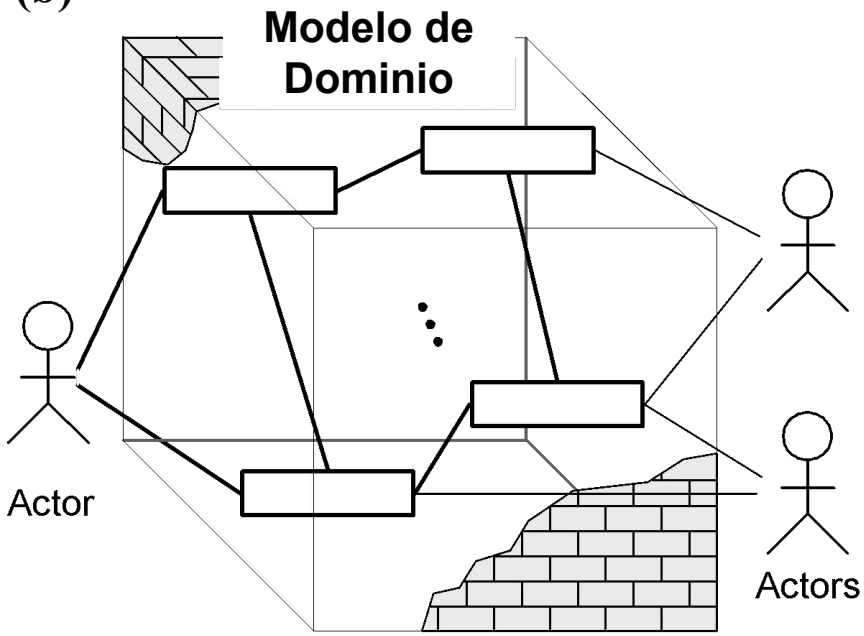
En el análisis de casos de uso, **consideramos el sistema como** "caja negra"

(a)

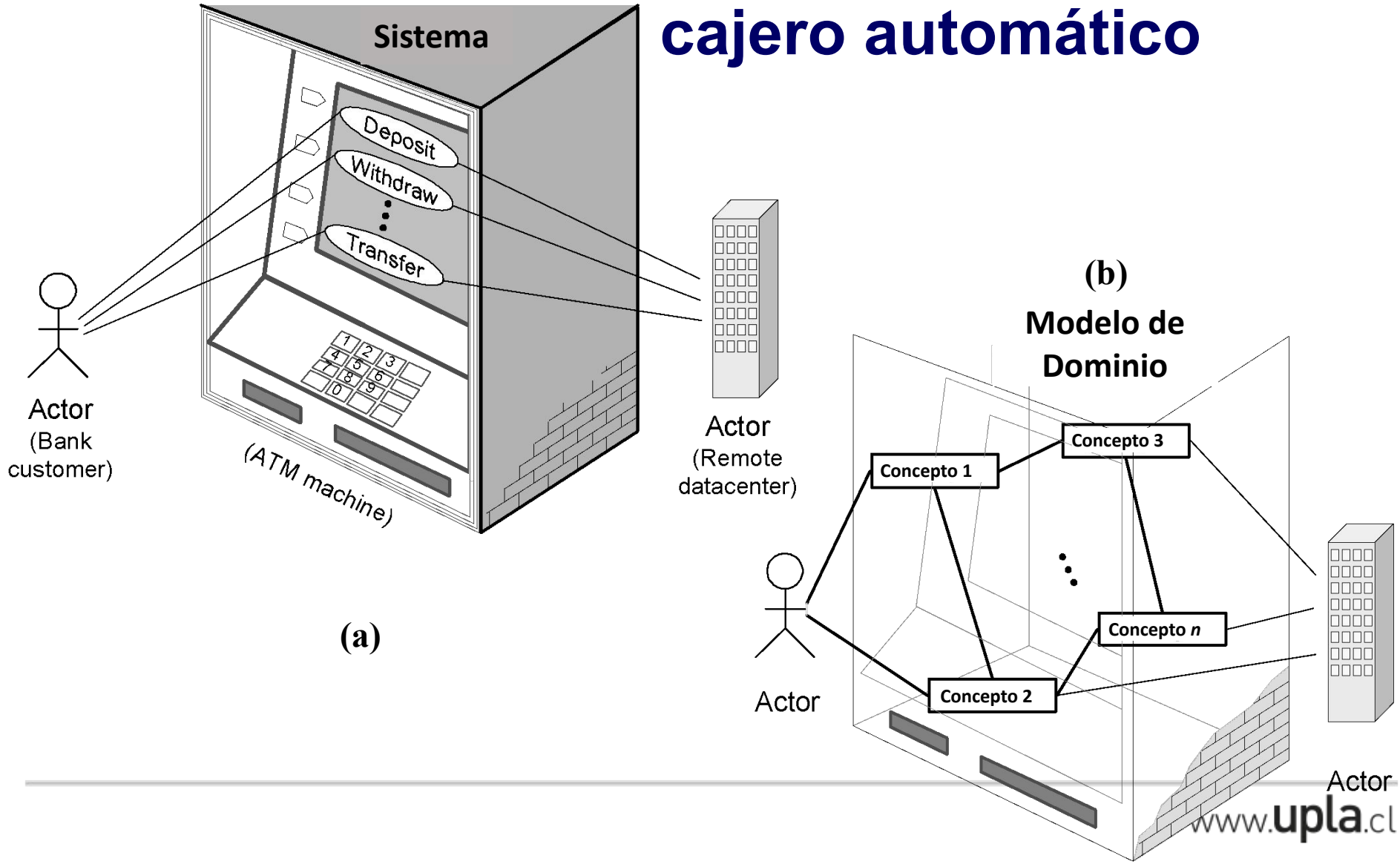


En el análisis de dominio, **consideramos el sistema como** "caja transparente"

(b)



Ejemplo de Modelo de dominio: cajero automático



Modelo conceptual - Diagramas de clase UML

- ¿Qué es un diagrama de clase UML?
  - **diagrama de clase UML**: un descripción de las clases en un sistema, sus campos y métodos, y conexiones entre las clases que interactúan o heredan entre ellas
- ¿Cuáles son algunas cosas que no son representadas en un diagrama de clase UML?
  - detalles de cómo las clases interactúan entre sí
  - detalles algorítmicos; cómo se implementa un comportamiento particular

Modelo conceptual - Clase

- Representa un concepto de la aplicación que es modelada
  - una entidad física, por ejemplo: automóvil
  - una entidad del sector aplicativo: factura
  - una entidad lógica: un gestor de pedidos
  - una entidad aplicativa: un botón sobre la interfaz de usuario
  - una entidad “informática”: una tabla hash
  - una entidad de comportamiento: una tarea
- Una clase representa colectivamente todos los objetos que tienen el mismo comportamiento

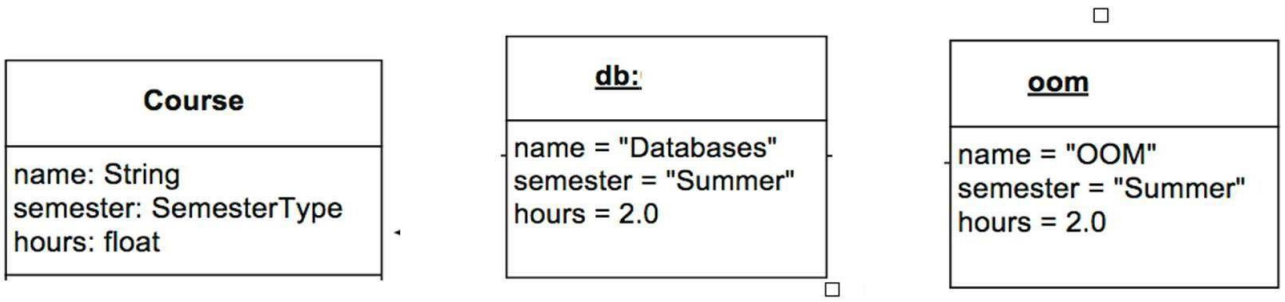
Clases y objetos

- Un objeto es una entidad caracterizada por
  - Una identidad
  - Un estado
  - Un comportamiento

- Una clase describe
  - un conjunto de objetos con características similares
  - es decir, objetos que tienen el mismo tipo

Clasificadores e instancias

- Las clases son clasificadores
- Los objetos son instancias
- Modelar a nivel de clasificador significa restringir modelos a nivel de instancia



Lo que veremos

Diagrama de clase

Diagrama de objetos

- también se llama diagrama de instancia
- puede ser útil cuando las conexiones entre objetos son complicadas

Diagrama de clase

- Una clase captura un concepto en el dominio del problema o la realización.

- El diagrama de clases describe:
  - El tipo de objetos que forman parte de un sistema sw o de su dominio.
  - Las relaciones estáticas entre ellos: los elementos y las relaciones entre ellos no cambian con el tiempo

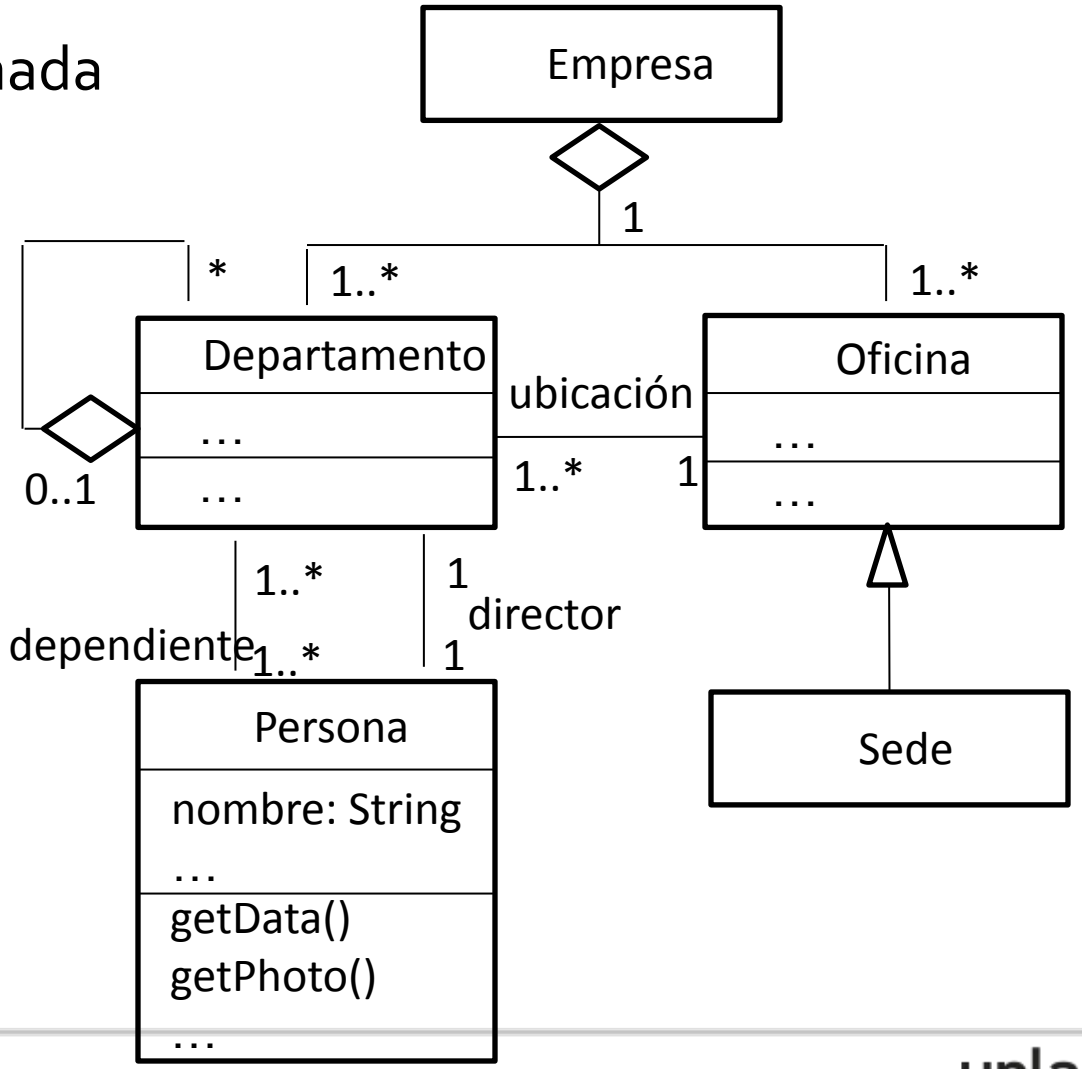
- Los diagramas de clases también muestran las propiedades y operaciones

Modelo conceptual - Diagrama de clase

- nombre de la clase arriba de la caja
  - se escribe <<interface>> arriba del nombre de interfaces
  - se usa cursiva para un nombre de clase abstracta
- atributos (opcional)
  - se debería incluir todos los campos del objeto
- operaciones / métodos (opcionales)
  - se puede omitir métodos banales (get/set)
    - Pero no hay que omitir ningún método desde una interfaz…
  - no se debería incluir métodos heredados

Ejemplo: clases

- Una empresa está formada por departamentos y oficinas
- Un departamento tiene un director y varios empleados.
- Un departamento está ubicado en una oficina.
- Existe una estructura jerárquica de los departamentos
- La sede son oficinas



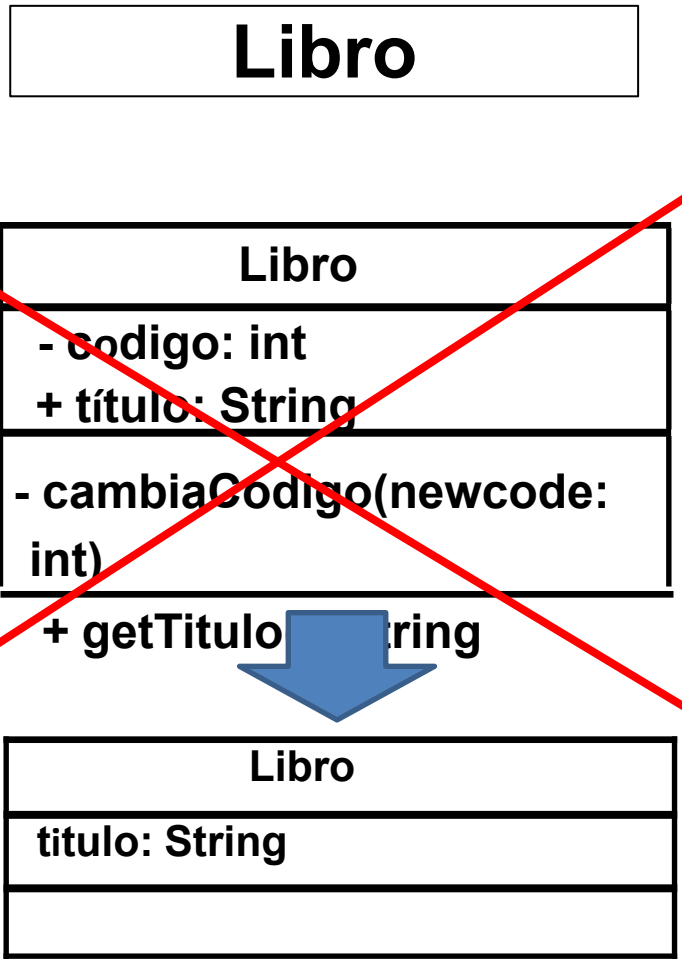
Usos del diagrama de clases

- El diagrama de clases se puede utilizar
  - en diferentes niveles de detalle
  - en diferentes fases del proyecto
  - hasta la generación del código en lenguajes OO

No es obligatorio indicar atributos y operaciones

- Se pueden omitir atributos y operaciones

- Por ejemplo, cuando se usa el diagrama de clases para describir el dominio
  - Las transacciones se consideran excesivamente detalladas y, por lo general, se omiten.
  - Se especifican los atributos útiles para caracterizar el elemento de dominio, los detalles de implementación no
  - La visibilidad no



Semántica

- Un objeto es una entidad caracterizada por
  - Una identidad, un estado, un comportamiento.
- (Los valores de) los atributos definen el estado del objeto
- Las operaciones definen su comportamiento

Atributos

- Atributo - es un valor de datos lógicos de un objeto

  Es una propiedad designada de una clase que describe valores aceptados por cada objeto de la clase

- Tipo de atributo: Una especificación del comportamiento externo y/o la implementación del atributo
  - Incluir los siguientos atributos en un modelo de dominio
    - aquellos para los que los requisitos sugieren o implican una necesidad por recordar información
  - Por ejemplo, un recibo de venta normalmente incluye un atributo de fecha y hora

Nombre Atributo:Tipo atributo

Atributos (cont.)

- Atributos en un modelo conceptual deberían ser preferiblemente atributos simples o valores de datos puros
- Tipos de atributo simple comunes incluyen
  - boolean, date, number, string, time

Sintaxis Atributos

visibilidad nombre: tipo [multiplicidad] = valorInicial {propiedad}

solo se requiere el nombre

multiplicidad:

para indicar un array de valores

color: Integer[3] ***Modelo RGB***

segundoNombre: String [0..1] ***cero para permitir valor null***

nombre: String ***La multiplicidad [1] se puede omitir***

propiedad

{> 0, <10} ***restricciones sobre los valores que el atributo puede tener***

{ordered}, {unique} ***tienen sentido cuando un atributo tiene una multiplicidad de valores:***

multiconjuntos

- ***ordered*** → ***listas ordenadas en lugar de conjuntos o***
- ***unique*** → ***sin repeticiones, como en los conjuntos***

Ejemplos de atributos

n: char caracter, tipo predeterminado

n: String cadena, tipo predeterminado

g: Gra con tipo Gra definido en el modelo

n: Integer = 1 {> = 0} entero no negativo, inicialmente = 1

p: Integer [2] {> 0} punto del cuadrante positivo

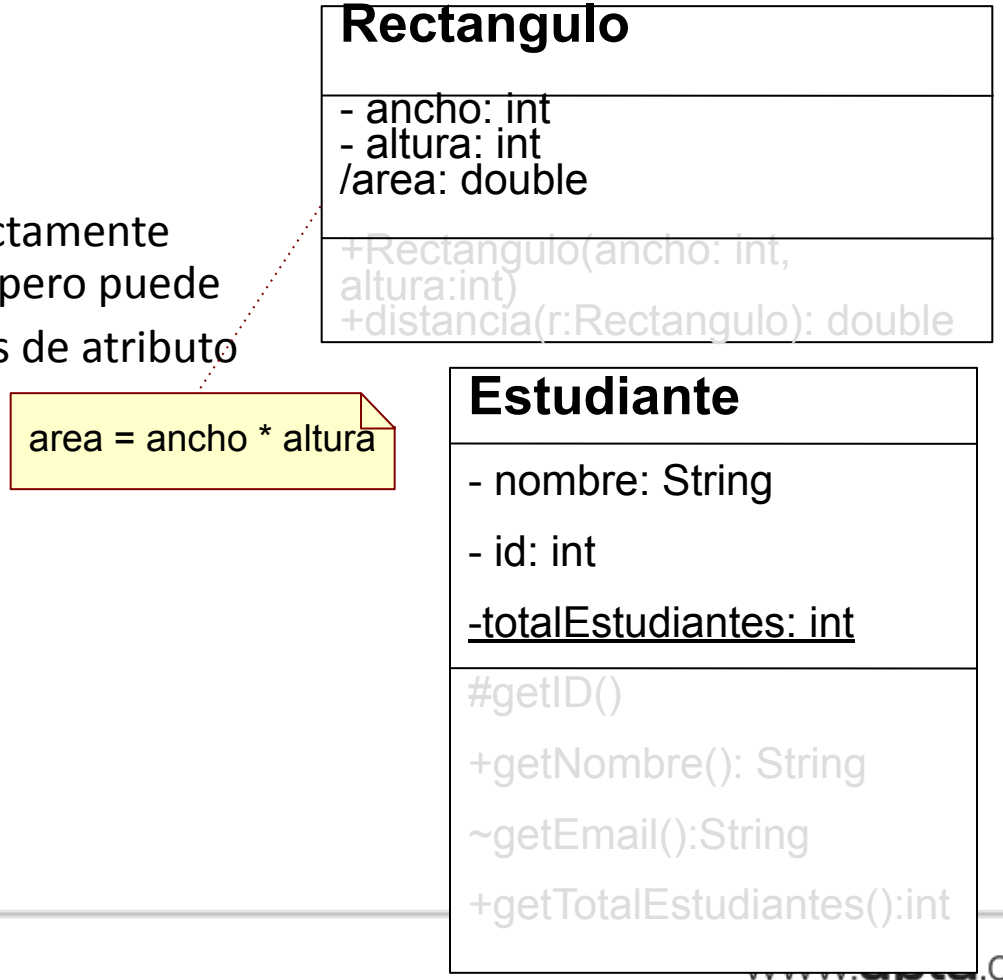
s: Integer [10] {> 3, <33, unique} conjunto (unique -> sin repeticiones) de 10 números entre 3 y 33

col: Integer [3] {> = 0, <= 255, ordered} lista (ordered -> la posición es significativa) de 3 número, entre 0 y 255

nombre: String [1..2] {ordered, unique} se debe tener un nombre, opcionalmente puede tener un segundo nombre, pero diferente al primero

Atributos de clase

- atributos (campos, variables de instancia)
  - *visibilidad nombre* : *tipo* [*multiplicidad*] = *valor_por_defecto {propiedad}*
  - subrayar atributos estáticos
  - **atributo derivado**: no es estrictamente necesario, no es almacenado, pero puede ser computado de otros valores de atributo
  - ejemplo de atributo:
    - balance : double = 0.00



Visibilidad de atributos y operaciones

- Un elemento es visible fuera del espacio de nombres que lo contiene, según su tipo de visibilidad.

  + public: accesible a cualquier elemento que pueda ver y usar la clase

  # protected: accesible a cada elemento descendente

  - private: solo las operaciones de la clase pueden ver y usar el elemento en cuestión

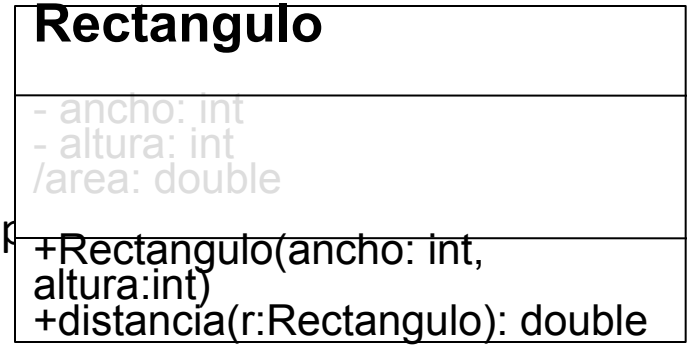
  ~ package: accesible solo a elementos declarados en el mismo paquete

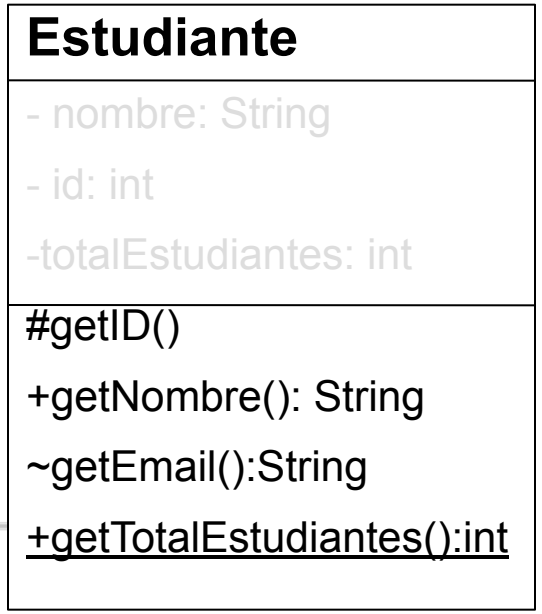
Registrando términos en Glosario

- Definir todos los términos que necesitan ser clarificados en un **glosario** o **diccionario del modelo.**

Operaciones / métodos de clase

- operaciones / métodos
  - visibilidad nombre (lista de parámetros) : tipo_retornado {propiedad}
  - subrayar métodos estáticos
  - tipos de parámetros elencados como (nombre: tip
  - omitir tipo_retornado en constructores y cuando el tipo de retorno es “void”
  - ejemplo de método:
    + distancia(p1: Punto, p2: Punto): double





Sintaxis de las operaciones

visibilidad nombre (listaParametros) : tipoRetorno

solo obligatorio nombre()

listaParametros :: = Ø | declaración de parámetro, listaParametros

declaración de parámetro:: = dirección nombre: tipo = default

solo obligatorio nombre (entendiéndose que no es obligatorio declarar los parámetros)

in, out, inout

Valor asignado al parámetro en ausencia de argumento

Ejemplos de operaciones

+ suma (a: Integer, b: Integer): Integer

método público que, dados dos enteros, devuelve un entero

+ suma (a: Integer, b: Integer= 10): Integer

como arriba, con 10 valores predeterminados del segundo parámetro

- gra (): Gra

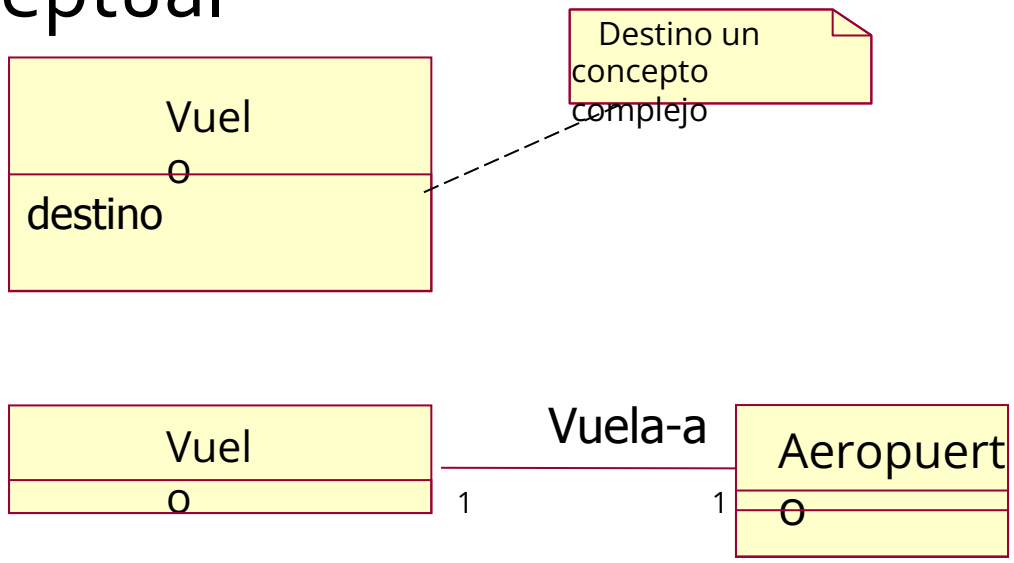
método privado que devuelve un objeto de tipo Gra

Atributos Complejos

- Valores de datos puros - expresados como atributos; ellos no ilustran comportamientos específicos;
  - Ejemplo – Número de teléfono
  - Una Persona pueden tener muchos teléfonos
- Tipo de atributos no primitivos
  - representa atributos como tipos no primitivos (conceptos u objetos) si
    - está compuesto de secciones separadas (nombre de una persona)
    - hay operaciones asociadas con él como validación
    - es una cantidad con una unidad (pago tiene una unidad de moneda)

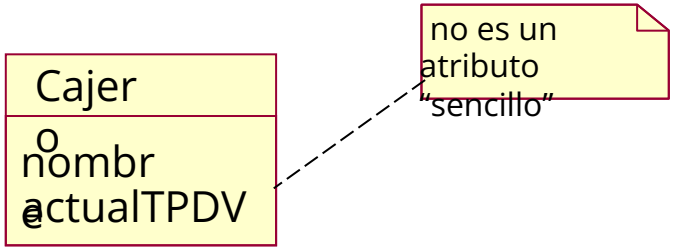
Atributos Complejos

- Es deseable mostrar atributos no primitivos como conceptos en un modelo conceptual

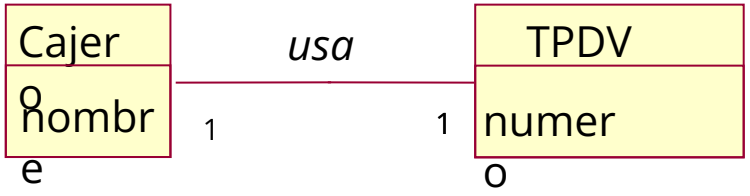


Atributos complejos

- peor ejemplo



- mejor ejemplo

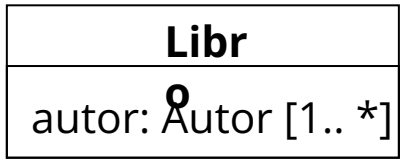


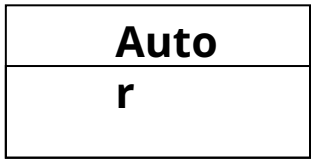
Relacionar dos elementos con relaciones, no con atributos, en el modelo conceptual.

Qué se modela con una clase

- autor como atributo

- vs autor como clase





Enumeraciones

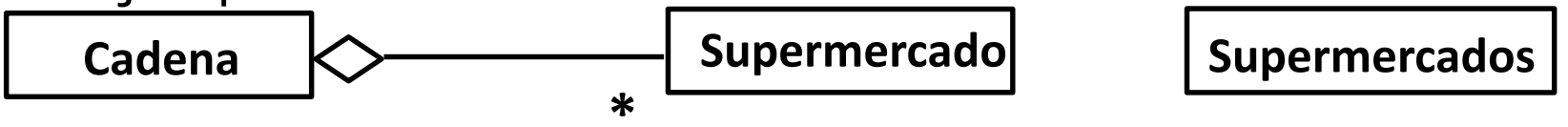
- Las enumeraciones se utilizan para especificar un conjunto de valores predeterminados que no tienen más propiedades que su valor simbólico.
  - Se utilizan cuando un atributo solo puede tener un conjunto fijo de valores.

- En UML están representados por clases.
  - etiquetado por el estereotipo <<enumeration>> (un estereotipo, pensar en una palabra clave)
  - con un nombre (el tipo) y el conjunto de valores que los atributos de ese tipo pueden asumir

La portada de los libros puede solo ser o toda roja, o toda blanca, o toda azul

Clases UML vs entidades (DB)

- DB: las clases están pensadas como colecciones. Se entiende que
  - hay varias instancias.
  - hay operaciones para visitar todas las instancias.
- Por lo tanto, por ejemplo, sustantivo singular vs sustantivo plural.
- La diferencia es más significativa desde una perspectiva de diseño que desde una descripción de dominio.
  - En el diseño OO y por lo tanto en UML utilizo "ListaDeQcosa" como un agregado de "Qcosa"
- Un ejemplo del dominio:



Clases UML

Entidad

Relaciones

- Una relación representa un vínculo
  - entre dos o más objetos
  - normalmente instancias de diferentes clases

Relaciones

- Entre elementos de un modelo
- Veremos lo siguiente:

Entre Clases	Entre Objetos
Asociación Agregación Composición	Conexión Agregación Composición
Generalización	(no definida)
Realización	(no definida)
Dependencia (de uso, de instancia, etc.)	

Relaciones entre clases

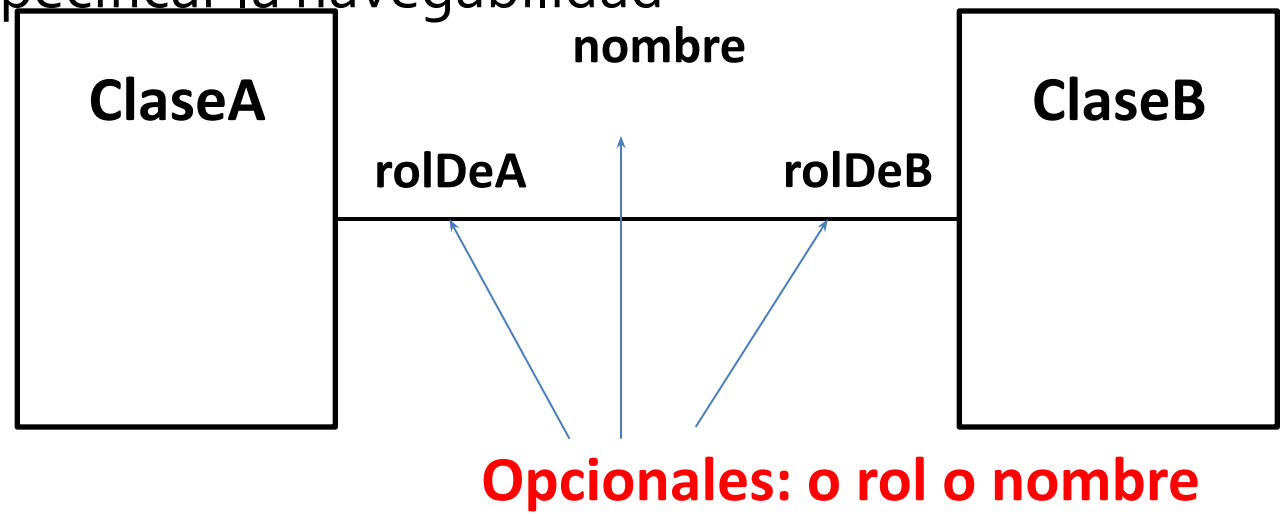
- asociación (una relación de uso)
  - dependencia
  - agregación
  - Composición
- generalización
  - herencia entre clases
  - implementación de interface

Modelo Conceptual - Asociación

- Objetivos
  - Identificar asociaciones dentro de un modelo conceptual
  - distinguir entre asociaciones que se deben conocer de asociaciones que hay que sólo entender

Asociación (binaria): sintaxis

- Asociación - una relación entre conceptos que indica alguna conexión significativa e interesante
  - A veces a nivel de implementación (no de descripción de dominio) una flecha (eventualmente doble) para especificar la navegabilidad



Asociación (binaria): nombre y roles

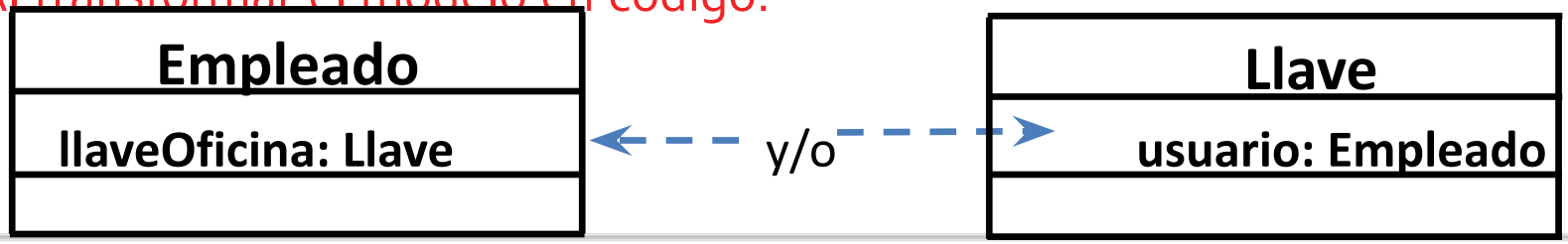
- nombre y roles: minúsculas
- nombre de asociación: normalmente un verbo
  puede ser seguido por ▶ para entender de qué lado leer
- rol: normalmente un sustantivo
- Formalmente opcionales,
  - es útil tener el nombre de la asociación o la indicación de los roles
  - inútil ambos

Asociación (binaria): ejemplo

- nombre de la asociación

- Los roles de los objetos en la relación se hacen explícitos.
  - está la llave de la oficina del
  - i es el usuario

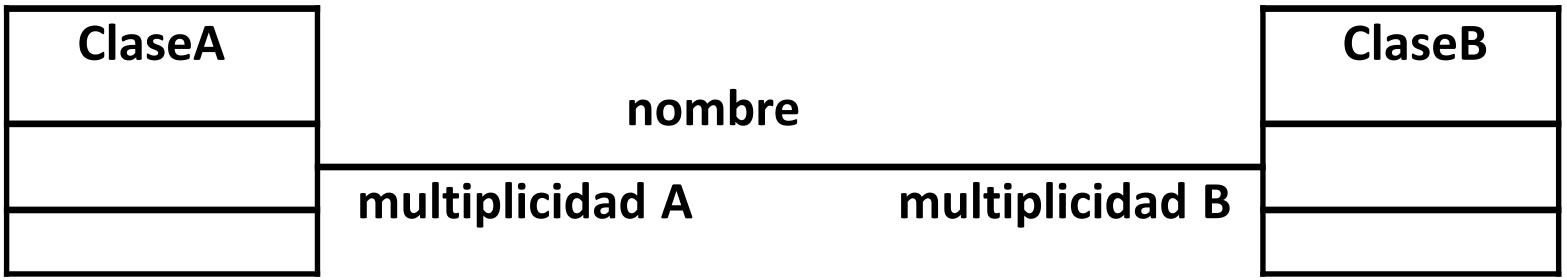
Al transformar el modelo en código:



Roles en Asociaciones

- Cada uno de los dos extremos de una asociación es llamada un rol. Los roles tienen
  - *nombre*
  - *expresión de multiplicidad*
  - *navegabilidad*

Asociaciones: restricciones de multiplicidad



Número de objetos involucrados en la asociación

en un instante dado

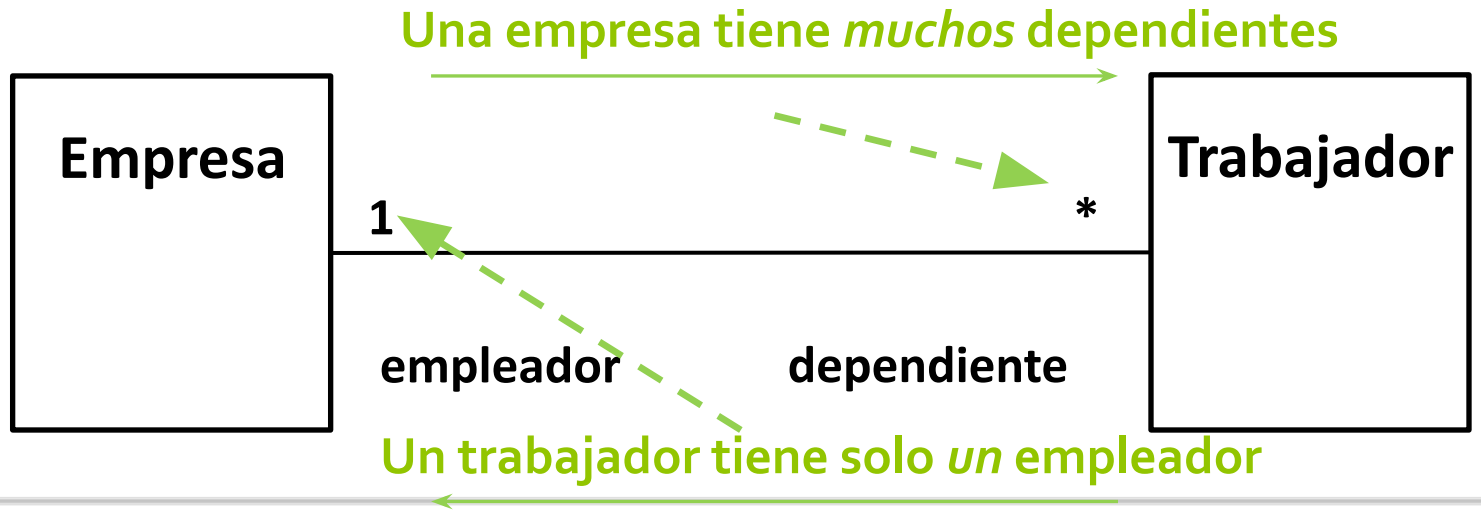
Multiplicidad de relaciones

- Las multiplicidades se pueden definir indicando los extremos inferior y superior de un intervalo.
  - Ejemplo 2..4 para los participantes en un juego de canasta

- el límite inferior puede ser cero o un número positivo
- el superior un número positivo o * (indefinido)
- n..n ≡n
- 0 .. * ≡*
- 1 es el predeterminado y se puede omitir

Multiplicidad: un ejemplo

- Multiplicidad en un momento particular en el tiempo
  - Un objeto Empresa puede estar en relación con muchos objetos Trabajador
  - Un objeto Trabajador puede estar en relación con un solo objeto Empresa

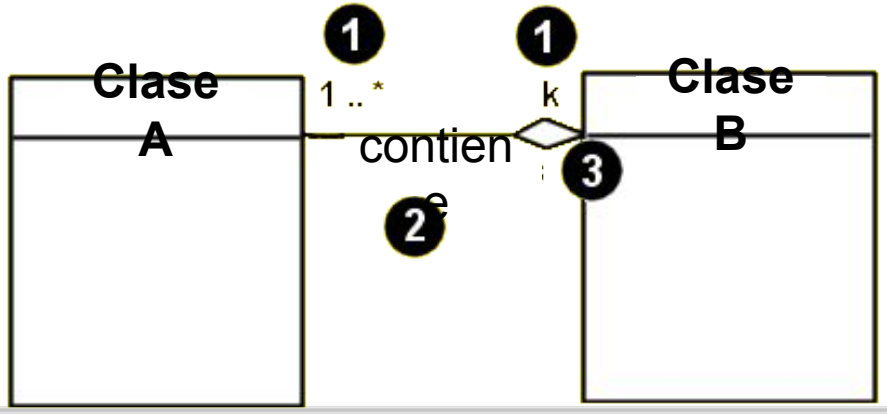


Asociación - Multiplicidad

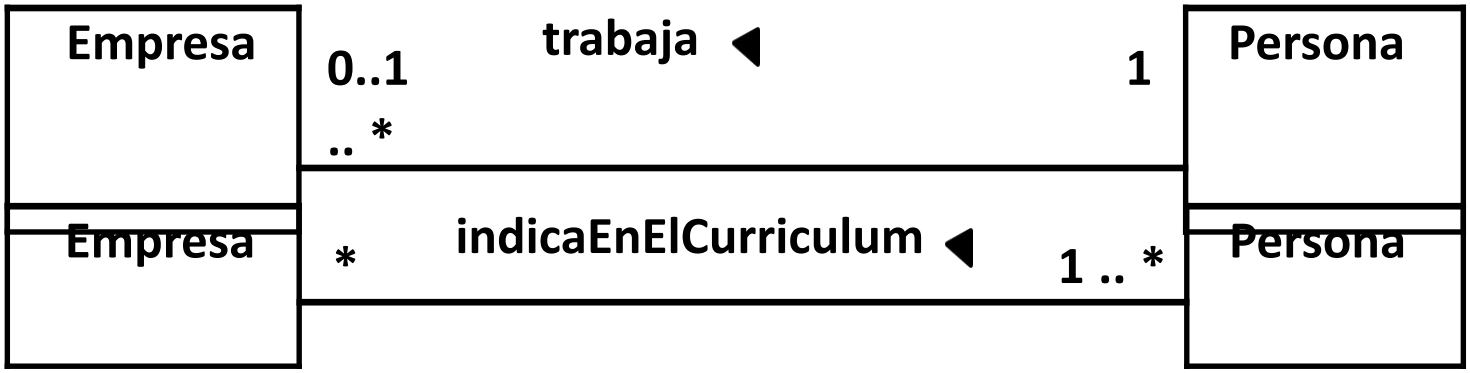
- Multiplicidad: indica el número de objetos de una clase que puede estar relacionada a un objeto único de una clase asociada
- Puede ser uno de los siguientes tipos
  - 1 to 1, 1 to 0..*, 1 to 1..*, 1 to n, 1 to 1..n

La multiplicidad está ligada al nombre de la asociación

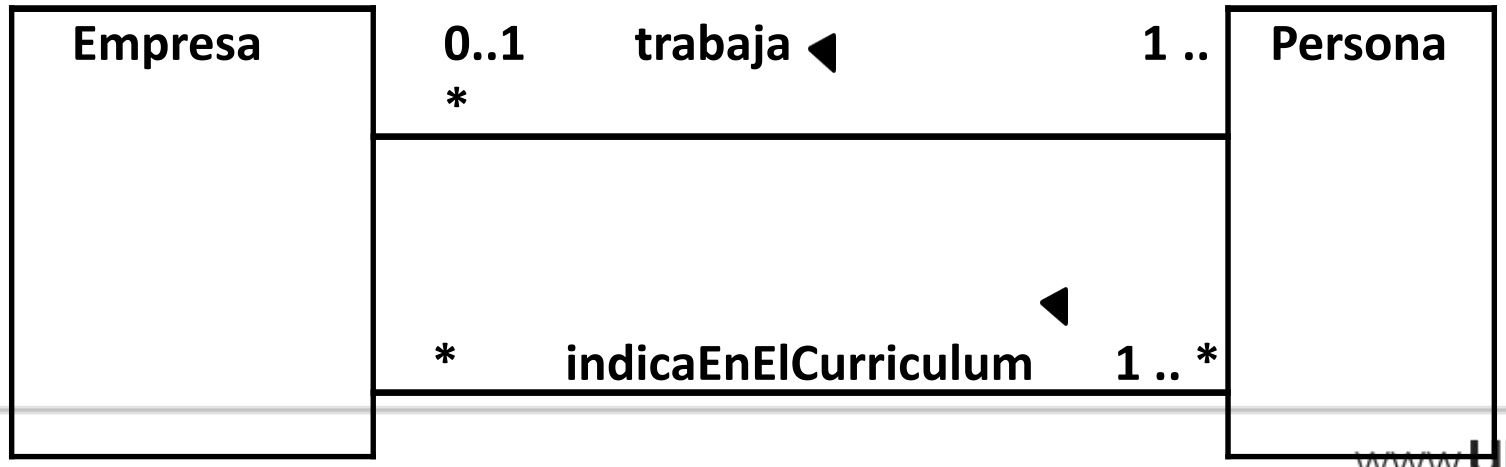
1. multiplicidad (cuántos son usados)
   - * ⇒ 0, 1, o más
   - 1 ⇒ 1 exactamente
   - 2..4 ⇒ entre 2 y 4, inclusive
   - 5..* ⇒ 5 o más
2. nombre (qué relación tienen los objetos)
3. navegabilidad (dirección)



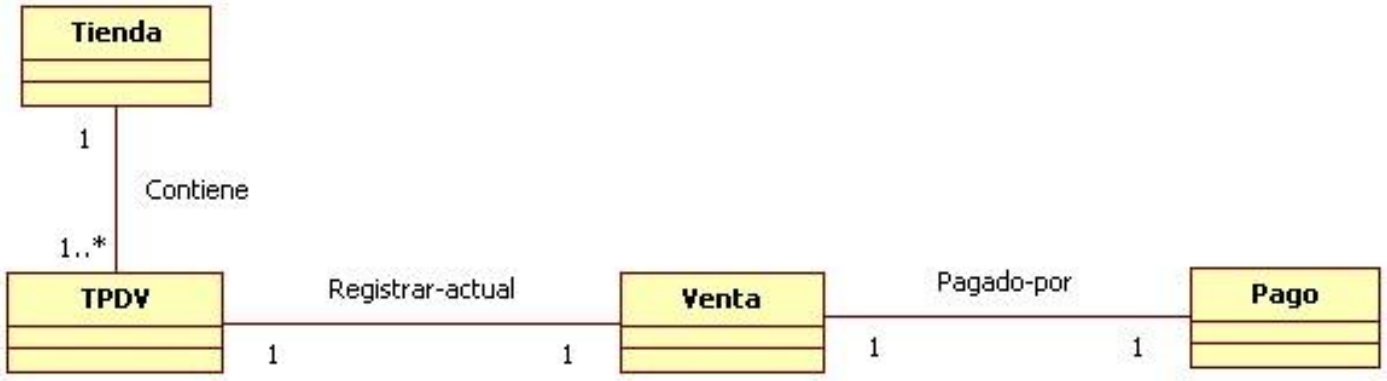
La multiplicidad está vinculada al nombre de la asociación



También se pueden tener múltiples asociaciones entre dos clases



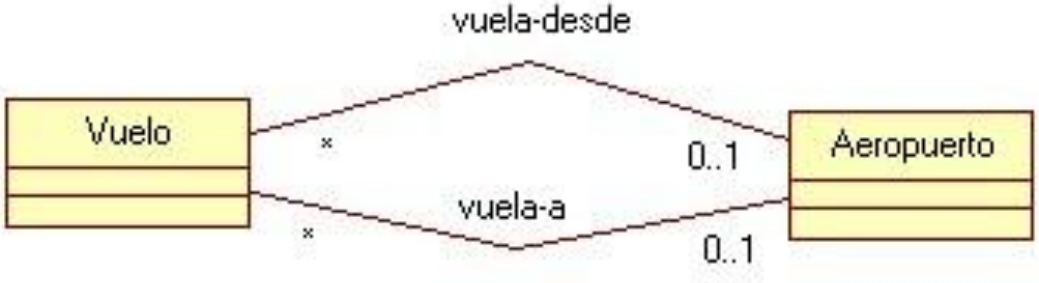
Asociaciones



Las asociaciones generalmente se leen de izquierda a derecha, de arriba hacia abajo

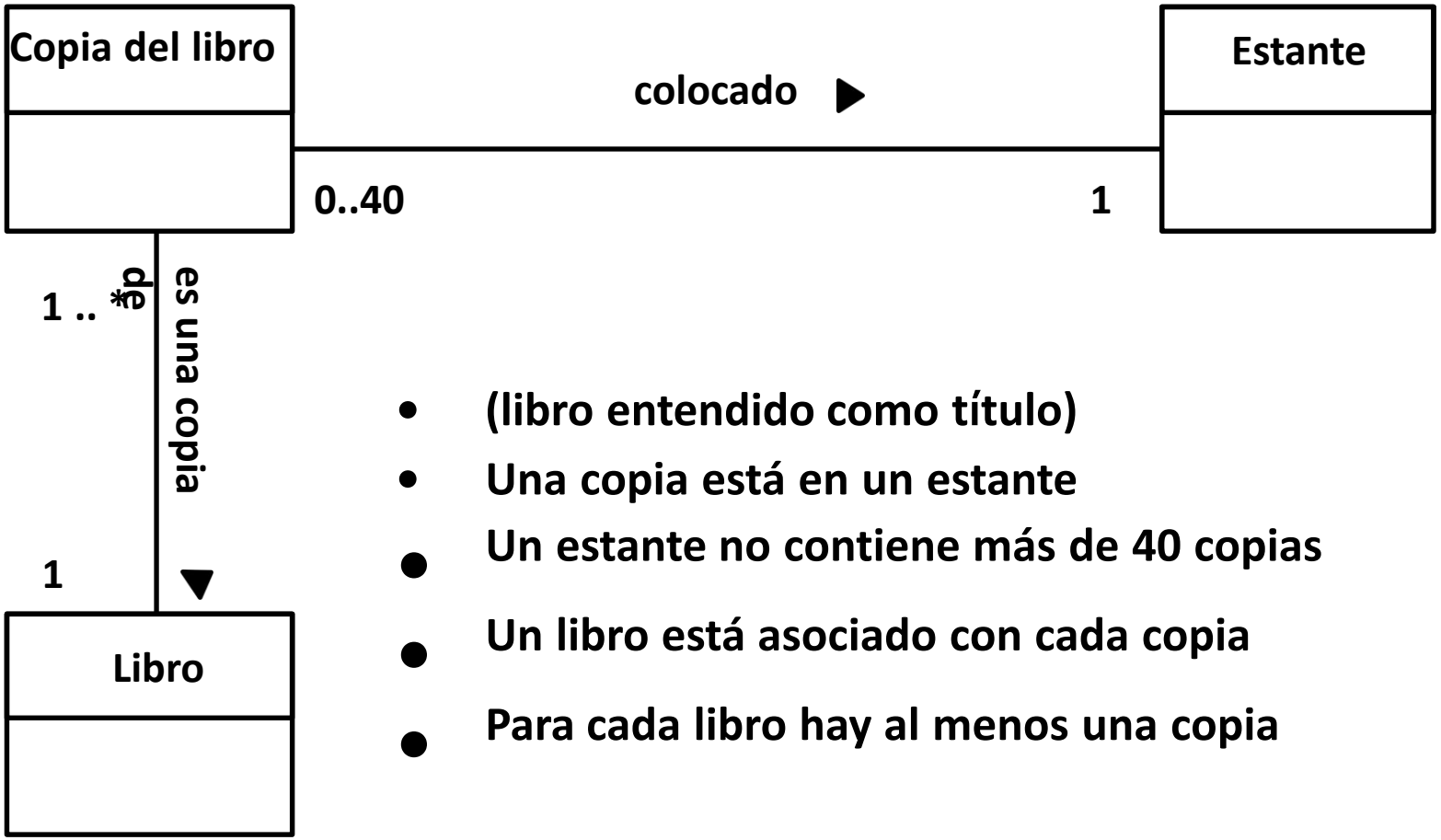
Asociaciones (cont.)

Se pueden tener más asociaciones entre dos clases



Durante la fase de análisis, una asociación *no* es una expresión acerca de flujo de datos, variables de instancia, o conexiones de objeto en la solución de software.

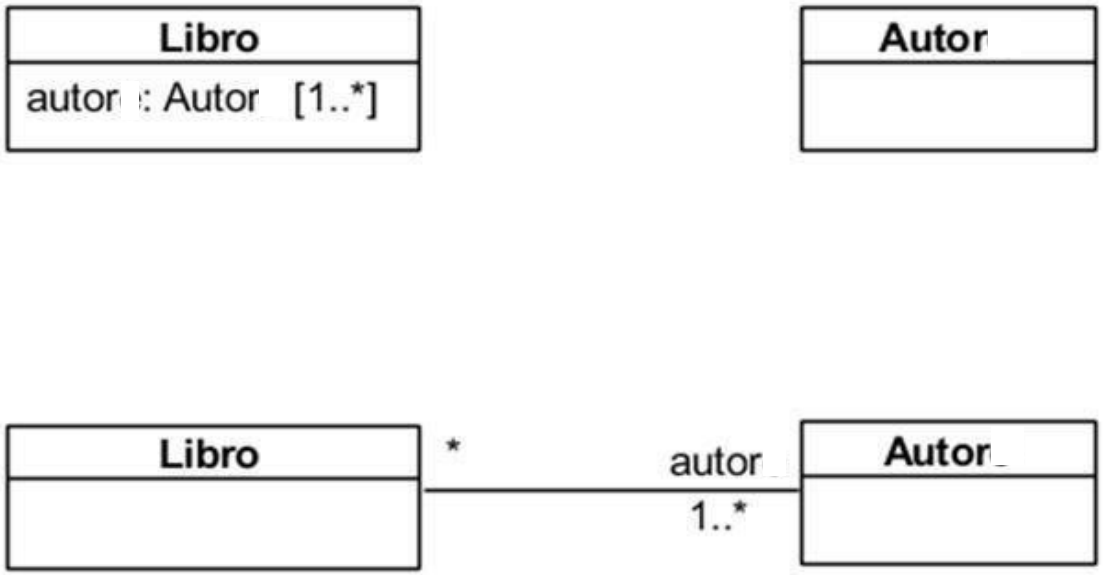
Multiplicidad: ejemplo



- (libro entendido como título)
- Una copia está en un estante
- Un estante no contiene más de 40 copias
- Un libro está asociado con cada copia
- Para cada libro hay al menos una copia

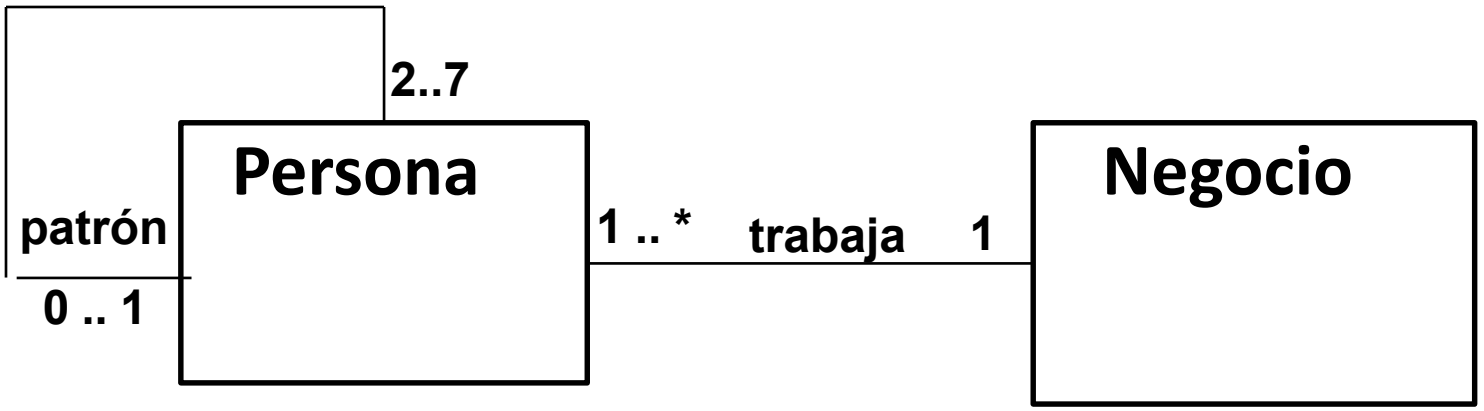
Asociaciones y atributos

- Otra forma de representar una propiedad



Asociaciones reflexivas

- En este caso es fundamental indicar el rol



Agregación y composición

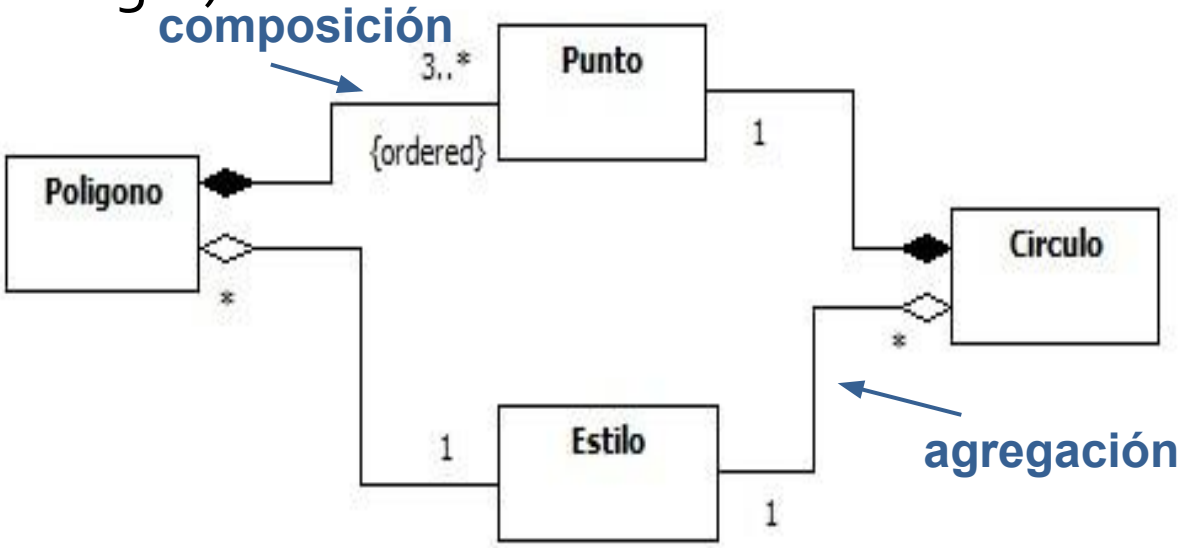
- La agregación y la composición son tipos particulares de asociación (un refinamiento)
  - Ambas especifican que un objeto de una clase es **una parte** de un objeto de otra clase
- Agregación → relación entre objetos poco fuerte

  ej. computador con los periféricos

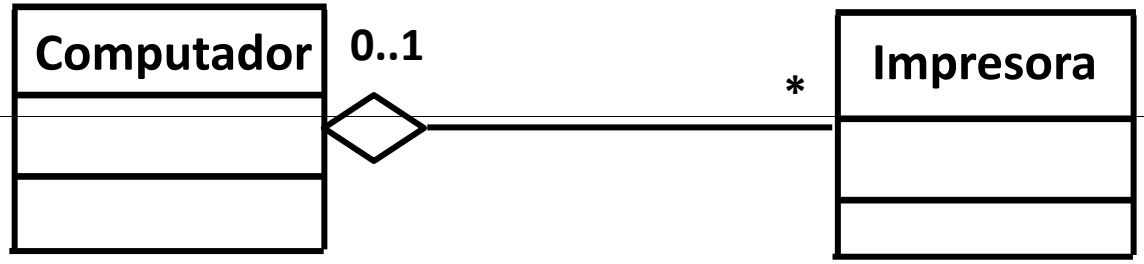
- Composición → relación entre objetos fuerte ej. árbol y sus hojas

Agregación y Composición

- agregación:
  - el estilo existe sin polígono, si el polígono es destruido el estilo sigue existiendo (simbolizado por un diamante sin relleno)
- composición:
  - las partes viven y mueren con el todo (simbolizado por un diamante negro)



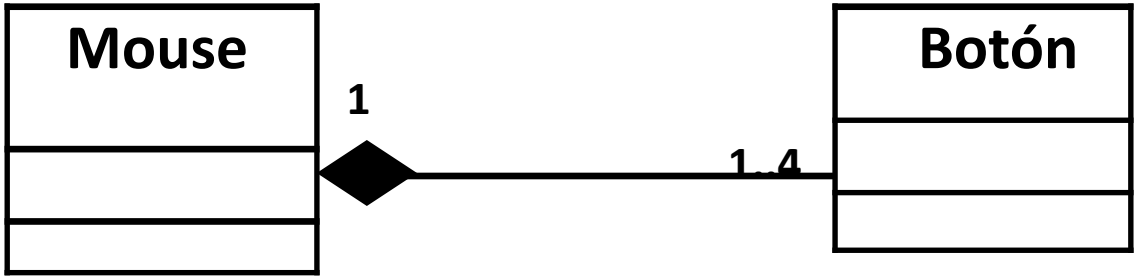
Sintaxis de la agregación



La agregación es una relación del tipo todo-parte

- Con el tiempo, la impresora se puede conectar a diferentes computadores.
- La impresora también existe sin computador
- Si el computador se destruye, la impresora aún existe

Sintaxis de la composición



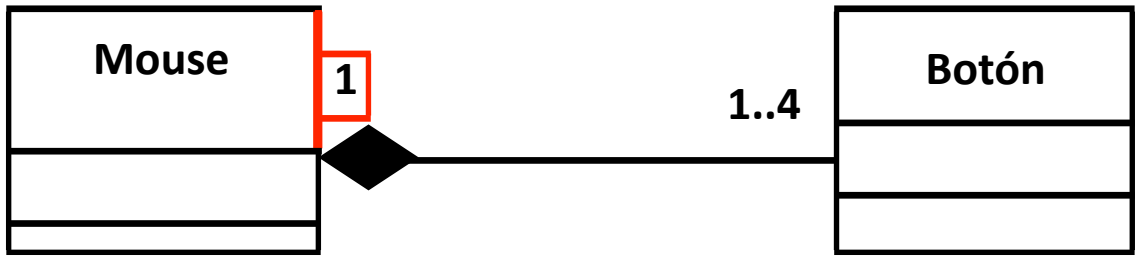
Una composición es una forma de asociación más fuerte que la agregación

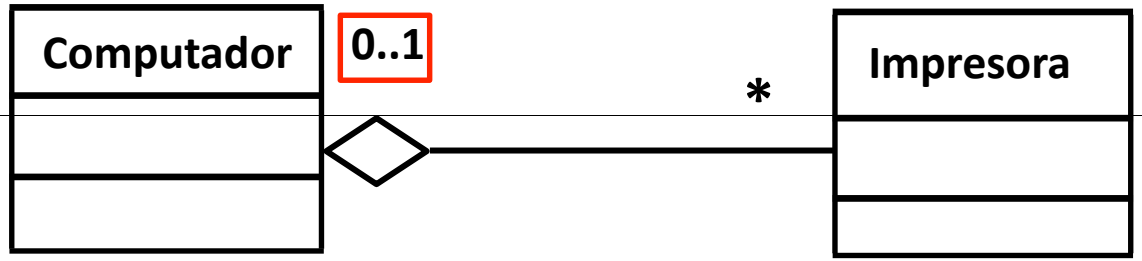
las partes no tienen significado sin el

todo

una parte pertenece a un todo

Una mirada a la multiplicidad





Encontrar Asociaciones

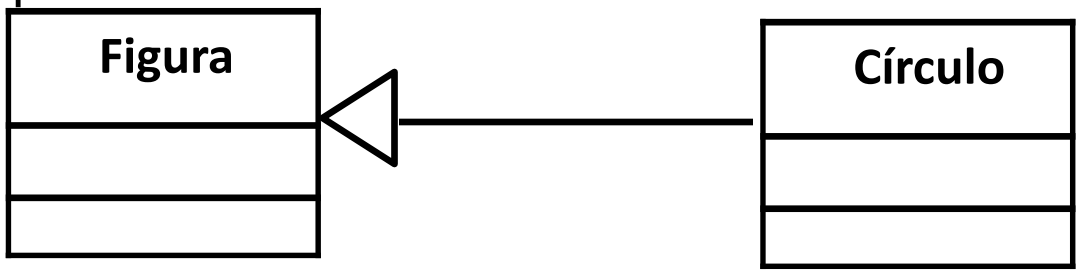
- Asociaciones de alta prioridad
  - A es una parte física o lógica de B
  - A es físicamente o logicamente contenido en B
  - A es registrado en B
- Otras asociaciones
  - A usa o maneja o controla B (Piloto -avión)
  - A posee B (Linea Aérea-avión)

Guías de Asociación

- Foco en aquellas asociaciones para las cuales se necesita que el conocimiento de la relación sea preservado por alguna duración (asociaciones que se necesitan conocer)
- Es más importante identificar conceptos que asociaciones
- Demasiadas asociaciones tienden a confundir el modelo conceptual
- Evitar mostrar asociaciones redundantes o derivables

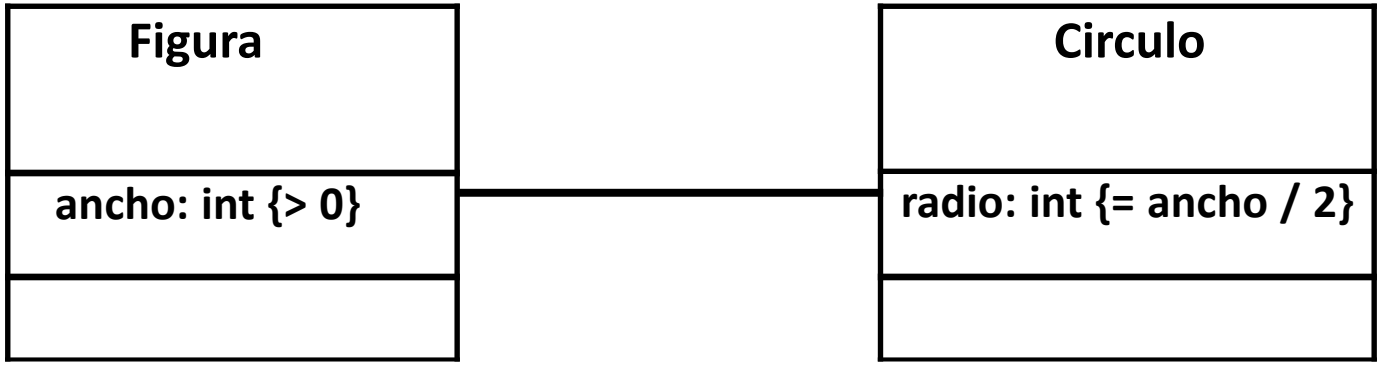
Generalización

- Relación entre un elemento genérico y uno más especializado
- El elemento más especializado es completamente consistente con el más genérico pero contiene más información
- Se aplica el principio de sustitución de Liskov: el elemento especializado se puede utilizar en lugar del elemento genérico
- "es un tipo de"



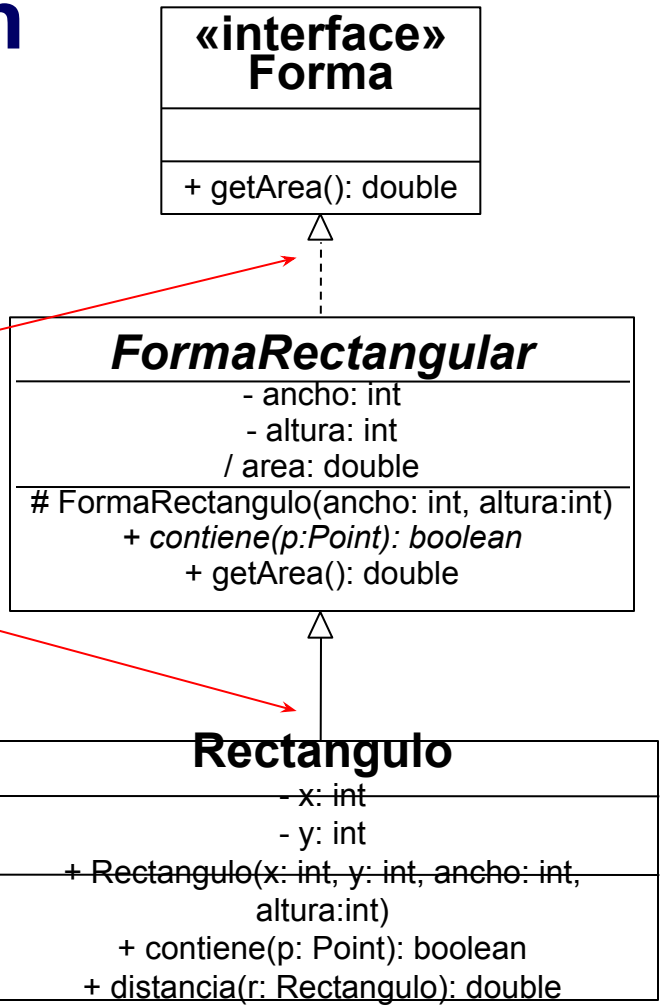
Herencia de clases

- Las subclases heredan todas las características de la superclase:
  - atributos, operaciones, relaciones y restricciones
- Las subclases pueden agregar características y redefinir operaciones

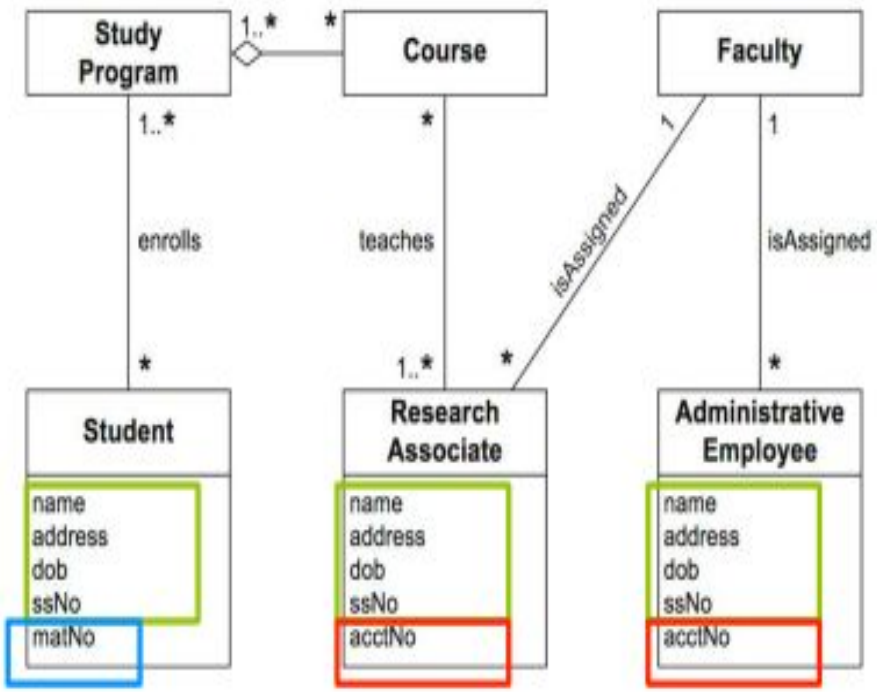


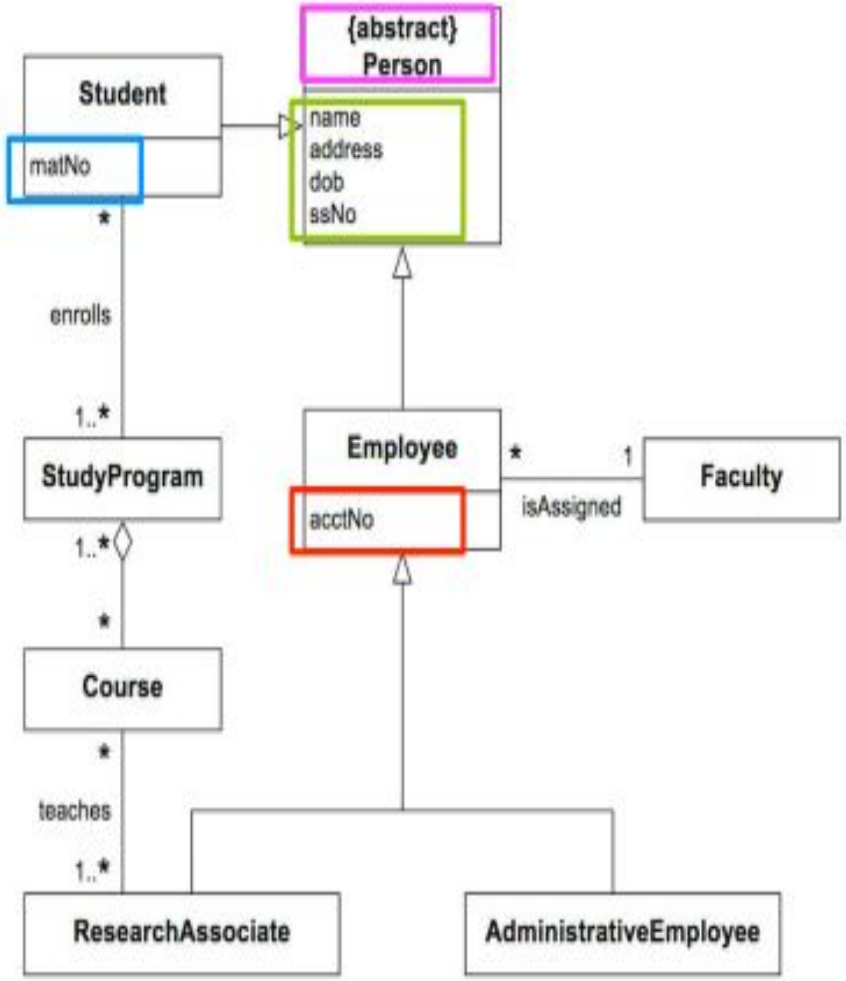
Modelo Conceptual - Relaciones de Generalización

- relaciones de generalización
  - jerarquías dibujadas con flechas apuntando hacia el padre
  - estilos de línea/flecha difieren, basados en si el padre es un(a):
    - interface (especifica / refina): línea discontinua, flecha blanca
    - clase abstracta: línea sólida, flecha blanca
    - clase (generaliza / especializa): línea sólida, flecha blanca
  - a menudo no dibujamos las relaciones de generalización banales / obvias, como dibujar la clase Object como un padre

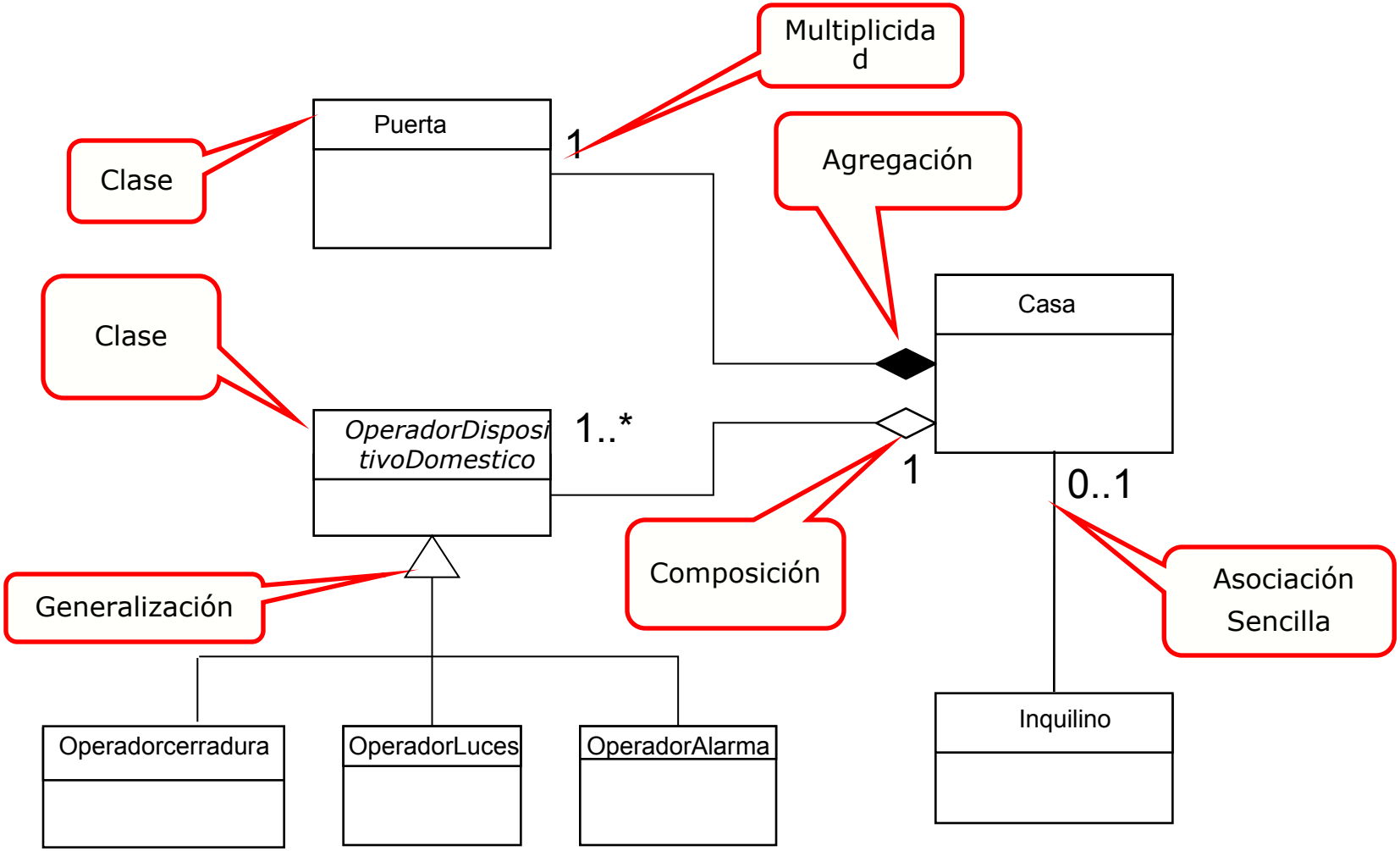


Ejemplo generalización



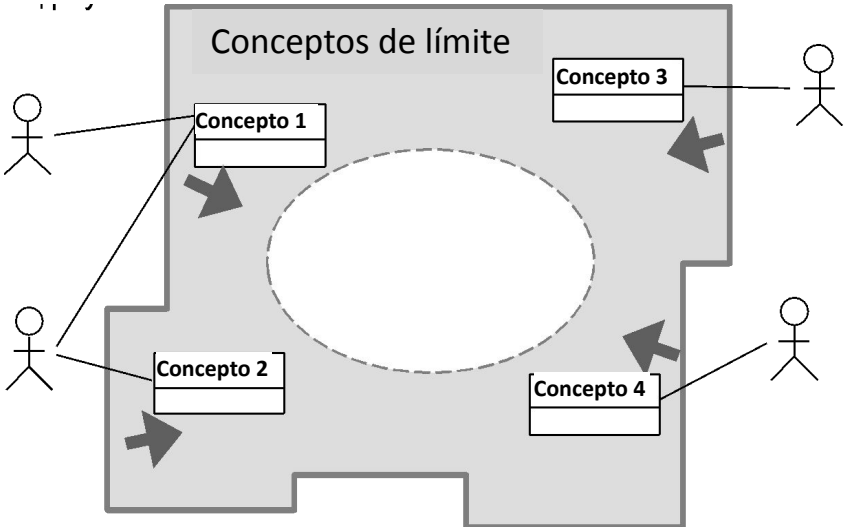


Un ejemplo de diagrama de clase

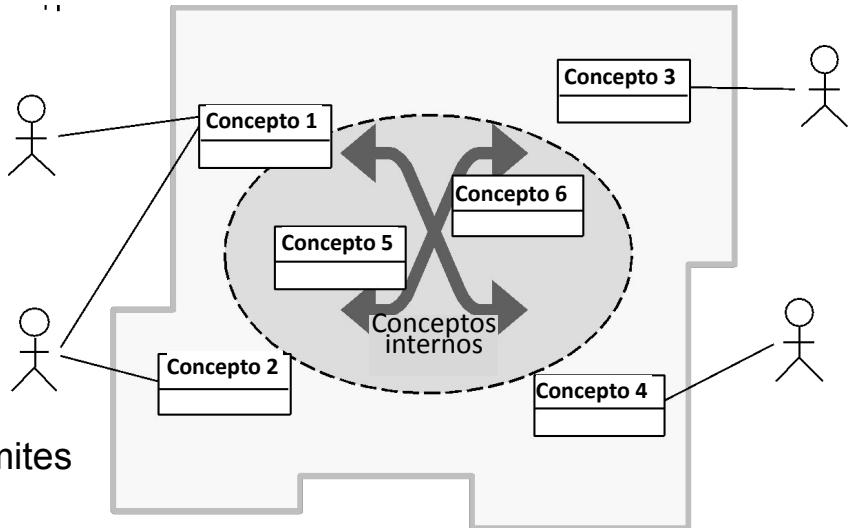


Creación de modelo de dominio a partir de casos de uso

Paso 1: Identificar los conceptos de límite



Paso 2: Identificar los conceptos internos



Paso 2: Identificar los conceptos internos

Los conceptos internos "encaminan" los datos entre los límites

- Conversión de formato de datos
- Políticas de protección de datos

Caso de uso 1: Desbloquear

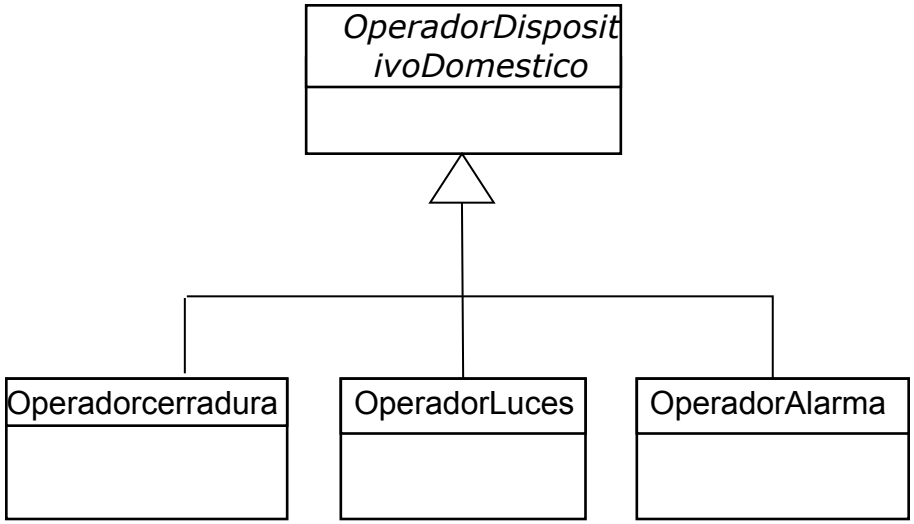
Caso de uso UC-1:	desbloquear	
Requisitos relacionados:	**REQ1, REQ3, REQ4 y REQ5 indicados en la Tabla 2-1**	
Actor iniciador:	**Cualquiera de: inquilino, arrendador**	
Objetivo del actor:	**Para desarmar la cerradura y entrar, y conseguir que el espacio se ilumine automáticamente.**	
Actores participantes:	**Dispositivo de bloqueo, interruptor de luz, temporizador**	
Precondiciones:	**• El conjunto de claves válidas almacenadas en la base de datos del sistema no está vacío.** **• El sistema muestra el menú de funciones disponibles; en el teclado de la puerta, las opciones del menú son "Bloquear" y "Desbloquear".**	
Postcondiciones:	**El temporizador de bloqueo automático ha iniciado la cuenta atrás desde autoLockInterval.**	
Flujo de eventos para el escenario principal de éxito:		
→	**1.**	**El inquilino / propietario llega a la puerta y selecciona el elemento del menú "Desbloquear"**
	2.	**incluir::*AuthenticateUser* (UC-7)**
←	**3.**	**El sistema (a) le indica al inquilino / arrendador el estado de la cerradura, por ejemplo, "desarmado", (b) le indica al LockDevice que desarme la cerradura y (c) le indica a LightSwitch que encienda la luz**
←	**4.**	**El sistema envía una señal al temporizador para iniciar la cuenta atrás del temporizador de bloqueo automático**
→	**5.**	**El inquilino / propietario abre la puerta, entra a la casa [y cierra la puerta y bloquea]**

Extracción de responsabilidades

Descripción de Responsabilidad	Tipo	Nombre del concepto
Coordine las acciones de todos los conceptos asociados con un caso de uso, una agrupación lógica de casos de uso o todo el sistema y delegue el trabajo a otros conceptos.	D	Controlador
Contenedor para datos de autenticación del usuario, como contraseña, sello de tiempo, identificación de puerta, etc.	K	Clave
Verifique si el código clave ingresado por el usuario es válido.	D	KeyChecker
Contenedor para la recogida de claves válidas asociadas a puertas y usuarios.	K	Almacenamiento de claves
Opere el dispositivo de bloqueo en posiciones de armado / desarmado.	D	LockOperator
Opere el interruptor de luz para encender / apagar la luz.	D	LightOperator
Opere la campana de alarma para señalar posibles robos.	D	AlarmOperator
Bloquee la entrada para denegar más intentos si hay demasiados intentos fallidos.	D	Controlador
Registre todas las interacciones con el sistema en almacenamiento persistente.	D	Logger

Grados de Refinamiento del modelo de dominio

- Caso más simple: todos los dispositivos domésticos son conceptualmente iguales, solo un interruptor de encendido / apagado para activar o desactivar
- Si cada dispositivo proporcionará una funcionalidad adicional, use diferentes objetos conceptuales
- El enfoque correcto depende de los requisitos



Caso de uso 5: inspeccionar el historial de acceso

Caso de uso UC-5:		Inspeccionar el historial de acceso
Requisitos relacionados:		REQ8 y REQ9 indicados en la Tabla 2-1
Actor iniciador:		Cualquiera de: inquilino, arrendador
Objetivo del actor:		Examinar el historial de acceso de una puerta en particular.
Actores participantes:		Base de datos, propietario
Precondiciones:		El inquilino / propietario está actualmente registrado en el sistema y se muestra un hipervínculo "Ver historial de acceso".
Postcondiciones:		Ninguna.
Flujo de eventos para el escenario principal de éxito:		
→	1.	El inquilino / propietario hace clic en el hipervínculo "Ver historial de acceso"
←	2.	El sistema solicita los criterios de búsqueda (p. Ej., Período de tiempo, ubicación de la puerta, función del actor, tipo de evento, etc.) o "Mostrar todo"
→	3.	El inquilino / propietario especifica los criterios de búsqueda y envía
←	4.	El sistema prepara una consulta de base de datos que mejor se adapta a los criterios de búsqueda del actor y recupera los registros de la base de datos.
→	5.	La base de datos devuelve los registros coincidentes
←↵	6.	El sistema (a) filtra además los registros recuperados para que coincidan con los criterios de búsqueda del actor; (b) presenta los registros restantes para su visualización; y (c) muestra el resultado para la consideración del inquilino / propietario
→	7.	El inquilino / propietario busca, selecciona registros "interesantes" (si los hay) y solicita una investigación adicional (con una descripción de la queja adjunta)
←	8.	El sistema (a) muestra solo los registros seleccionados y confirma la solicitud; (b) archiva la solicitud en la base de datos y le asigna un número de seguimiento; (c) notifica al propietario sobre la solicitud; e (d) informa al inquilino / arrendador sobre el número de seguimiento

Extracción de responsabilidades

Descripción de Responsabilidad	Tipo	Nombre del concepto
Rs1. Coordinar acciones de conceptos asociados con este caso de uso y delegar el trabajo a otros conceptos.	**D**	**Controlador**
Rs2. Formulario que especifica los parámetros de búsqueda para la recuperación del registro de la base de datos (de UC-5, Paso 2).	**K**	**Solicitud de búsqueda**
Rs3. Procese los registros recuperados en un documento HTML para enviar al navegador web del actor para su visualización.	**D**	**Creador de páginas**
Rs4. Documento HTML que muestra al actor el contexto actual, las acciones que se pueden realizar y los resultados de las acciones anteriores.	**K**	**Página de interfaz**
Rs5. Prepare una consulta de base de datos que se adapte mejor a los criterios de búsqueda del actor y recupere los registros de la base de datos (de UC-5, Paso 4).	**D**	**Conexión de base de datos**
Rs6. Filtre los registros recuperados para que coincidan con los criterios de búsqueda del actor (de UC-5, Paso 6).	**D**	**Postprocesador**
Rs7. Lista de registros "interesantes" para mayor investigación, descripción de la queja y número de seguimiento.	**K**	**Solicitud de investigación**
Rs8. Archive la solicitud en la base de datos y asígnele un número de seguimiento (de UC-5, Paso 8).	**D**	**Archivador**
Rs9. Notifique al propietario sobre la solicitud (de UC-5, Paso 8).	**D**	**Notificador**

Extrayendo las asociaciones

Par de conceptos	Descripción de la asociación	Nombre de la asociación
Controlador ↔ Creador de páginas	El controlador pasa las solicitudes a Page Maker y recibe las páginas posteriores preparadas para mostrar	transmite solicitudes
Creador de páginas ↔ Conexión de base de datos	La conexión de la base de datos pasa los datos recuperados a Page Maker para representarlos para su visualización	proporciona datos
Creador de páginas ↔ Página de interfaz	Page Maker prepara la página de interfaz	prepara
Controlador ↔ Conexión de base de datos	El controlador pasa las solicitudes de búsqueda a la conexión de la base de datos	transmite solicitudes
Controlador ↔ Archivador	El controlador pasa una lista de registros "interesantes" y una descripción de la queja al Archiver, que asigna el número de seguimiento y crea una solicitud de investigación.	transmite solicitudes
Archivador ↔ Solicitud de investigación	Archiver genera una solicitud de investigación	genera
Archivador ↔ Conexión de base de datos	Archiver solicita la conexión a la base de datos para almacenar las solicitudes de investigación en la base de datos	solicitudes guardar
Archivador ↔ Notificador	Archiver solicita al notificador que notifique al propietario sobre las solicitudes de investigación	solicitudes notificar

Extrayendo los atributos

Concepto	Atributos	Descripción del atributo
Solicitud de búsqueda	identidad del usuario	Se utiliza para determinar las credenciales del actor, que a su vez especifican qué tipo de datos está autorizado a ver este actor.
	parámetros de búsqueda	Marco de tiempo, papel del actor, ubicación de la puerta, tipo de evento (desbloqueo, bloqueo, corte de energía, etc.).
Postprocesador	parámetros de búsqueda	Copiado de la solicitud de búsqueda; necesario para filtrar los registros recuperados para que coincidan con los criterios de búsqueda del actor.
Solicitud de investigación	lista de registros	Lista de registros "interesantes" seleccionados para una mayor investigación.
	descripción de la queja	Describe las sospechas del actor sobre los registros de acceso seleccionados.
	el número de rastreo	Permite el seguimiento del estado de la investigación.
Archivador	número de seguimiento actual	Necesario para asignar un número de seguimiento a las quejas y solicitudes.
Notificador	Información del contacto	Información de contacto del propietario que acepta quejas y solicitudes de investigación adicional.

Matriz de trazabilidad (1)

Mapeo: requisitos del sistema para casos de uso

REQ1: Mantenga la puerta bloqueada y con bloqueo automático
REQ2: Bloquear cuando se presiona "LOCK"
REQ3: Desbloquear cuando se proporcione una clave válida
REQ4: Permitir errores pero prevenir ataques de diccionario
REQ5: Mantener un registro de historial
REQ6: Agregar / eliminar usuarios en tiempo de ejecución
REQ7: Configuración de las preferencias de activación del dispositivo
REQ8: Inspección del historial de acceso
REQ9: Presentación de consultas

UC1: Desbloquear
UC2: bloqueo
UC3: AddUser
UC4: Eliminar usuario
UC5: InspectAccessHistory
UC6: SetDevicePrefs
UC7: AuthenticateUser
UC8: Iniciar sesión

Req't	PW	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8
REQ1	5	X	X						
REQ2	2		X						
REQ3	5	X						X	
REQ4	4	X						X	
REQ5	2	X	X						
REQ6	1			X	X				X
REQ7	2						X		X
REQ8	1					X			X
REQ9	1					X			X
Max PW		5	2	2	2	1	5	2	1
Total PW		15	3	2	2	3	9	2	3

Matriz de trazabilidad (2)

Mapeo: casos de uso para el modelo de dominio

UC1: Desbloquear
UC2: bloqueo
UC3: AddUser
UC4: Eliminar usuario
UC5: InspectAccessHistory
UC6: SetDevicePrefs
UC7: AuthenticateUser
UC8: Iniciar sesión

		Domain Concepts														
Use Case	PW	Controller-SS1	StatusDisplay	KeycodeEntry	Key	KeyStorage	KeyChecker	HouseholdDeviceOperator	Controller-SS2	SearchRequest	InterfacePage	PageMaker	Archiver	DatabaseConnection	Notifier	InvestigationRequest
UC1	15	X	X	X	X			X								
UC2	3	X	X					X								
UC3	2								X		X	X		X		
UC4	2								X		X	X		X		
UC5	3								X	X	X	X	X	X	X	X
UC6	9								X		X	X		X		
UC7	2	X	X		X	X	X									
UC8	3								X		X	X		X		

Contratos: Precondiciones y postcondiciones

Operación	desbloquear
Precondiciones	• el conjunto de claves válidas conocidas por el sistema no está vacío
	• numOfAttempts ≤ maxNumOfAttempts
	• numOfAttempts = 0, para el primer intento del usuario actual
Postcondiciones	• numOfAttempts = 0, si la clave ingresada ∈ Claves válidas
	• La instancia actual del objeto Key se archiva y se destruye.

Operación	Cerrar con clave
Precondiciones	Ninguno (es decir, ninguno digno de mención)
Postcondiciones	• lockStatus = "armado", y
	• lightStatus permanece sin cambios (ver texto para discusión)


Universidad de
Playa Ancha
www.upla.cl