UNIVERSIDAD DE LIMA | 60 años

FACULTAD DE INGENIERÍA Y ARQUITECTURA
CARRERA DE INGENIERÍA DE SISTEMAS

UNIDAD 3:
Análisis de requisitos orientado a objetos
Modelado en UML

INGENIERIA DE SOFTWARE I

2022| 2022-1

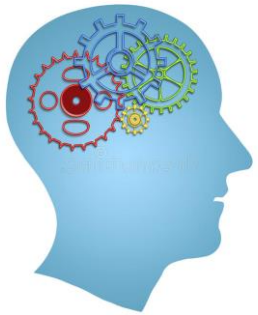
Agenda

- Modelado UML
- Diagrama de Caso de Uso de Sistema
- Diagrama de Paquetes
- Diagrama Conceptual
- Diagrama de Clases

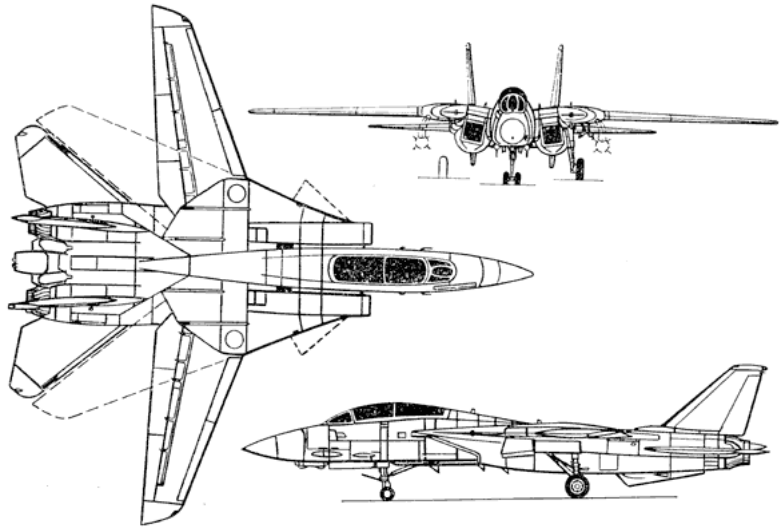
¿ Que es Modelar ?

Mundo Real

…Es el proceso de "generar un modelo".



Es un proceso intelectual en el cual se representa, a través de un modelo ciertas **características o cualidades de un objeto.**



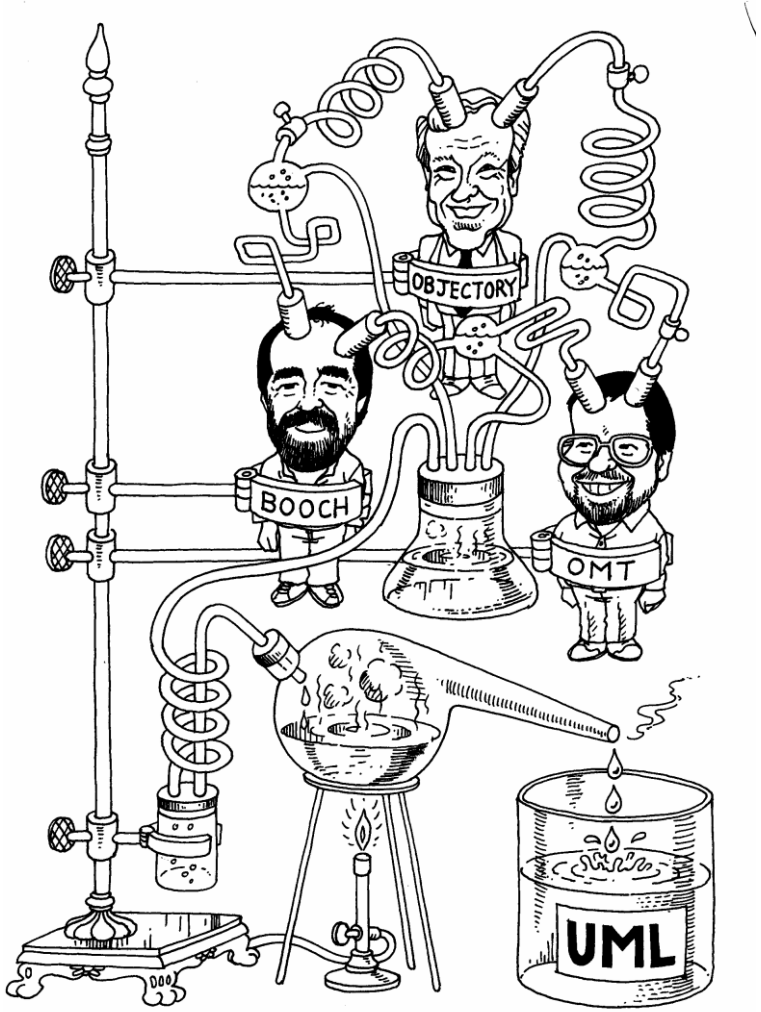
¿Qué es un modelo?

El modelo es una ***representación abstracta***, conceptual o formal, de un objeto, fenómeno o proceso de alguna parte del ***mundo real.***

UML

El camino hacia la unificación

Tres corrientes



Modelado Orientado a Objetos con UML

- UML es un lenguaje estándar para escribir planos de software.
- UML es utilizado para expresar gráficamente el proceso de generación de software.
- UML es independiente del lenguaje de implementación del software.

Lenguaje : Proporciona la sintaxis, vocabulario y las

Modelado : El UML es visual.

Unificado : Unifica varias técnicas de modelado en una única.

Uso de UML

UML es un de Modelado para:

- **Visualizar** : Representar y Comunicar Ideas.
- **Especificar** : Modelos precisos, no ambiguos, completos.
- **Construir** : Trasladar en forma directa a un lenguaje de programación.
- **Documentar**: Los artefactos construidos durante un proyecto.

Presenta diferentes vistas de un sistema.

Vistas

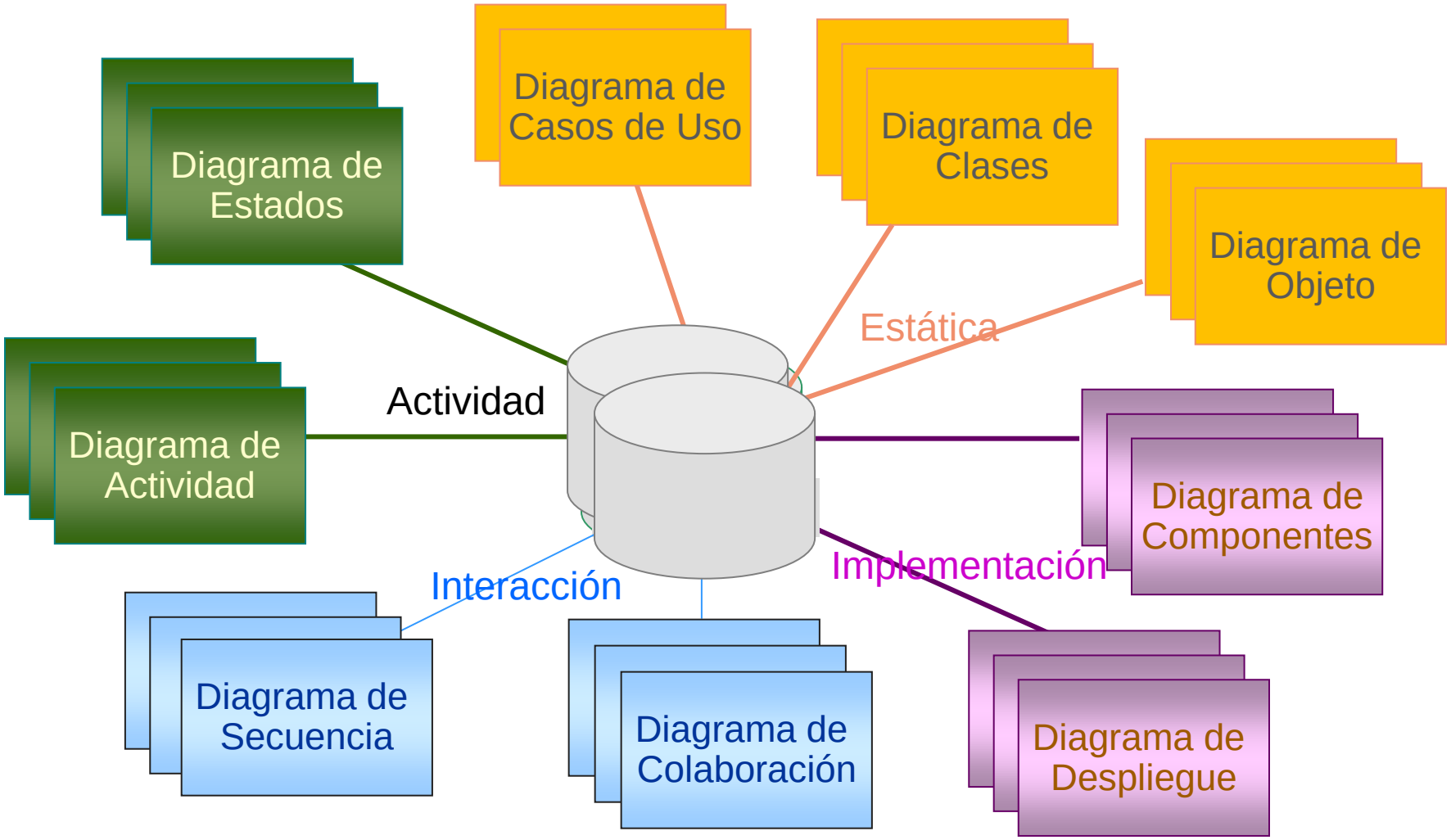


Diagrama de Estados
Diagrama de Casos de Uso
Diagrama de Clases
Diagrama de Objeto
Estática
Actividad
Diagrama de Actividad
Diagrama de Componentes
Implementación
Interacción
Diagrama de Secuencia
Diagrama de Colaboración
Diagrama de Despliegue

Vista 4+1


Vista de diseño
Vista de implementación
Vista de casos de uso
Vista de procesos
Vista de despliegue

Vista 4+1

Vista de Casos de Uso: Engloba los Casos de Uso que describen el comportamiento del sistema como lo verían los usuarios finales, los analistas y demás componentes del equipo de desarrollo. No especifica la organización del sistema. Con UML los **aspectos estáticos** de esta vista se pueden concretar con los diagramas de *Casos de Uso*; los **aspectos dinámicos** con los diagramas de iteración (*secuencia* y *colaboración*), diagramas de *estados* y de *actividades*.

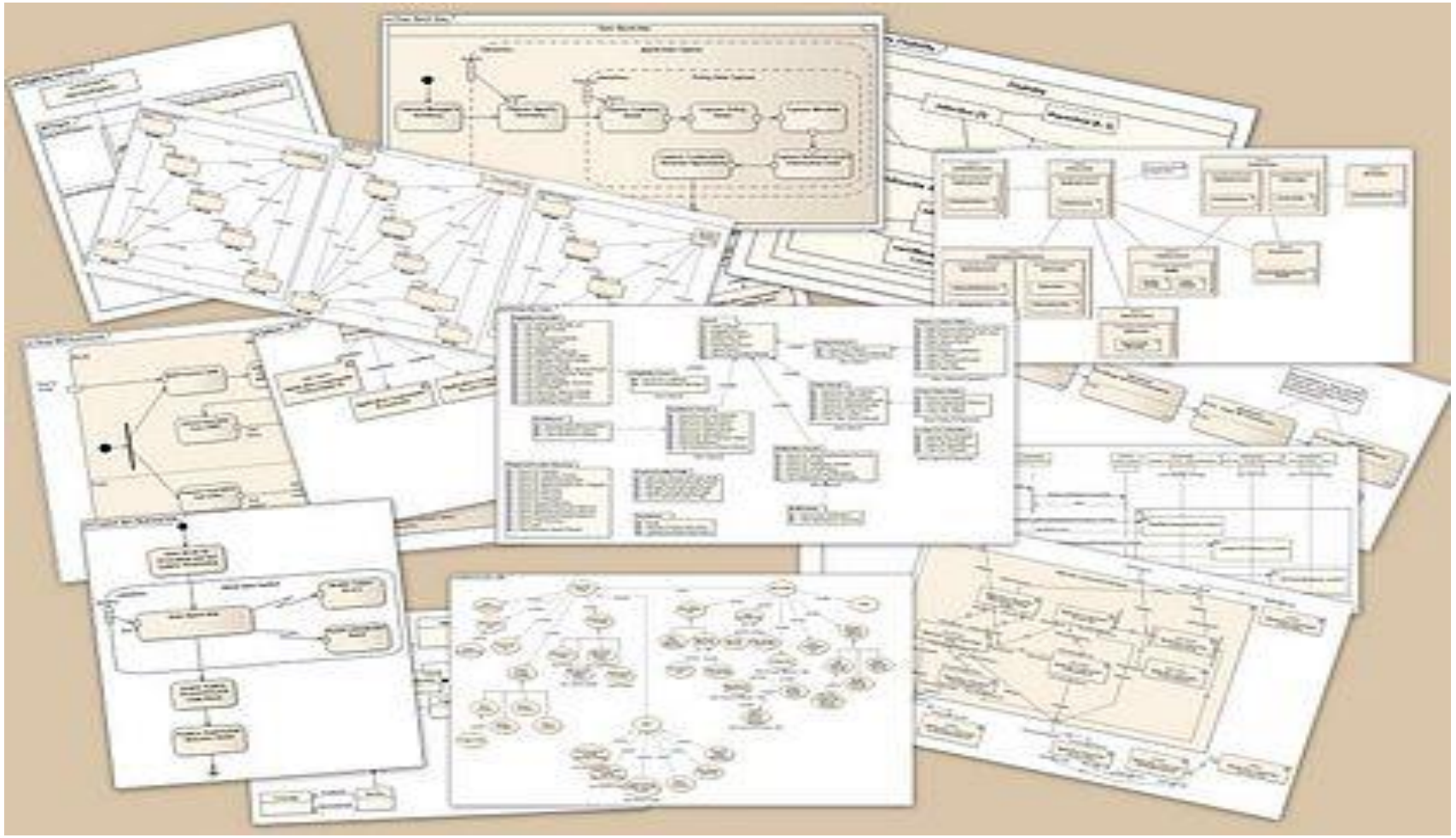
Vista de Diseño: Engloba las clases e interfaces que conforman el vocabulario del problema y su solución. Da soporte a los requisitos funcionales del sistema, es decir los servicios que proporciona a los usuarios finales. Con UML los **aspectos estáticos** de esta vista se pueden concretar con los diagramas de *clases* y de *objetos*; los **aspectos dinámicos** con los diagramas de iteración (*secuencia* y *colaboración*), diagramas de *estados* y de *actividades*.

Vista de Procesos: Engloba los hilos y procesos que forman los mecanismos de sincronización y concurrencia del sistema. Da soporte al funcionamiento, capacidad de crecimiento y rendimiento del sistema. Con UML los **aspectos estáticos** de esta vista se pueden concretar con los diagramas de *clases*, de *clases activas* y de *objetos*; los **aspectos dinámicos** con los diagramas de iteración (*secuencia* y *colaboración*), diagramas de *estados* y de *actividades*.

Vista de Implementación: Engloba los componentes y archivos empleados para hacer posible el sistema físico. Da soporte a la gestión de configuraciones de las distintas versiones del sistema, a partir de componentes y archivos. Con UML los **aspectos estáticos** de esta vista se pueden concretar con los diagramas de *componentes*; los aspectos dinámicos con los diagramas de iteración (*secuencia* y *colaboración*), diagramas de *estados* y de *actividades*.

Vista de Despliegue: Engloba los nodos que forman la topología hardware sobre el que se ejecuta el sistema. Da soporte a la distribución, entrega e instalación de las partes que conforman el sistema físico. Con UML los **aspectos estáticos** de esta vista se pueden concretar con los diagramas *despliegue*; los **aspectos dinámicos** con los diagramas de iteración (*secuencia* y *colaboración*), diagramas de *estados* y de *actividades*.

Diagramas de UML



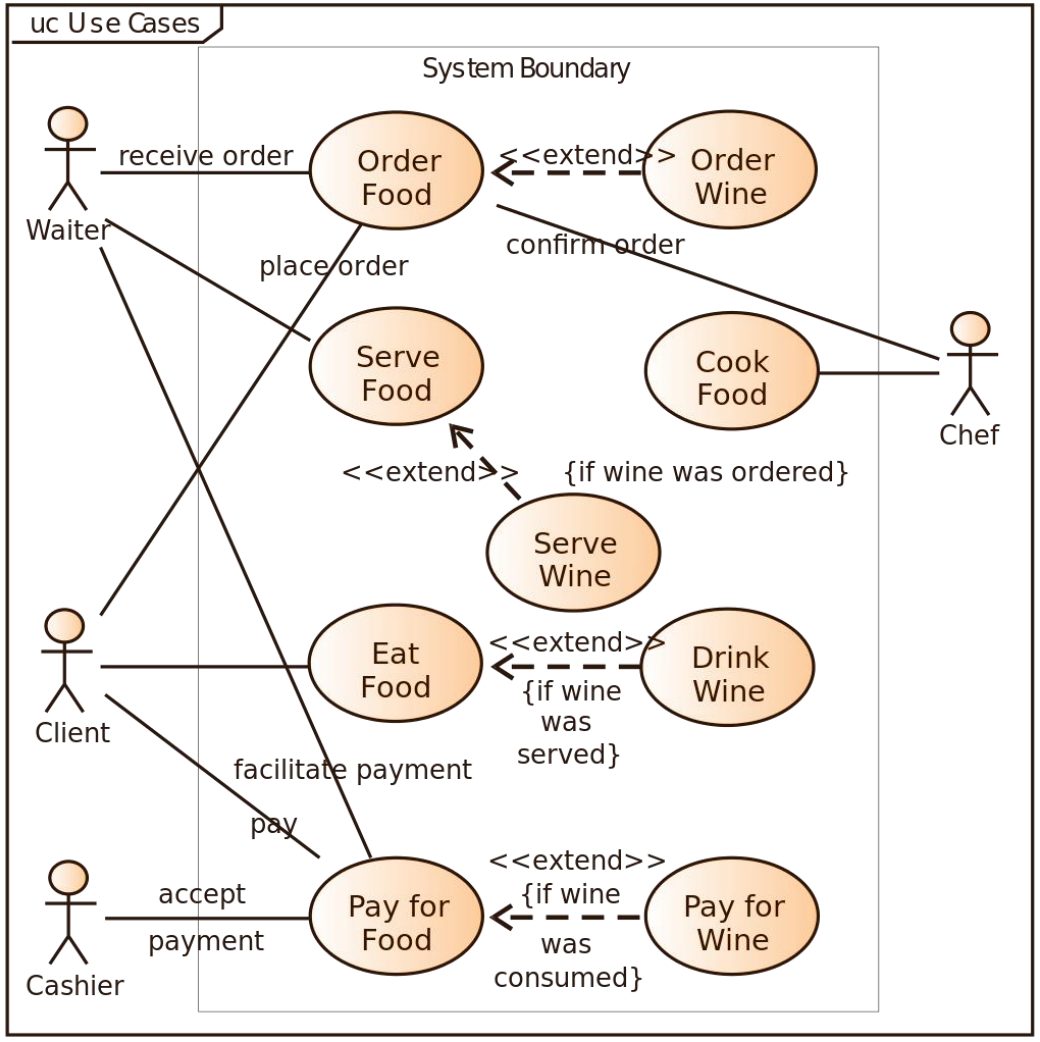


Diagrama de Caso de Uso de Sistema

Modelado de Casos de Uso

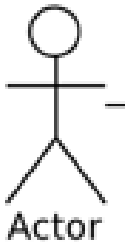
- Especifica un comportamiento deseado del sistema.Representan los **requisitos funcionales** del sistema.

  *"Un caso de uso especifica un **conjunto de secuencias de acciones**, incluyendo **variantes**, que el **sistema puede ejecutar** y que produce un **resultado observable** de **valor para un particular actor."***

- Describen ***qué*** hace el sistema, **no *cómo*** lo hace.

Actor de Sistema

Un actor representa el rol jugado por una persona o cosa que actúa con el sistema.



Dos tipos de actores:

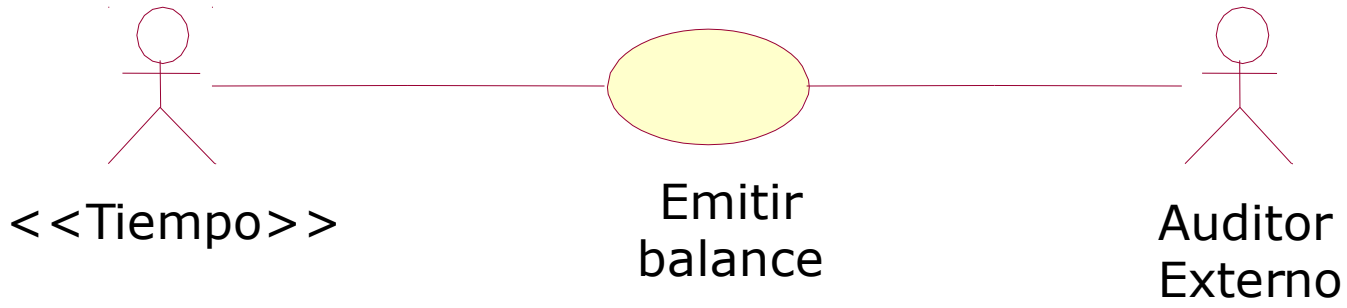
Principal: Requiere al sistema el cumplimiento de un objetivo.

Secundarios: El sistema necesita de ellos para satisfacer un objetivo.

Ejemplos: "Cliente, Administrador, Usuario no Registrado (Autenticado), Usuario Registrado (Autenticado), Jefe de Compras, Jefe de Personal, Moderador, Jefe de Departamento, Obrero de Planta, Supervisor..."

¿Quienes son Actores?

- Roles jugados por personas,
- dispositivos
- otros sistemas
- El tiempo



Casos de Uso

Los casos de uso son de vital importancia en los proyectos de software (*Procesos Guiados por Casos de Uso*)

- Se describen bajo la forma de ***acciones y reacciones*** el comportamiento de un sistema ***desde el punto*** *de* ***vista de un usuario***

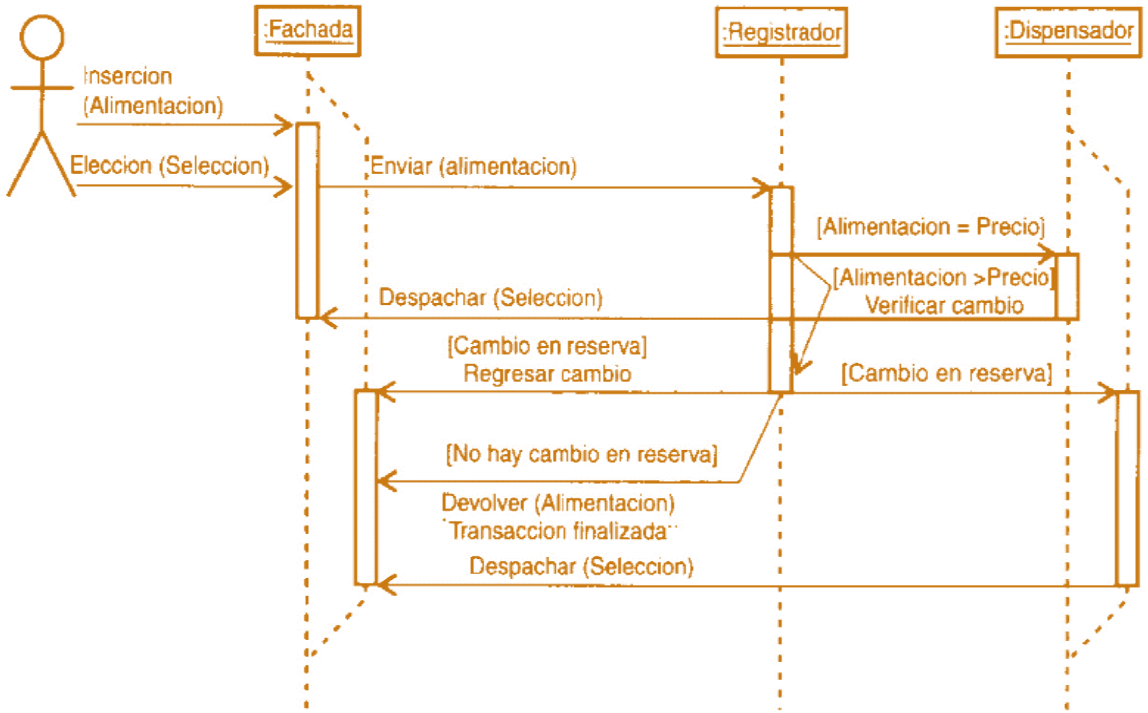
- Permiten **definir los límites**

Propiedades de los casos de uso

- Son iniciados por un actor
- Puede incluir secuencias **alternativas**
- El sistema es considerado como una **"caja negra**
- **El conjunto completo** de casos de uso especifica todas las posibles formas de usar el sistema.

Escenario

Es una **secuencia** de ***acciones*** e interacciones (pasos) entre los usuarios (*actores*) y el sistema



Identificando los Casos de Uso

En UML los Casos de Uso son la vía con cual se **capturan los requerimientos** de los usuarios.

- Los Casos de Uso determinan todo lo que un actor desea hacer con el sistema.
  - Ejecutar las tareas del sistema.
  - Leer, escribir y cambiar información del sistema.
  - Informar al sistema de los cambios del mundo real.
  - La necesidad de ser informado acerca de los cambios del sistema.

Especificación de un caso de uso

- Texto estructurado informal
- Texto estructurado formal (**plantillas**)
- Pseudocódigo
- Notaciones gráficas: **diagramas de secuencia**

Descripción Textual de un Caso de Uso

Nombre:	*<nombre del caso de uso>*
Autor:	*<nombre del autor (o autores) del caso de uso>*
Fecha:	*<fecha de creación del caso de uso>*
Descripción: *<breve descripción del caso de uso>*	
Actores: *<actores participantes en el caso de uso>*	
Precondiciones: *<condiciones que deben cumplirse para poder ejecutar el caso de uso>*	
Flujo Normal: *<flujo normal (feliz) de ejecución del caso de uso>*	
Flujo Alternativo: *<flujos alternativos de ejecución del caso de uso>*	
Poscondiciones: *<condiciones que deben cumplirse al finalizar la ejecución del caso de uso>*	

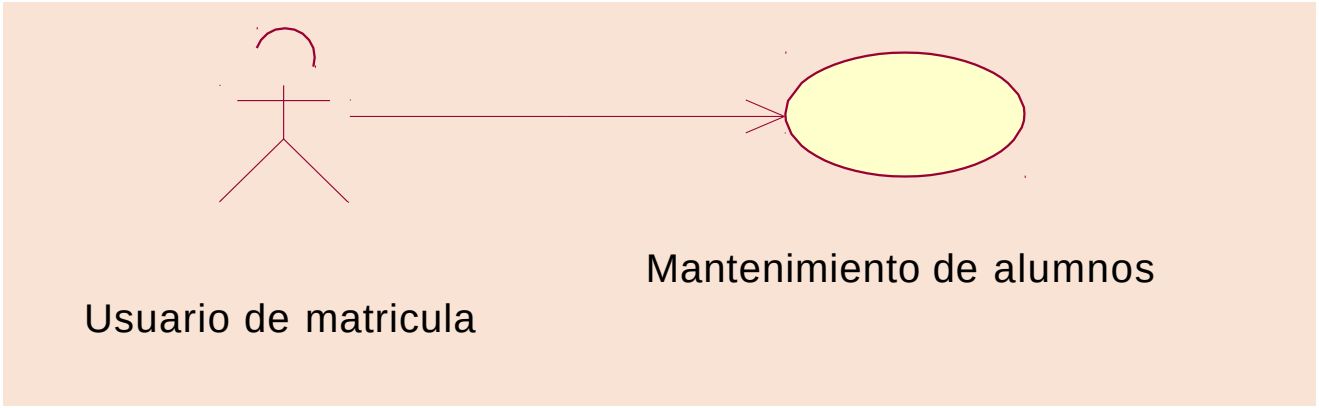
Descripción Textual de un Caso de Uso

Nombre:	Crear mensaje foro
Autor:	Pedro Pérez
Fecha:	21/04/09
Descripción: Permite crear un nuevo mensaje (hilo) en el foro de discusión.	
Actores: Usuario / Moderador	
Precondiciones: El usuario debe de estar autenticado en el sistema.	
Flujo Normal: 1.- El actor pulsa sobre el botón para crear un nuevo mensaje. 2.- El sistema muestra una caja de texto para introducir el título del mensaje y una zona de mayor tamaño para introducir el cuerpo del mensaje. 3.- El actor introduce el título del mensaje y el cuerpo del mismo. 4.- El sistema comprueba la validez de los datos y los almacena. 5.- El moderador recibe una notificación de que hay un nuevo mensaje. 6.- El moderador acepta y el sistema publica el mensaje si éste fue aceptado por el moderador.	
Flujo Alternativo: 4.A.- El sistema comprueba la validez de los datos, si los datos no son correctos, se avisa al actor de ello permitiéndole que los corrija. 7.B.- El moderador rechaza el mensaje, de modo que no es publicado sino devuelto al usuario.	
Poscondiciones: El mensaje ha sido almacenado en el sistema y fue publicado.	

Descripción Textual de un Caso de Uso

Flujo Normal: 1.- El actor pulsa sobre el botón para crear un nuevo mensaje. 2.- El sistema muestra una caja de texto para introducir el título del mensaje y una zona de mayor tamaño para introducir el cuerpo del mensaje. 3.- El actor introduce el título del mensaje y el cuerpo del mismo. 4.- El sistema comprueba la validez de los datos y los almacena. 5.- El moderador recibe una notificación de que hay un nuevo mensaje. 6.- El moderador acepta y el sistema publica el mensaje si éste fue aceptado por el moderador.
Flujo Alternativo: 4.A.- El sistema comprueba la validez de los datos, si los datos no son correctos, se avisa al actor de ello permitiéndole que los corrija. 7.B.- El moderador rechaza el mensaje, de modo que no es publicado sino devuelto al usuario.
Poscondiciones: El mensaje ha sido almacenado en el sistema y fue publicado.

Ejemplo



ID: UC576

Caso de Uso: **Mantenimiento de Alumnos**

Actor: Usuario de Matrícula

Descripción: Este caso de uso permite registrar la información de los alumnos. Esto incluye las opciones de agregar, modificar y eliminar los alumnos del sistema.

Precondición: El usuario de Matrícula debe haber ingresado al sistema.

Flujo de Eventos:

1. El usuario selecciona la opción agregar alumno.
2. El sistema muestra un formulario de ingreso de alumnos en blanco.
3. El usuario ingresa la siguiente información del estudiante: nombre, fecha de nacimiento, dirección, teléfono, estado.
4. El sistema **valida la información** ingresada para asegurar el adecuado formato y busca si existe algún alumno con el mismo nombre.
5. El sistema crea un nuevo alumno y le asigna un código único del sistema.
6. Los pasos 2 al 5 son repetidos para cada alumno por agregar al sistema. Cuando el usuario haya finalizado de agregar nuevos alumnos el caso de uso termina.

Post condición: El alumno queda registrado exitosamente

Flujo Alterno "Modificar Alumno":

1. El usuario selecciona la opción modificar alumno.
2. El sistema muestra un formulario donde ingresar el Id del alumno a modificar.
3. El usuario ingresa el Id del alumno que desea modificar.
4. El sistema obtiene la información del alumno y la muestra en un formulario.
5. El usuario modifica uno o más datos del alumno.
6. Cuando los cambios estén completos el usuario selecciona grabar.
7. El sistema actualiza la información del alumno.
8. Los pasos del 2 al 7 son repetidos por cada alumno que el usuario desee modificar. Cuando las modificaciones estén completas el caso de uso termina.

Post condición: Los datos del alumno quedan actualizados.

Flujo Alterno "Eliminar Alumno":

1. El usuario selecciona la opción eliminar alumno.
2. El sistema muestra un formulario donde ingresar el Id del alumno a eliminar.
3. El usuario ingresa el Id del alumno que desea eliminar.
4. El sistema obtiene la información del alumno y la muestra en un formulario.
5. El usuario confirma la eliminación.
6. El alumno es eliminado del sistema.
7. Los pasos del 2 al 6 son repetidos por cada alumno que el usuario desee eliminar.Cuando el usuario haya finalizado de eliminar alumnos, el caso de uso termina.

Post condición: El alumno es eliminado del sistema.

Flujo Alterno "Alumno ya Existe":

1. Esta alternativa empieza en el paso 4 del flujo normal de eventos, cuando se agrega un nuevo alumno que el sistema encuentra que ya existe.
2. El sistema muestra un mensaje de error "Alumno ya Existe".
3. El usuario puede cambiar el nombre del alumno, confirmar la creación de un alumno con el mismo nombre o cancelar la operación terminando el caso de uso.

Flujo Alterno "Alumno no Existe":

1. Esta alternativa empieza en el paso 4 del flujo alterno "Modificar Alumno" o en el paso 4 del flujo alterno "Eliminar Alumno" cuando el sistema detecta que el alumno indicado no existe.
2. El sistema muestra un mensaje de error "Alumno no Existe".
3. El usuario puede cambiar el Id del alumno o cancelar la operación finalizando el caso de uso.

Reglas de Estilo

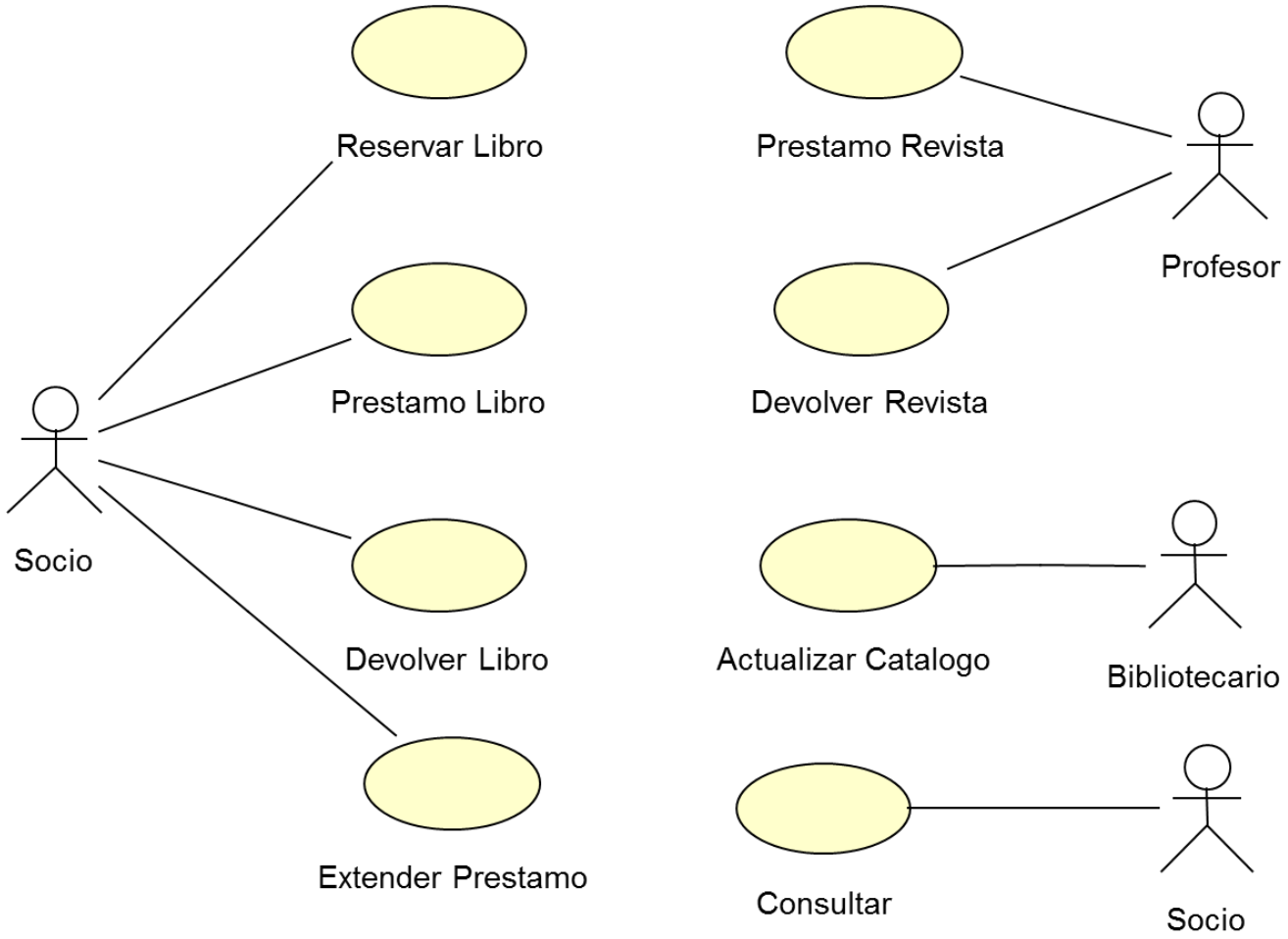
- Narrar el flujo de eventos usando ***voz activa***, *en* ***tiempo presente*** y *desde* ***la perspectiva del actor.***
- Evitar el uso de la *voz pasiva*: La clave ***es introducida*** por el usuario
- Preferir la *voz activa* El usuario ***introduce*** la clave"
- Exprese cada paso del flujo usando la forma ***llamada y respuesta***:
- El caso de uso que se describe **debe expresar *un solo requisito funcional***
- Sin embargo, un caso de uso ***puede*** **expresar *más* de un requisito NO funcional**

Reglas de Estilo

- Cada actor y caso de uso debe tener un nombre único
- Los actores deben tener nombres y/o iconos representativos. Los nombres de los actores deben *representar roles*
- El nombre de un caso de uso debe *indicar acción* y debe ser *claro y conciso*
- Evitar el cruce de líneas
- Evite tener demasiados casos de uso en el mismo diagrama
- Evite el uso complejo de relaciones de extensión, especialización e inclusión

Verbo (Infinitivo) + Predicado

Ejemplo : DCU


Reservar Libro
Prestamo Revista
Profesor
Prestamo Libro
Devolver Revista
Socio
Devolver Libro
Actualizar Catalogo
Bibliotecario
Extender Prestamo
Consultar
Socio

CU y Colaboraciones

Un caso de uso se describe el comportamiento esperado del sistema, pero ***no se especifica cómo se implementa.*** Un caso de uso se implementa a través de una ***colaboración:***

"Sociedad de clases y otros elementos que colaborarán para realizar el comportamiento expresado en un caso de uso"

- Una colaboración tiene dos partes:
  - **parte estática** (diagramas de clases) y
  - **parte dinámica** (diagramas de secuencia).

Casos de uso y Colaboraciones

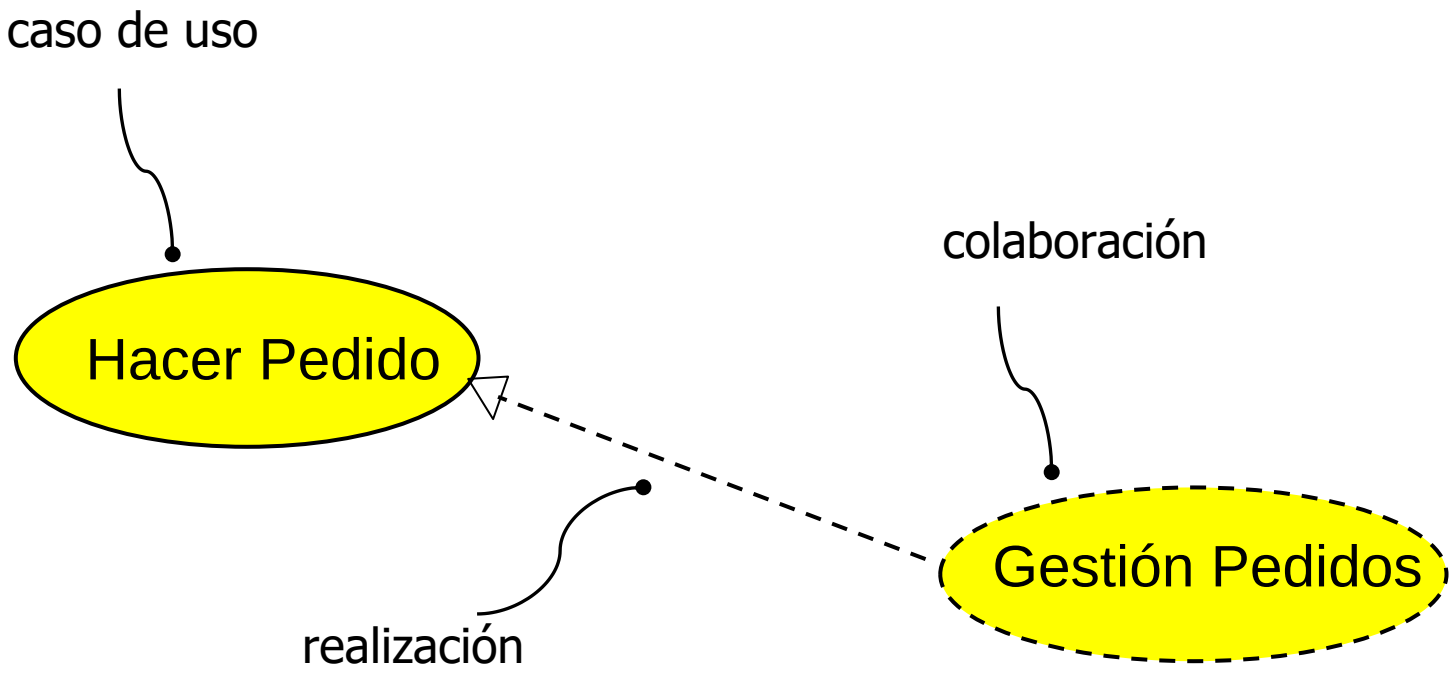
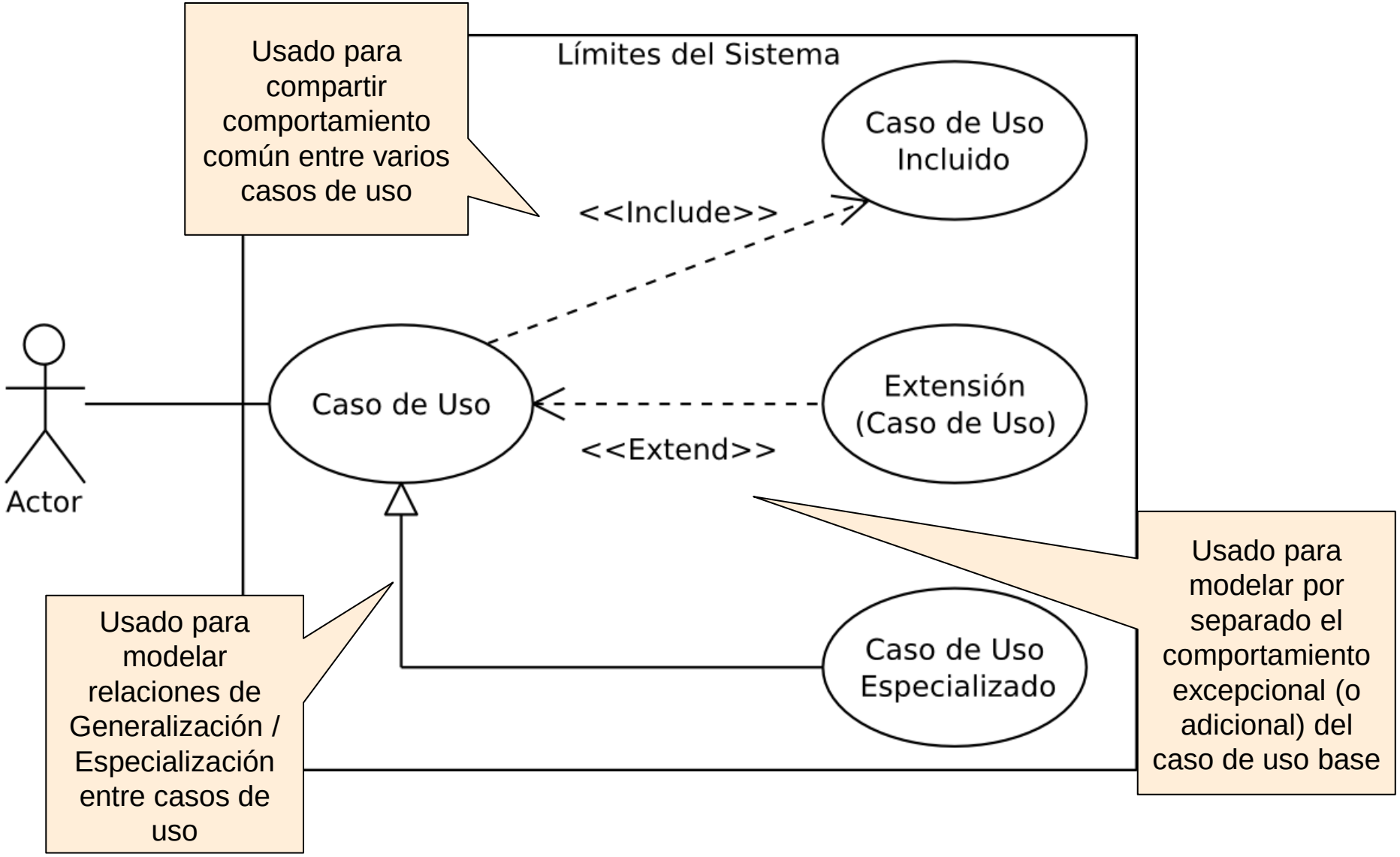


Diagrama de Casos de Usos



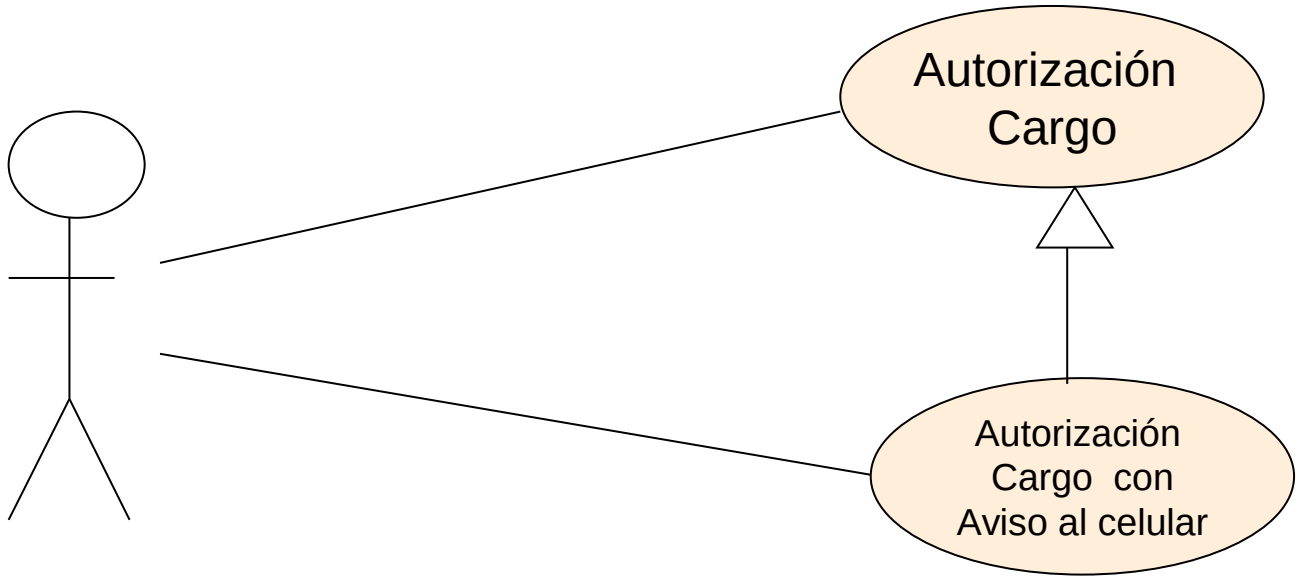
Organización de Casos de uso

Tres tipos de relaciones:

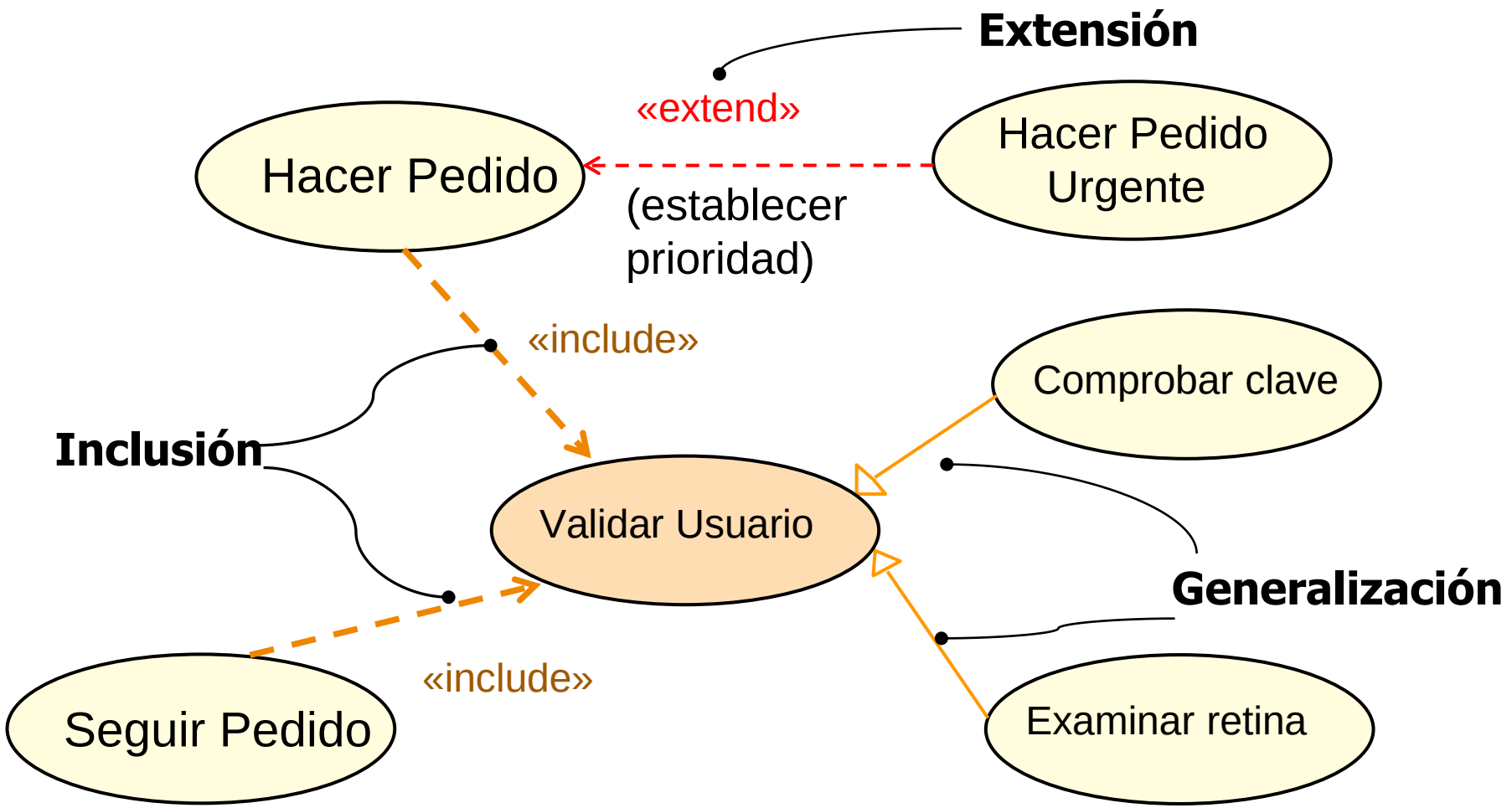
- **Generalización :** Un cdu hereda el comportamiento y significado de otro.
- **Inclusión:** Un cdu base incorpora explícitamente el comportamiento de otro en algún lugar de su secuencia.
- **Extensión :** Un cdu base incorpora implícitamente el comportamiento de otro cdu en el lugar especificado indirectamente por este otro cdu.

Generalización

- La herencia indica que un objeto tiene desde el momento de su creación, acceso a todas las propiedades de otra clase.
- Esto mismo se aplica a los actores y a los Casos de Uso, se conoce como generalización y a veces se especifica con una **relación "es un"**



Ejemplo


Extensión
«extend»
Hacer Pedido
Hacer Pedido Urgente
(establecer prioridad)
«include»
Inclusión
Comprobar clave
Validar Usuario
Generalización
«include»
Seguir Pedido
Examinar retina

Relación de inclusión

Permite **factorizar un comportamiento** en un caso de uso aparte y evitar repetir un mismo flujo en diferentes casos de uso.

Ejemplo:

Hacer Pedido:

Obtener y verificar el número de pedido;
Incluir *"Validar usuario"*;
Recoger los ítem del pedido del usuario;
…

Relación de extensión

Sirve para modelar la **parte opcional** del sistema, o un subflujo que sólo **se ejecuta bajo ciertas condiciones. El** caso de uso base incluye una serie de puntos de extensión.

Ejemplo:

Hacer Pedido:
Incluir "*Validar usuario*";
Recoger los ítem del pedido del usuario;
***Establecer prioridad*: punto de extensión**
Enviar pedido para ser procesado según la prioridad.

Relación de extensión

- Ejemplo:

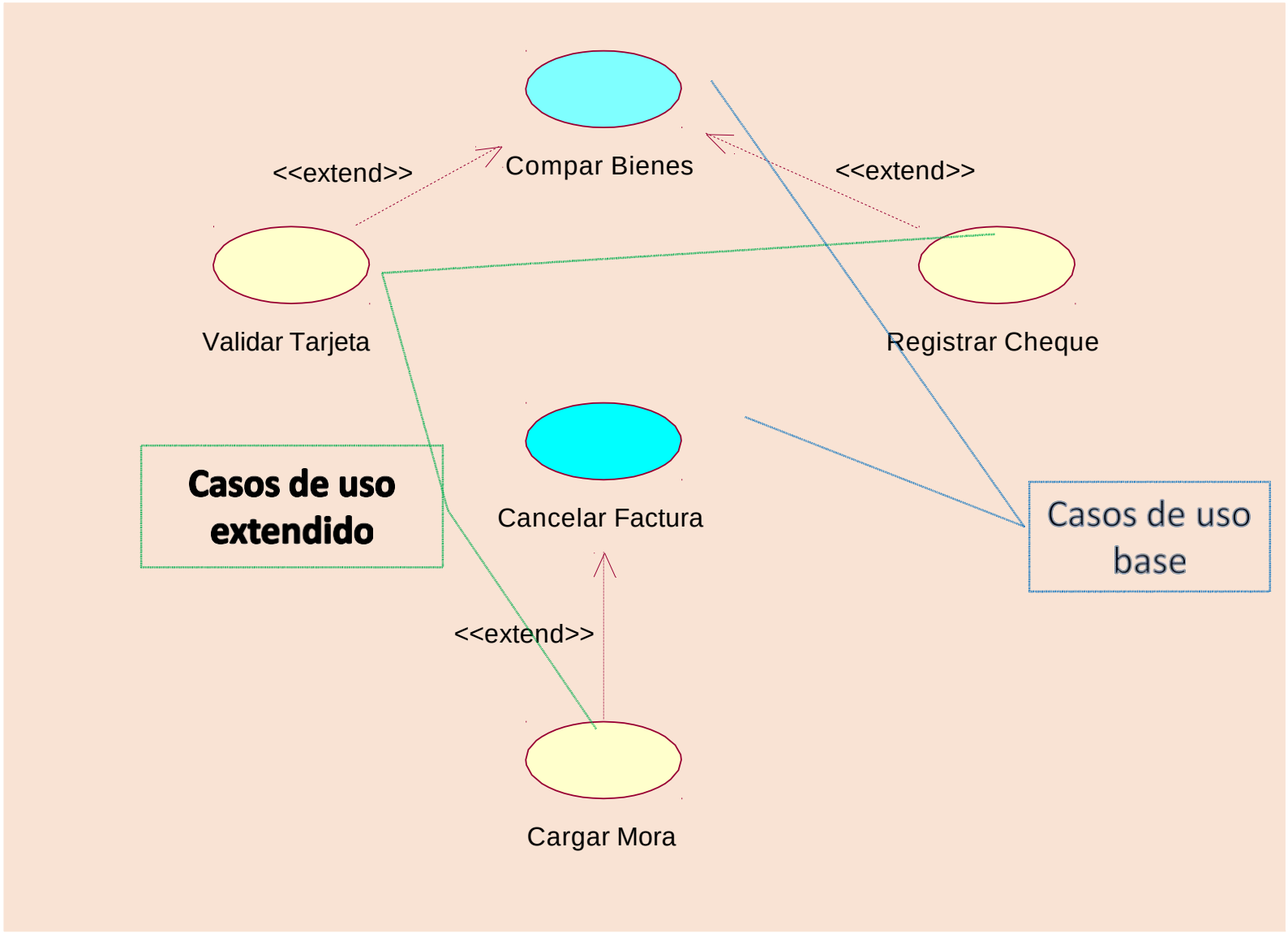
Hacer Pedido:

Incluir "*Validar usuario*";

Recoger los ítem del pedido del usuario;

***Establecer prioridad*: punto de extensión**

Enviar pedido para ser procesado según la prioridad.


Compar Bienes
<<extend>>
<<extend>>
Validar Tarjeta
Registrar Cheque
Casos de uso extendido
Cancelar Factura
Casos de uso base
<<extend>>
Cargar Mora

Ejemplo del escenario primario de un <<extend>>

Caso de Uso: **Comprar Bienes**

Descripción: Realiza la compra de un bien indicado

Punto de Extensión

Flujo de Eventos:

1. El cajero selecciona la opción comprar bienes.
2. El cajero ingresa el código del bien a comprar.
3. El sistema calcula el precio e impuesto del bien. **<Extend Point 1>**
4. El cajero selecciona aceptar y el caso de uso termina.

- **Caso de Uso**: **Validar Tarjeta**
- **Descripció**n: Verifica la Tarjeta de crédito de un cliente.
- Insertar Segmento en **<Extend Point 1>**
  1. El cajero ingresa el número de la tarjeta de crédito del cliente.
  2. El sistema verifica que el numero de la tarjeta sea valido
  3. El sistema aprueba la tarjeta
  4. Se carga el monto a la tarjeta del cliente.

Ejercicio:

Una empresa gestiona un conjunto de inmuebles, que administra en calidad de propietaria.

Cada inmueble puede ser bien un local (local comercial, oficinas, etc.), un departamento o bien un edificio que a su vez tiene departamento y locales. Como el número de inmuebles que la empresa gestiona no es un número fijo, la aplicación debe permitir tanto introducir inmuebles nuevos, así como darlos de baja, modificarlos y consultarlos.

Asimismo, que una empresa administre un edificio determinado no implica que gestione todos sus departamento y locales, por lo que la aplicación también deberá permitir introducir nuevos pisos o locales, darlos de baja, modificarlos y hacer consultas sobre ellos.

Cualquier persona que tenga una nómina, un aval bancario, un contrato de trabajo o venga avalado por otra persona puede alquilar el edificio completo o alguno de los departamentos o locales que no estén ya alquilados, y posteriormente desalquilarlo.

Por ello, deberán poder ser dados de alta, si son nuevos inquilinos, con sus datos correspondientes (nombre, DNI, edad, sexo, …), poder modificarlos, darlos de baja, consultarlos, etc. La aplicación ofrece acceso web para que un inquilino puede modificar o consultar sus datos, pero no darse de baja o de alta. Para la realización de cualquiera de estas operaciones es necesaria la identificación por parte del inquilino.

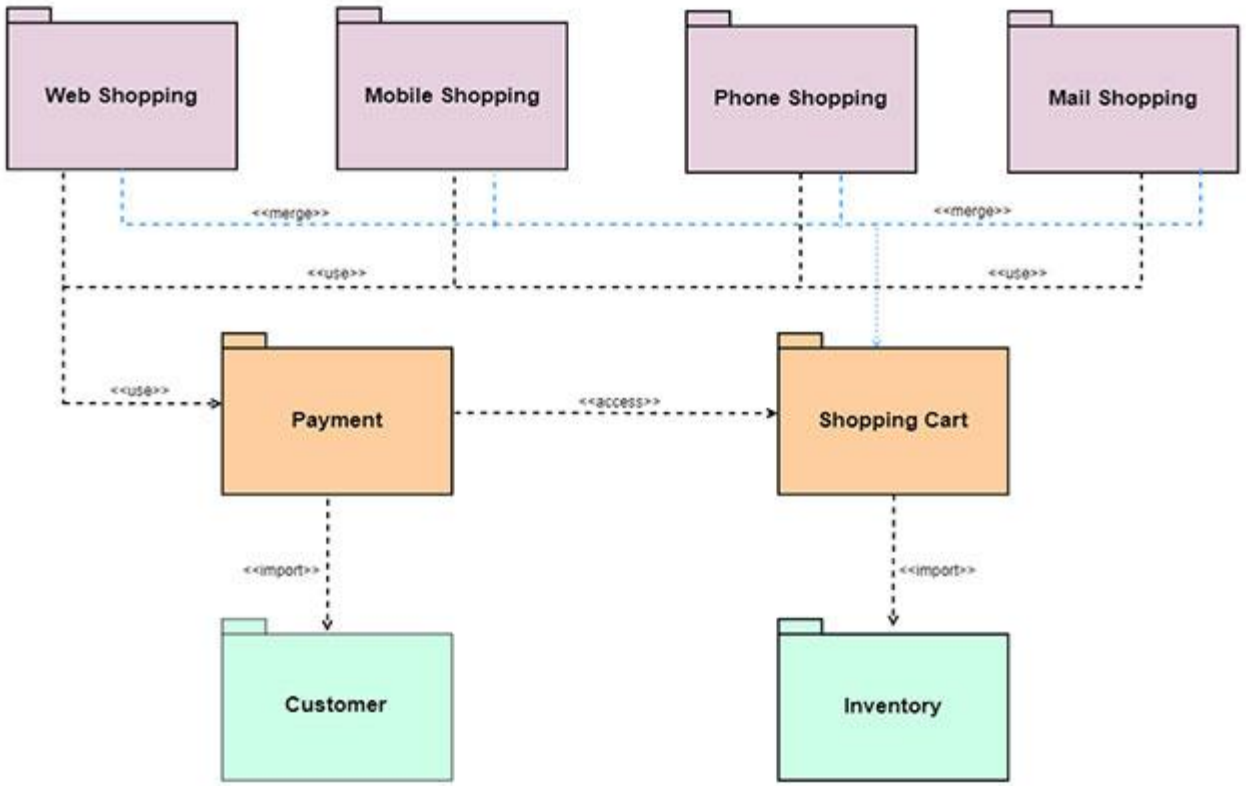


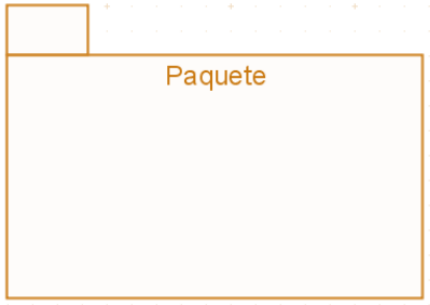
Diagrama de Paquetes

Paquetes

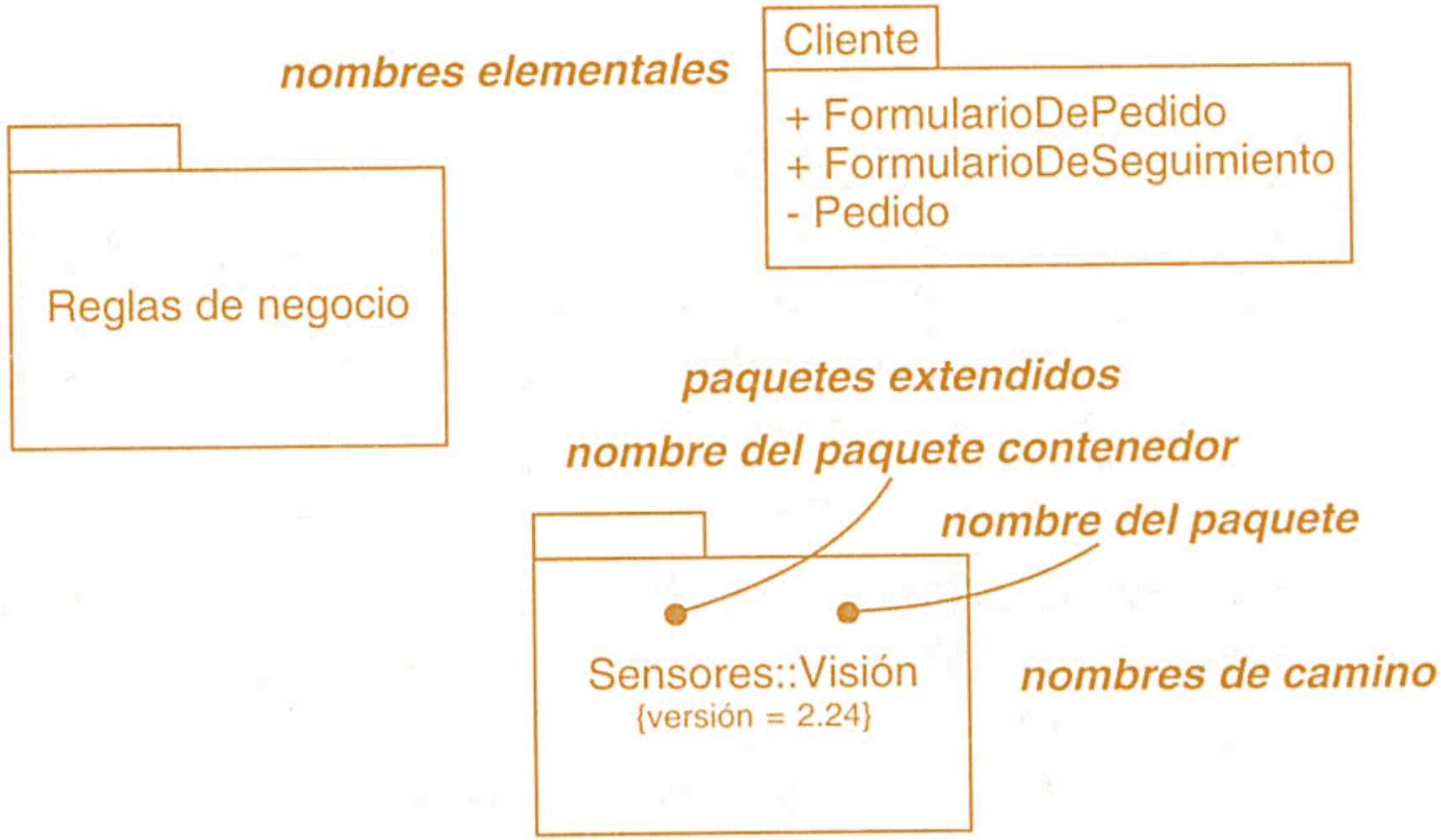
Un paquete es un mecanismo de propósito general para organizar un modelo de manera jerárquica.

Uso:

- Organizar los elementos en los modelos para comprenderlos más fácilmente.
- Controlar el acceso a sus contenidos para controlar las líneas de separación de la arquitectura del sistema

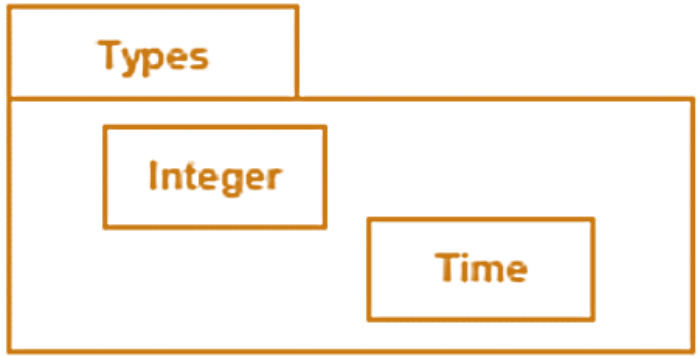


Notación Paquetes

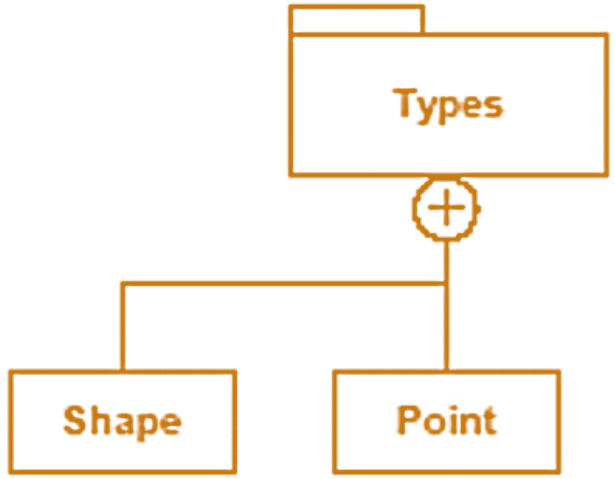


Representación de Paquetes

Notación sin contenido



Notación interna



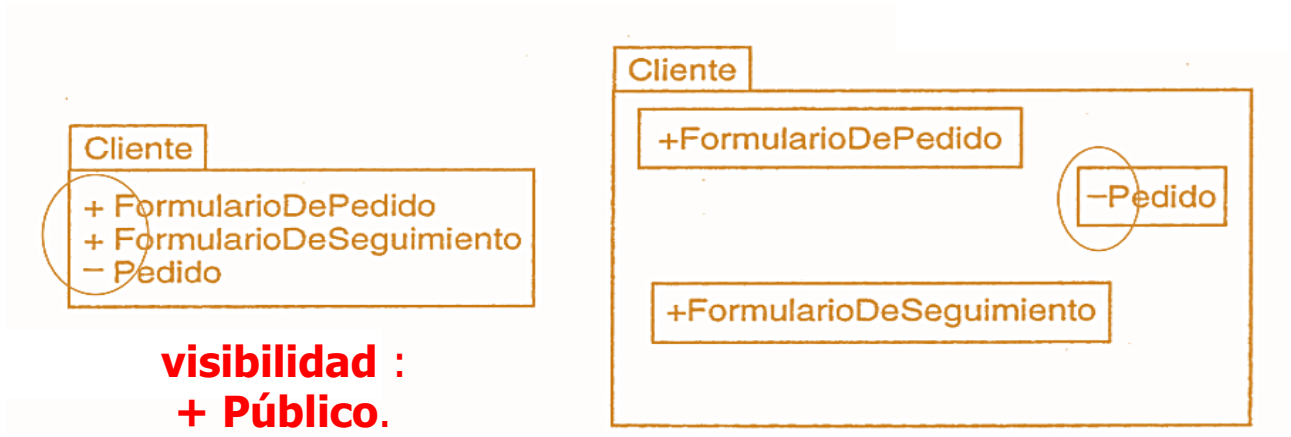
Notación externa

- Las clases son abstracciones de aspectos del problema o la solución.
- Los paquetes son mecanismos para organizar, pero no tienen identidad (no puede haber instancias de paquetes)

Paquetes - Contenido

Los paquetes pueden contener **elementos** como clases, interfaces, componentes, nodos, colaboraciones, casos de uso e incluso otros paquetes.

- Entre un paquete y sus elementos existe una **relación de composición**
- Un paquete forma un **espacio de nombres** (namespace)

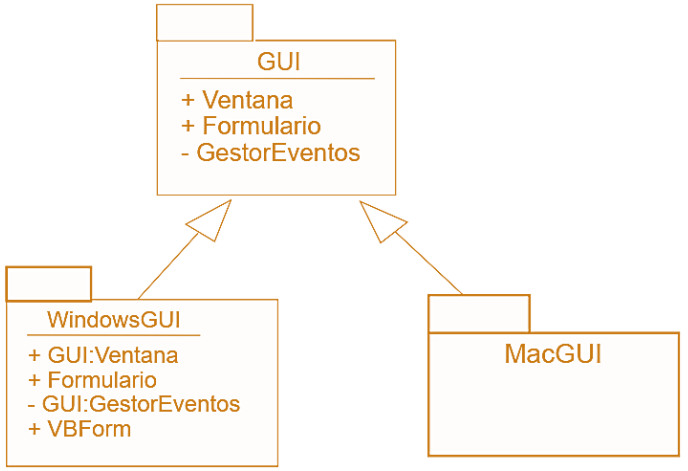


visibilidad :
+ Público.

visibilidad :
- Privado.

Paquetes - Relaciones

Generalización: Un paquete especializado puede usarse en sustitución de un paquete más general, del cual hereda.



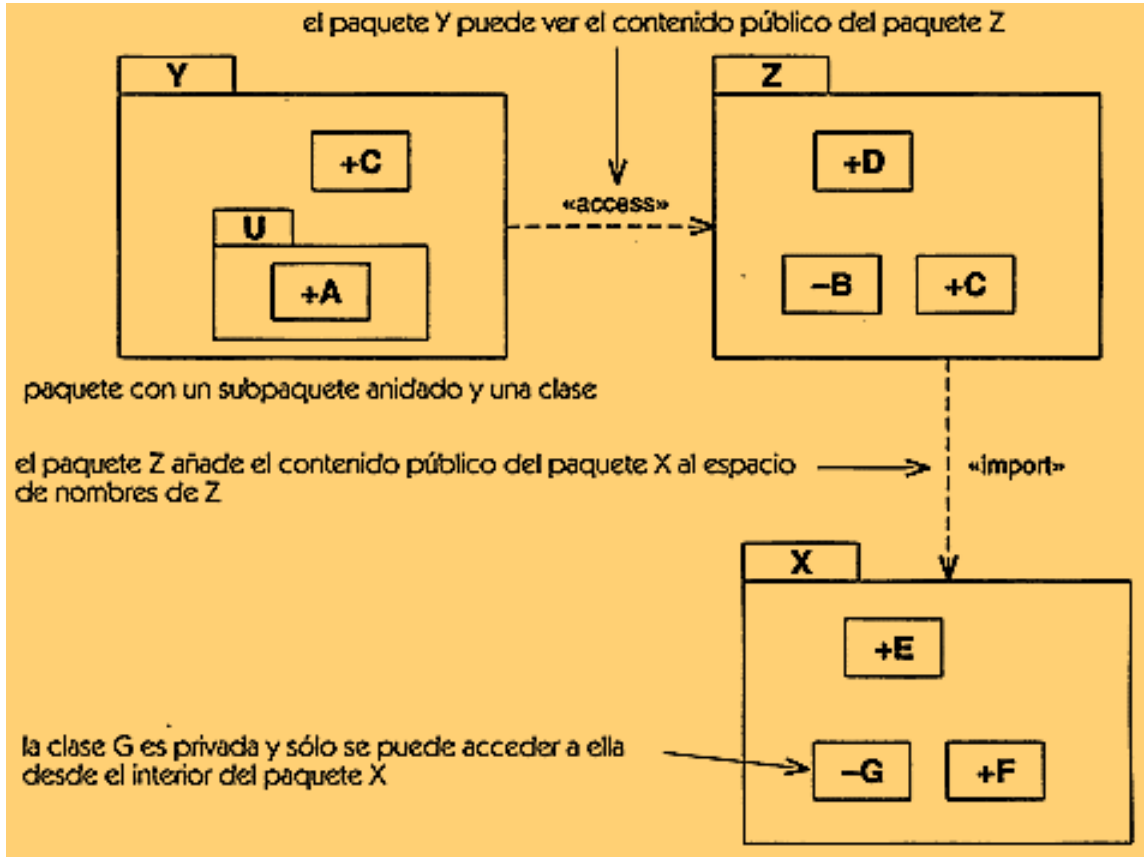
Dependencias: enotan que algún elemento de un paquete depende de los elementos en otro paquete. (3 tipos **Importación, Exportación, Acceso)**

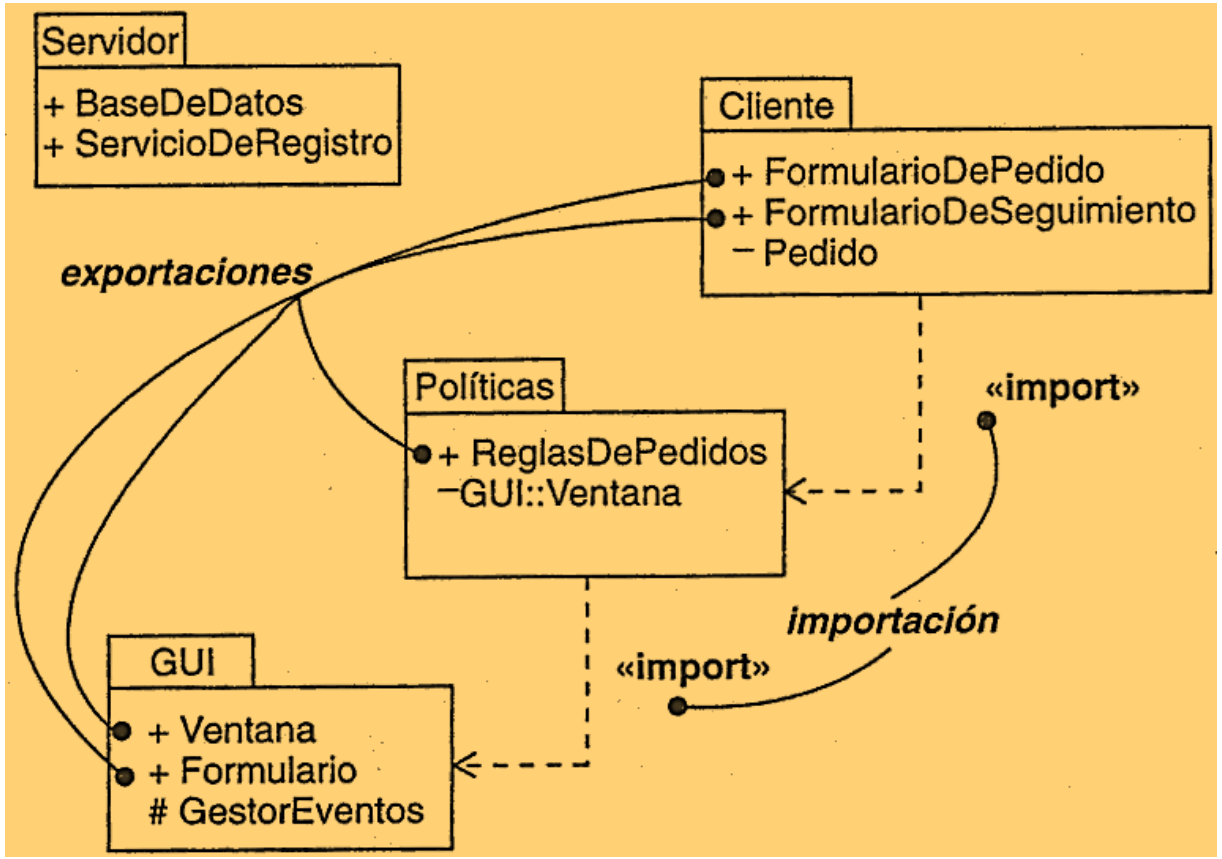
Paquetes - Relaciones

En este caso, Webshop al importar a Sopping Cart, también está importando lo que este importó de Types. Sin embargo, esto no sucede con Auxiliary, este es solo usable por Shopping Cart.

Exportación : La parte pública de un paquete son sus exportaciones

Paquetes - Relaciones





Paquetes - Fusión

Una relación de fusión (merge) entre dos paquetes especifica que el contenido del paquete origen (receptor) se extiende con el contenido del paquete destino .

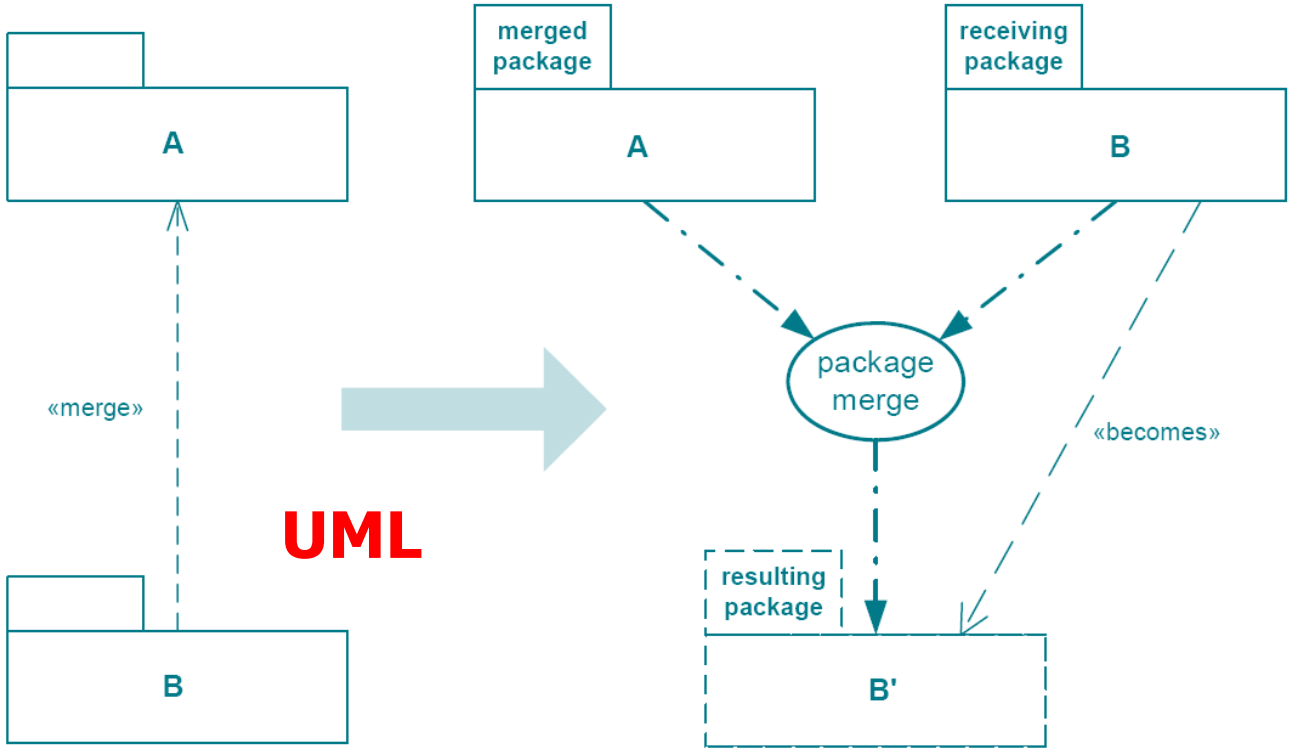
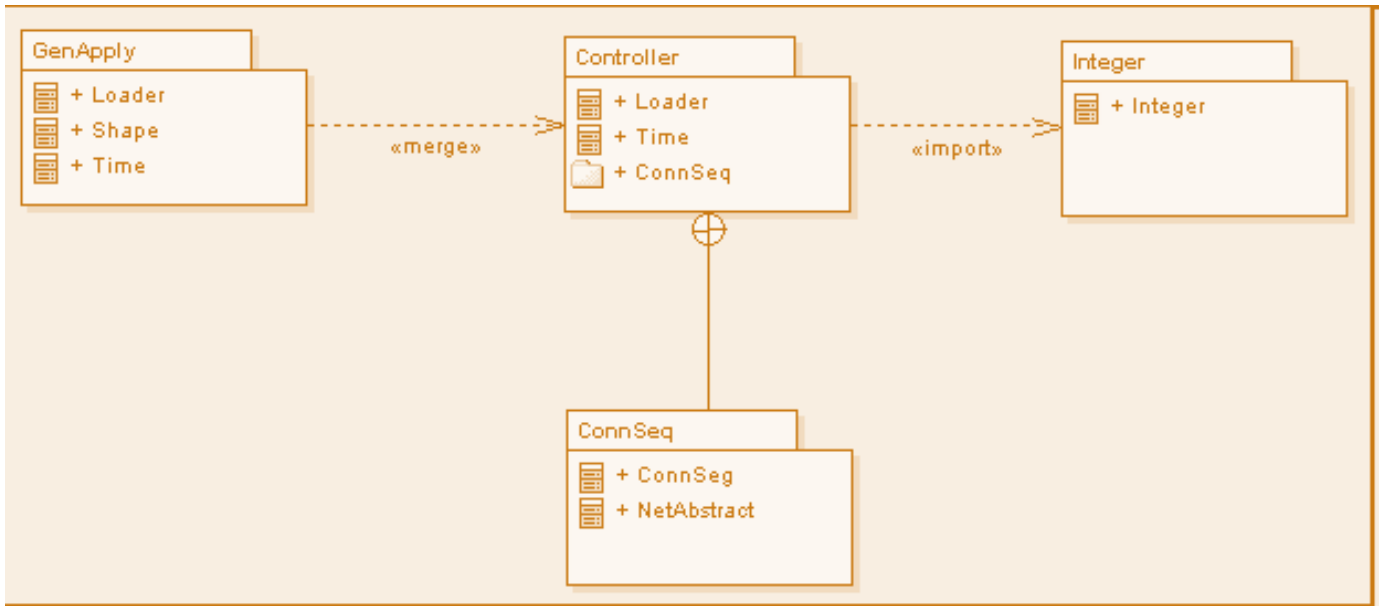


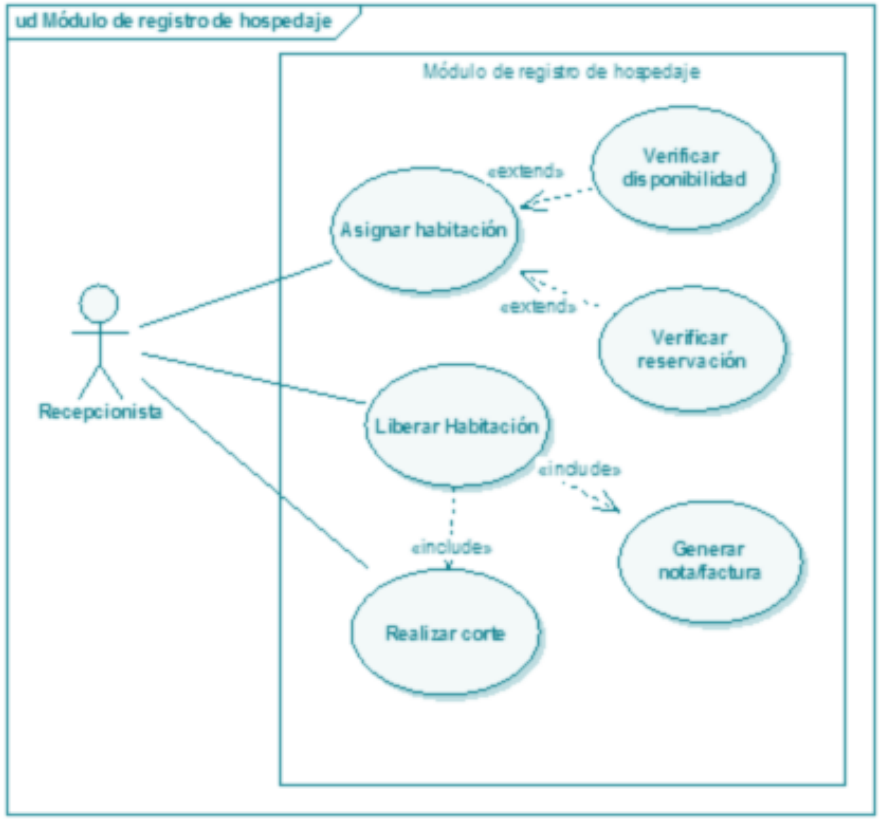
Diagrama de Paquetes

Un diagrama de paquetes es un diagrama de estructura cuyo contenido es, principalmente, paquetes y sus relaciones.



Ejemplo:

Diagrama de Casos de Uso simplificado con el uso de paquetes.



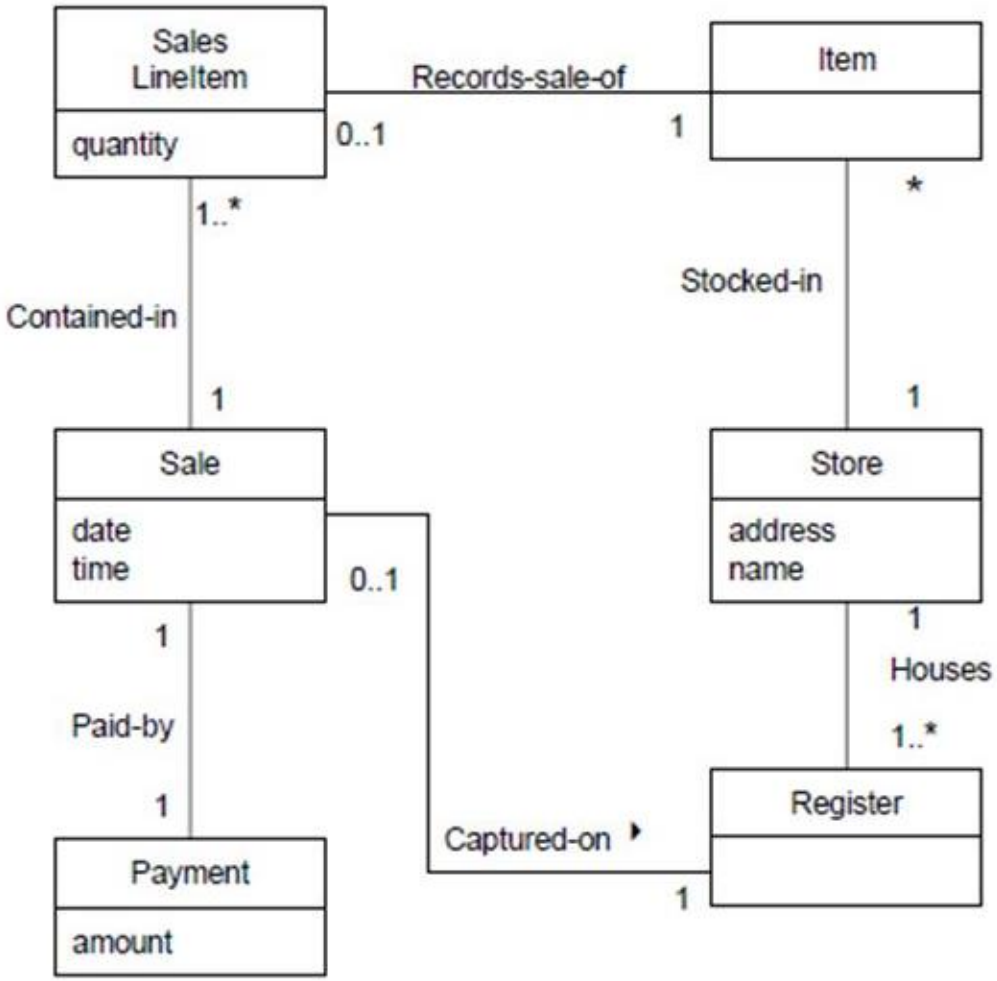


Diagrama Conceptual

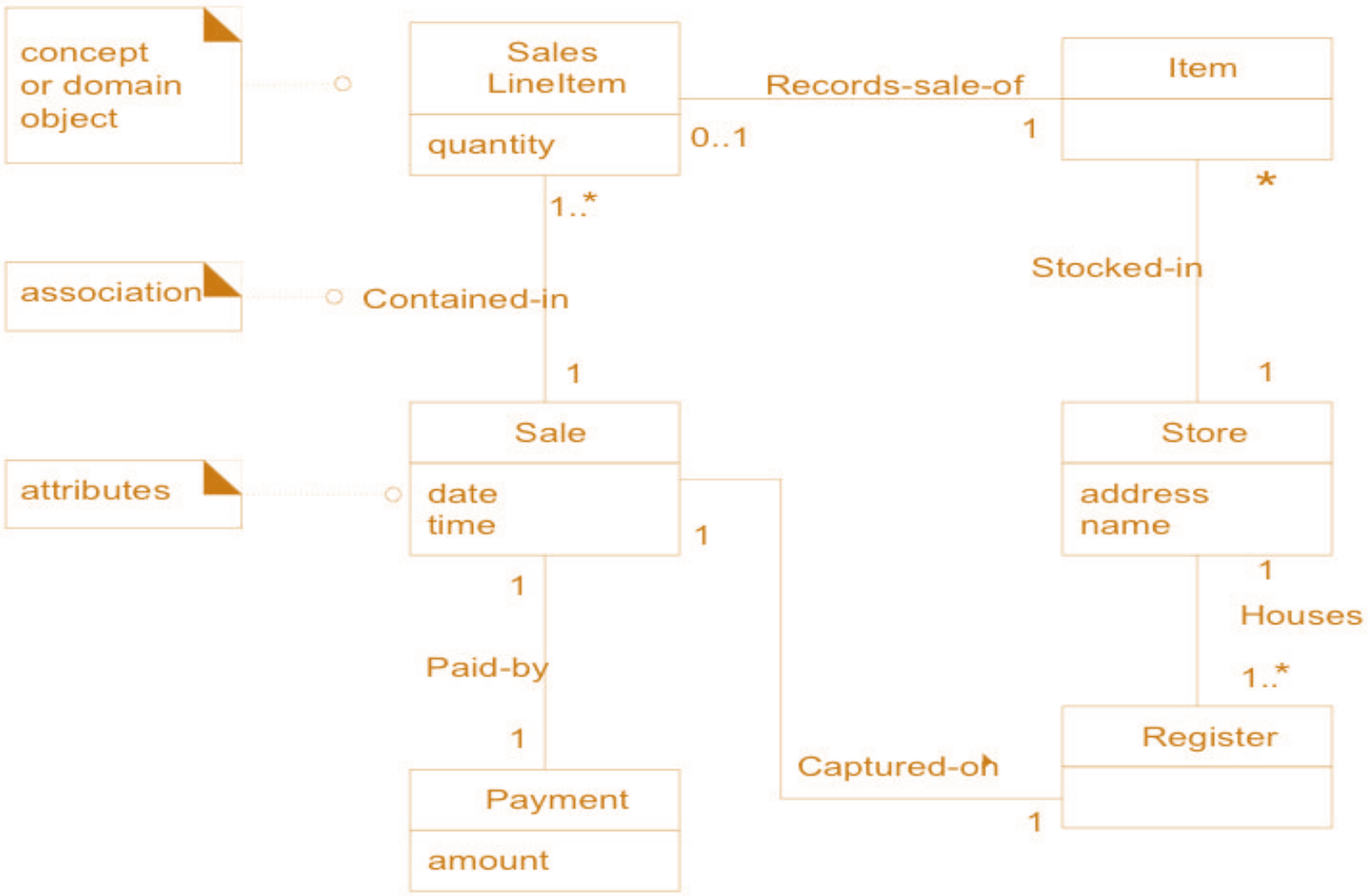
Qué es un Modelo de Dominio

Un Modelo de Dominio es una representación *visual* de clases conceptuales o de objetos reales en un dominio de interés.

Un Modelo de Dominio consiste en un conjunto de diagramas de clases, sin definición de operaciones

- **Entrada:**
  - Description del problema,
  - Casos de Uso
- **Salida:**
  - Un conjunto de diagramas de clases

Qué es un Modelo de Dominio



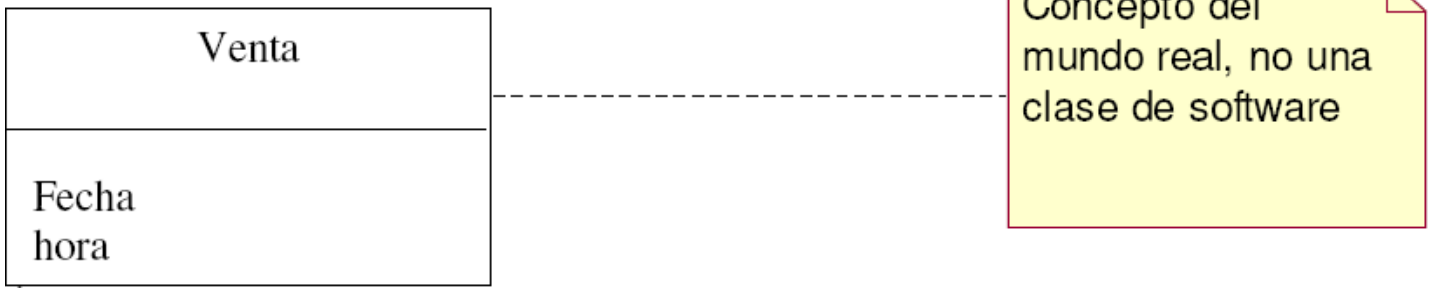
Qué es un Modelo de Dominio

Los Modelos Conceptuales permiten , descomponer el espacio del problema en unidades comprensibles (conceptos), contribuye a *esclarecer la terminología o nomenclatura del dominio.* Podemos verlo como un modelo que comunica (a los interesados como pueden serlo los desarrolladores) cuáles son los términos importantes y cómo se relacionan entre sí.

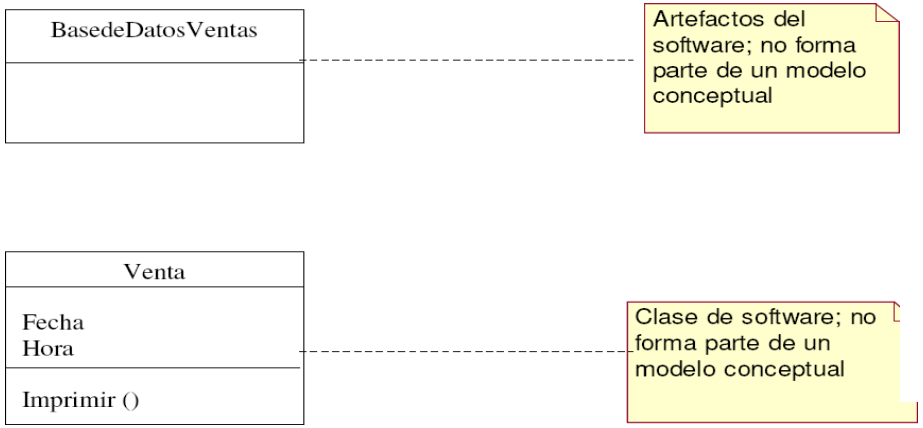
No corresponden al Modelo conceptual, los artefactos del software, como una ventana o una base de datos, salvo que el dominio a modelar se refiera a conceptos de software; por ejemplo, un modelo de interfaces gráficas para el usuario o las responsabilidades o métodos.

Construyendo el Modelo de Dominio

Conceptos: :*En términos informales el concepto es una idea, cosa u objeto.* En un lenguaje más formal, podemos considerarlo a partir de su símbolo, intensión y extensión



- ✓ **Símbolo:** palabras o imágenes que representan un concepto.
- ✓ **Intensión:** la definición del concepto.
- ✓ **Extensión:** el conjunto de ejemplos a que se aplica el concepto.



¿Como identificar conceptos?

✓Obtención de conceptos a partir de una lista de categorías de conceptos

Categoría de concepto	Ejemplos
Objetos físicos o tangibles	Puesto de venta Avión
Especificaciones, diseño o descripciones de cosas	EspecificaciondeProducto Descripcionde Vuelo
Lugares	Tienda Aeropuerto
Transacciones	Venta, Pago Reservación
Línea o renglón de elemento de transacciones	VentasLineadeProducto
Papel de personas	Cajero Piloto
Contenedores de cosas	Tienda, Cesto Avión
Cosas dentro de un contenedor	Producto Pasajero
Otro sistemas de cómputos Electromecánicos externos al sistema	SistemadeAutorizaciondeTarjetadeCredito ControldeTraficoAereo

¿Como identificar conceptos?

Categoría de concepto	Ejemplos
Otro sistemas de cómputos Electromecánicos externos al sistema	SistemadeAutorizaciondeTarjetadeCredito ControldeTraficoAereo
Conceptos de nombres abstractos	Hambre Acrofobia
Organizaciones	Departamentode VentasObj etoLineaAerea
Procesos (a menudo no están repre sentados como conceptos, pero pueden estarlo)	VentaUnProduct ReservaAsiento
Reglas y Políticas	PoliticadeReembolso PoliticadeCancelaciones
Catálogos	CatalogodeProducto Catalogodepartes
Registro de finanzas, de trabajo, de contratos de asuntos legales	Recibo, Mayor, ContratodeEmpleo BitácoradeMantenimiento
Instrumentos y servicios financieros	LineadeCredito Existencia
Manuales, libros	ManualdePersonal ManualdeReparaciones

Modelo de Dominio

El modelo de dominio proporciona una perspectiva conceptual, **objetos** del dominio o **clases** conceptuales Asociaciones entre clases conceptuales y atributos de las clases conceptuales.

Un objeto: Es una cosa con identidad única en un dominio de problema. Todos los objetos tienen una identidad y son distiguibles.
Ejemplo: Carlos Pérez, USB, son objetos

Una clase: Describe un grupo de objetos con las mismas propiedades, comportamientos y relaciones posibles. Los objetos de un dominio son el foco del modelado. Un objeto es una instancia de una clase. *Ejemplo : Alumno, Persona, Universidad y País.*

Encontrar clases conceptuales

El modelo de dominio es una visualización de elementos de un dominio de interes en el mundo real.

Tres estrategias:

1. Reusar o modificar modelos existentes, existen modelos de dominio y de datos publicados y bien elaborados para dominios comunes: inventario, finanzas, salud, etc.
2. Usar una lista de categorías.
3. Identificar sustantivos/frases nominales

Listas de Categorías

- **Objetos físicos y tangibles :** TDPV
- **Especificaciones o descripciones de cosas:** EspecificaciónDeProducto
- **Lugares:** Tienda
- **Transacciones:** Venta, Pago
- **Línea o reglón de elemento de transacciones:** VentasLineaDeProducto
- **Roles de personas:** Cajero
- **Contenedores de otras cosas :** Tienda
- **Cosas dentro de un contenedor:** Producto
- **Otros Sistemas:** Sistema de Autorización de tarjetas de crédito

Identificar Sustantivos

Analizar la descripción textual de dominio e identificar sustantivos (nombres) y frases nominales. Estos indican candidatos a clases, objetos y atributos.

- El *cliente* **llega a un** *puesto de venta* **con** *mercaderías* y/o *servicios* **que comprar.**
- El *cajero* comienza una nueva *venta.*
- **El** *cajero* **introduce el** *identificador del artículo.*
- El *sistema* registra **la** *línea de venta y* **presenta la** *descripción del artículo, precio* y *suma* parcial.

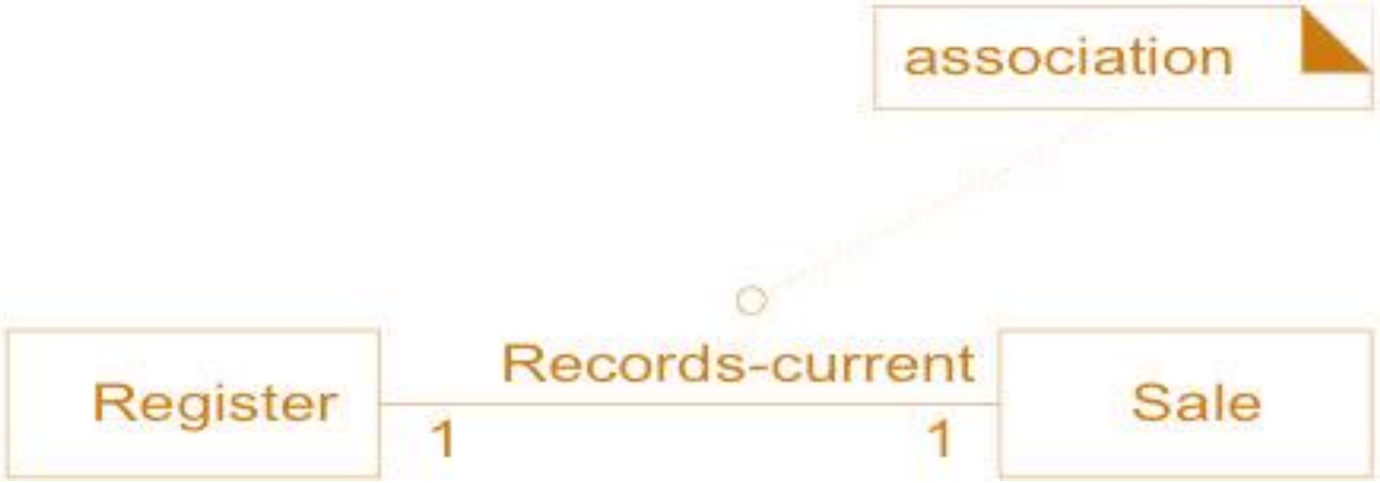
El *cajero* repite los pasos 3 y 4 hasta que se indique.

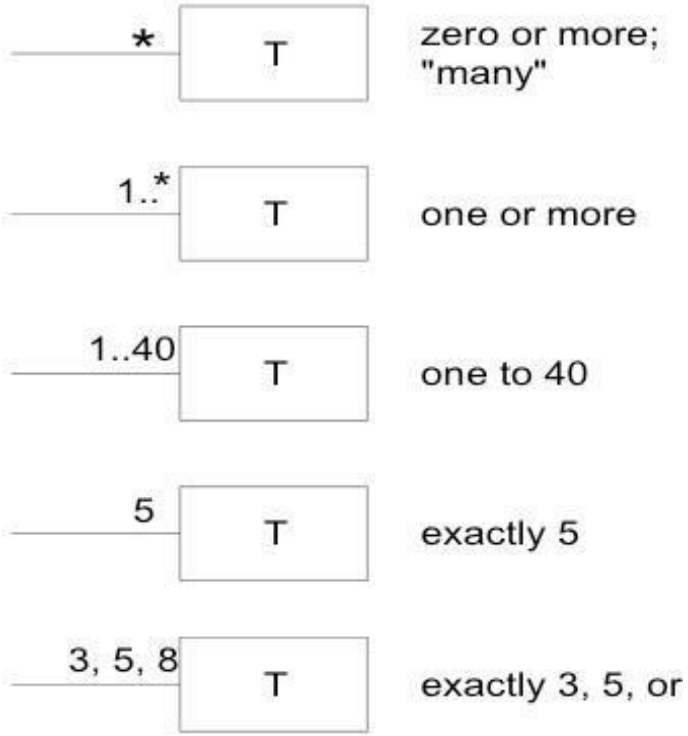
- El *sistema* presenta el *total* con los *impuestos* calculados.
- El *cajero* le dice al *cliente* el *total* y solicita el *pago.*

Clases conceptuales candidatas para el dominio de ventas : Cliente, puesto de venta, mercadería, servicio, cajero, venta, identificador de artículo, sistema, línea de venta, descripción de artículo, precio, etc..

Relacionar con Asociaciones

Una Asociación: es una relación entre objetos que indica alguna conexión con significado e interesante. Los objetos usualmente están relacionados por conexiones físicas o conceptuales.





Multiplicidad

Encontrar Asociaciones : Usar la lista de asociacione

A es una parte física de B : Caja-TDPV

A es una parte lógica de B : VentasLineaDeProducto-Venta

A está contenido físicamente en B: TDPV-Tienda

A está contenido lógicamente en B : DescripciónDeProducto – Producto

A es una descripción de B : DescripciónDeProducto – Producto

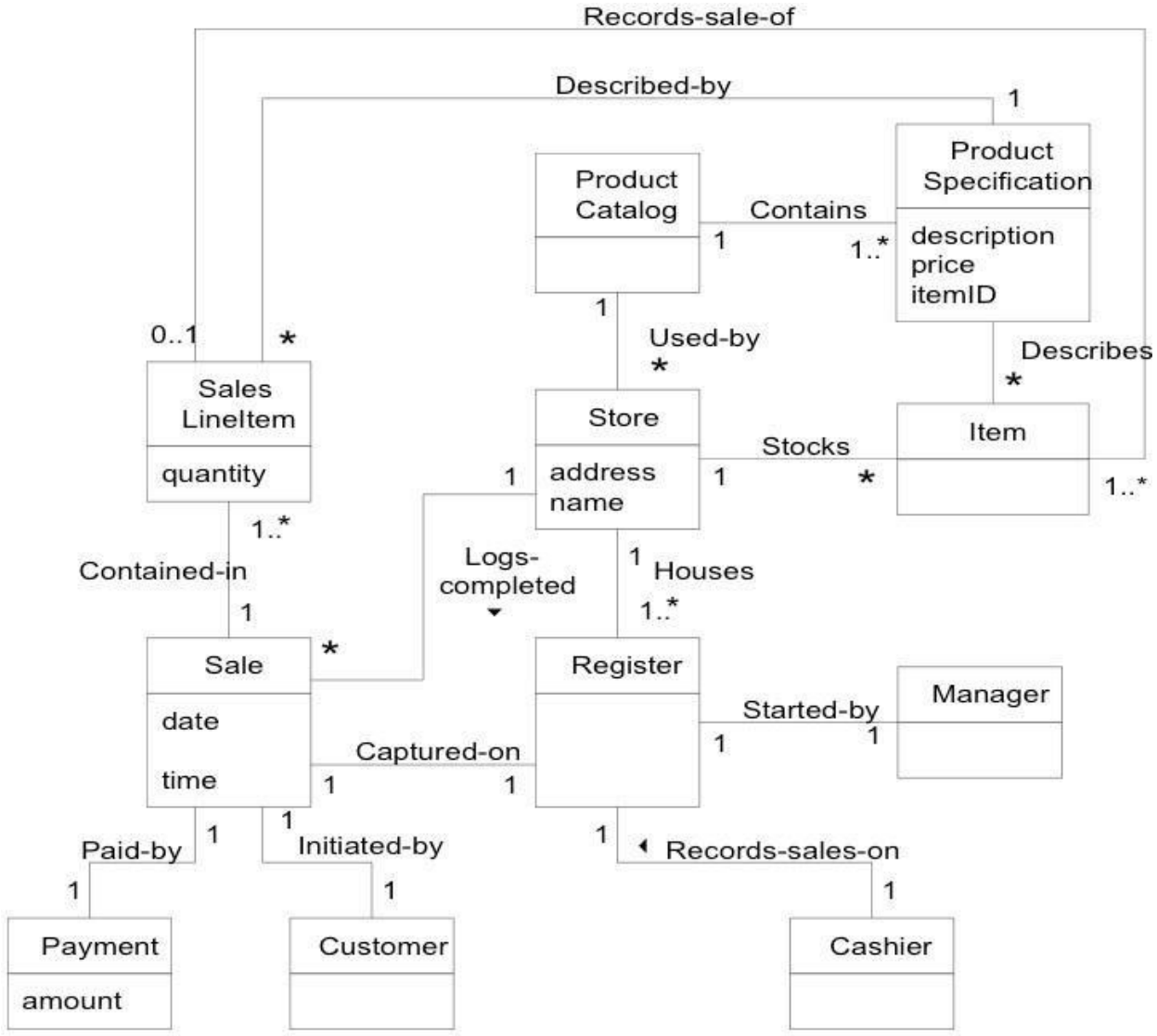
A es un elemento de línea en una transacción o reporte B: VentasLineaDeProducto-Venta

A se conoce/introduce/registra/presenta/captura B: Venta-TDPV

A es miembro de B : Cajero-Tienda

A es una sub-unidad organizacional de B : Departamento-Tienda

Ejemplo


Records-sale-of
Described-by
1
Product Specification
description
price
itemID
Product Catalog
Contains
1
1..*
1
Used-by
*
Describes
*
0..1
*
Sales LineItem
quantity
Store
address
name
Stocks
1
*
Item
1..*
1
1..*
Contained-in
1
Logs-completed
1
Houses
1..*
Sale
date
time
*
Register
Manager
Started-by
1
1
Captured-on
1
1
1
1
Paid-by
1
Initiated-by
1
1
Records-sales-on
1
1
Payment
amount
Customer
Cashier

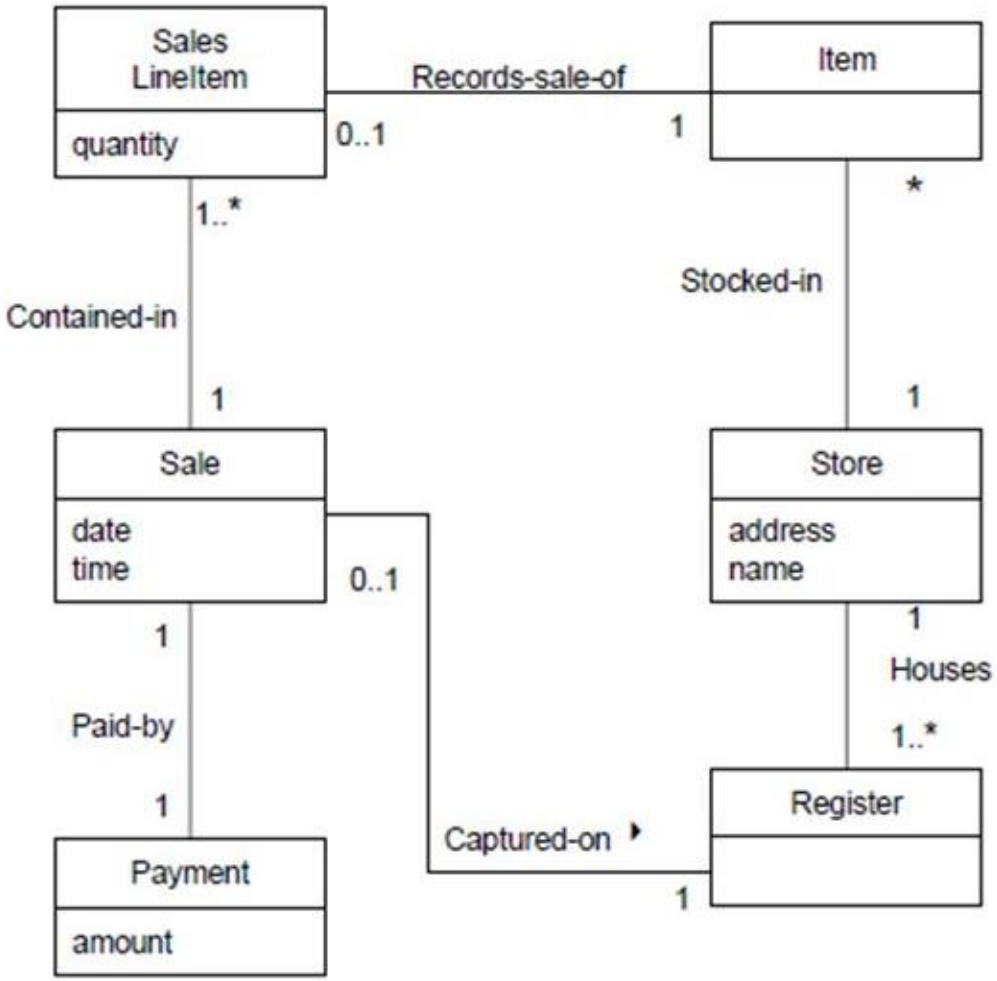


Diagrama de Clases

Introducción

- Usamos el diagrama de clases para modelar la estructura estática de un sistema, por lo tanto describe los elementos del sistema y las relaciones entre ellos.

- Estos elementos y las relaciones entre ellos no cambian en el tiempo. Es uno de los diagramas más utilizados en UML.

- El nivel de detalle del diagrama **varía en cada fase** del proceso de desarrollo de software: en fases iniciales muestra una vista conceptual y el vocabulario común a utlizar; se puede refinar dicho vocabulario hasta llegar al punto de implementación en un lenguaje de programación.

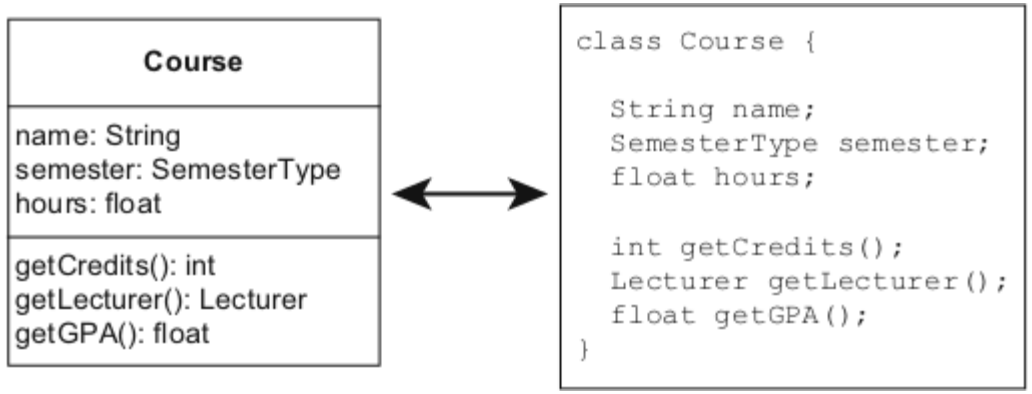
Clases

- Una “clase” es la plantilla de construcción de un conjunto de “objetos” similares que aparecen en el sistema a ser especificado.

- Pueden representar personas, animales, cosas, eventos e incluso conceptos abstractos. La clase define características estructurales (atributos) y comportamiento (operaciones).

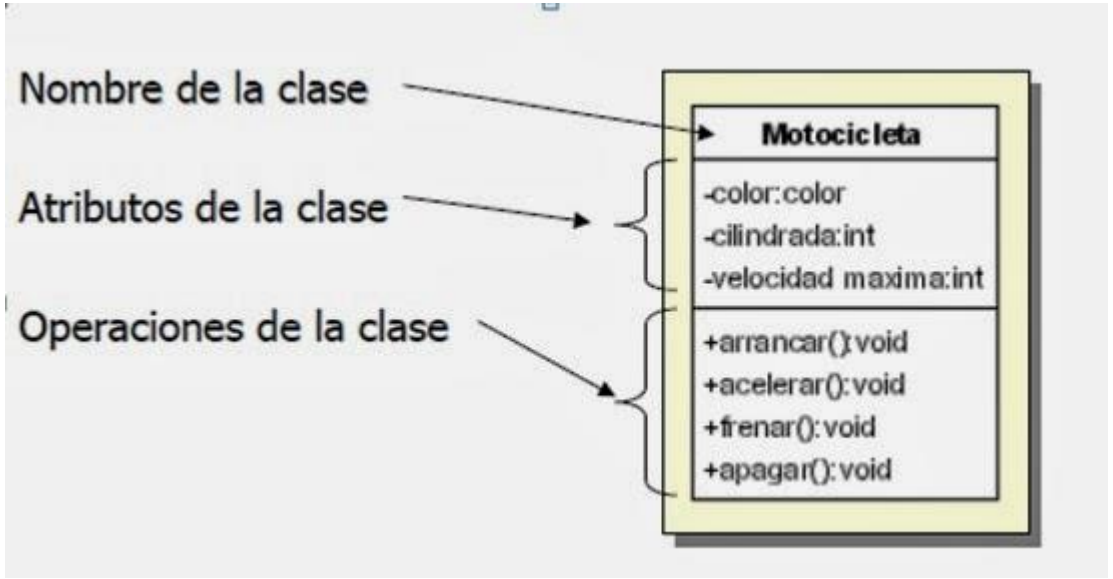
- Los “objetos” representan la forma concreta de las clases y son conocidos como “instancias”

- Debe efectuarse una **abstracción** para que el modelo no sea complejo: incluir sólo información necesaria.



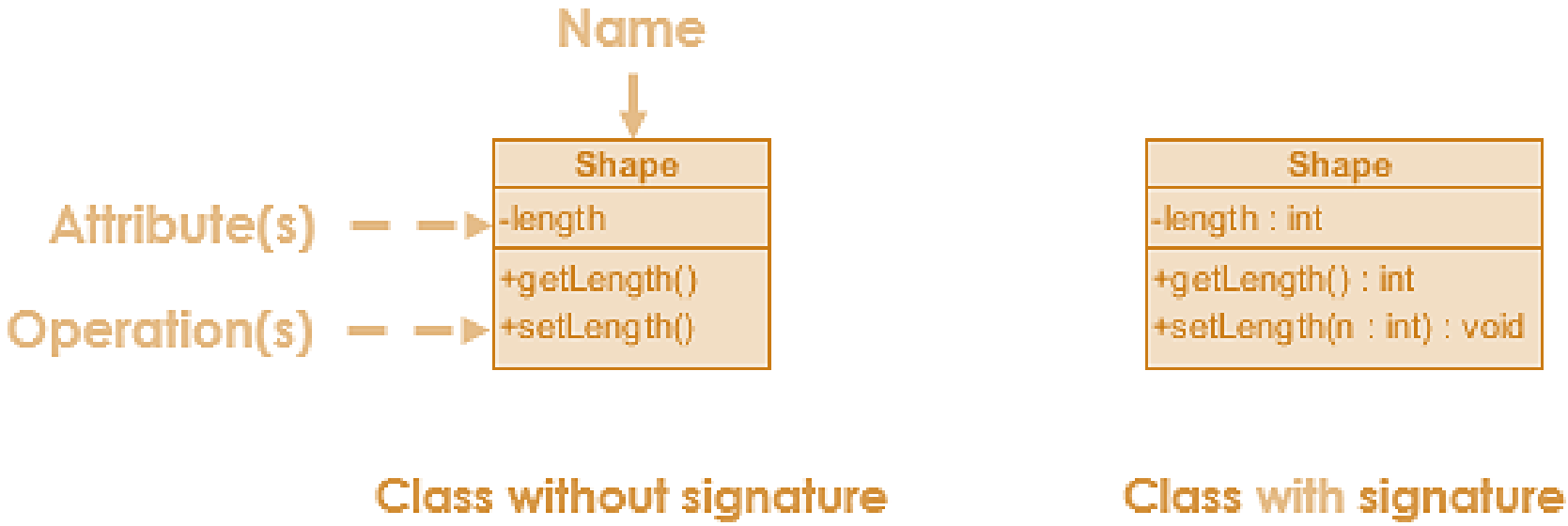
Notación

- La clase se representa por un rectángulo dividido en partes.
- La primera parte tiene el nombre de la clase (Sustantivo en Singular). El nombre normalmente se refiere a vocabulario del dominio.
- La segundo parte contiene los atributos de la clase.
- La tercera parte contiene las operaciones de la clase
- Los detalles **van apareciendo** a medida que progresa el proceso de desarrollo



Representación de clases en UML

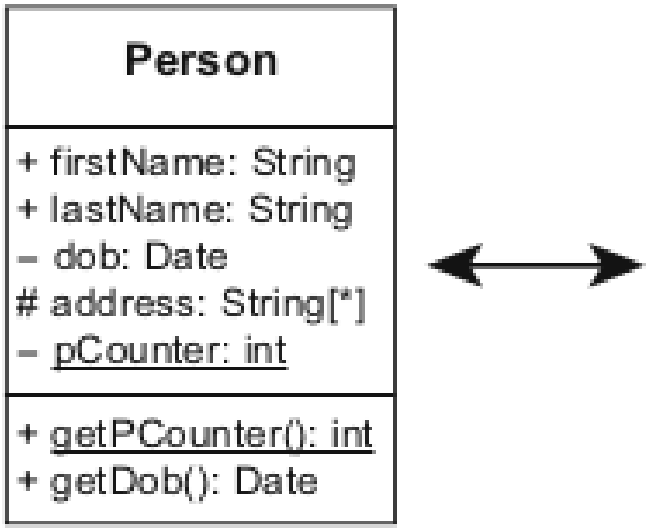
Una clase es un tipo de algo. Se puede pensar en una clase como una plantilla o plano desde el cual se pueden construir objetos de ese mismo tipo.



Elementos

- Atributos
- Operaciones
- Visibilidad
- Variables y operaciones de clase

+ publica
- privada
 # protegida
~ package



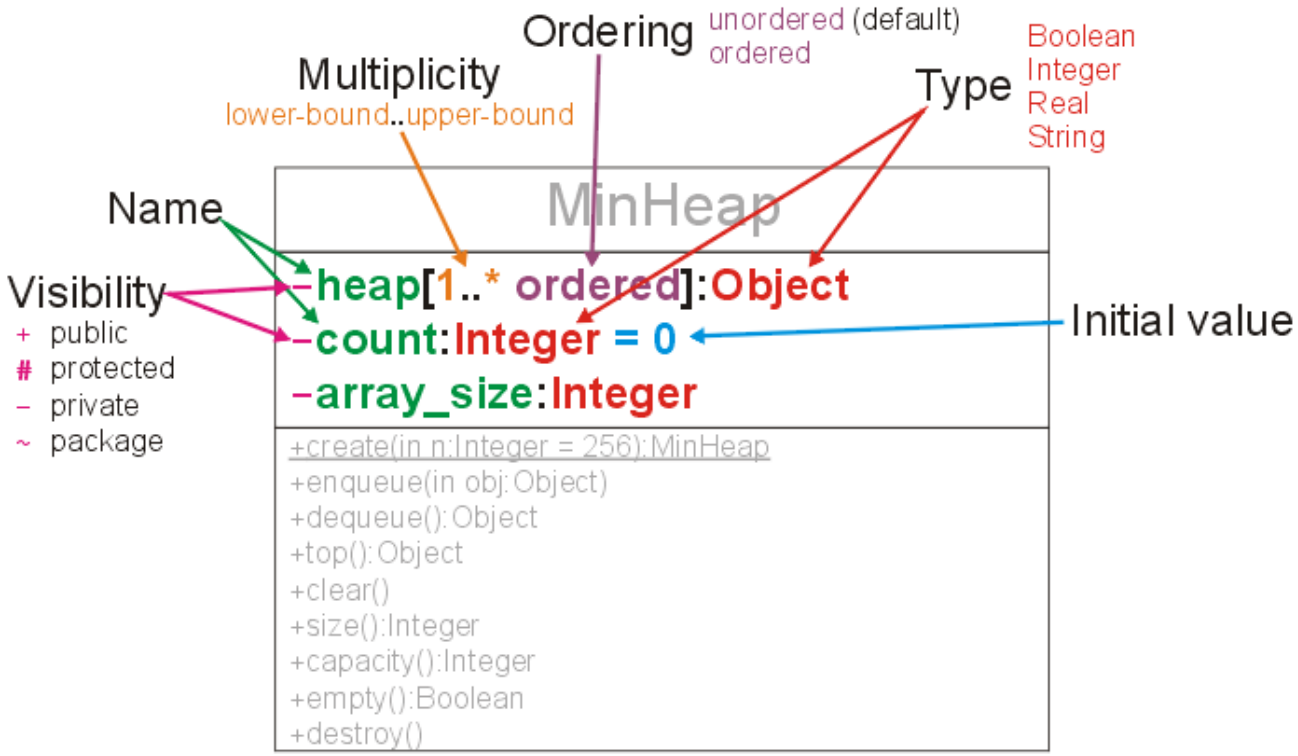

```
class Person {

  public String firstName;
  public String lastName;
  private Date dob;
  protected String[] address;
  private static int pCounter;

  public static int getPCounter() {…}
  public Date getDob() {…}
}
```

Visibilidad

¿Cómo una clase revela selectivamente sus operaciones y datos a otras clases? Mediante el uso de la visibilidad. Una vez que se aplican las características de visibilidad, se puede controlar el acceso a atributos, operaciones e incluso clases enteras para aplicar eficazmente la encapsulación.

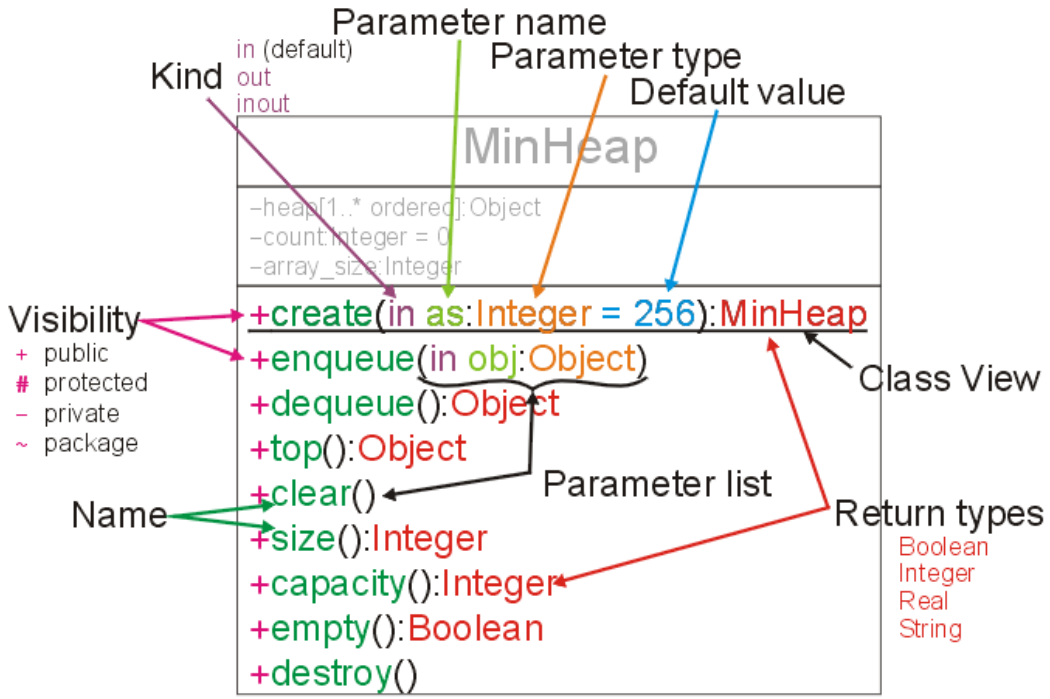
Atributos

Los atributos de una clase es la información que representan el estado de un objeto. Estos atributos se pueden representar en un diagrama de clases, ya sea colocándolos dentro de su sección del cuadro de clase, conocido como atributos en línea, o por asociación con otra clase



Parametros

Los parámetros se utilizan para especificar la información proporcionada a una operación para permitirle completar su trabajo



Multiplicidad

A veces, un atributo representará más de un objeto. De hecho, un atributo podría representar cualquier cantidad de objetos de su tipo; en software, esto es como declarar que un atributo es una lista o arreglo.

1	no mas de uno
0..1	cero o uno
*	muchos
0..*	cero o muchos
1..*	uno o muchos

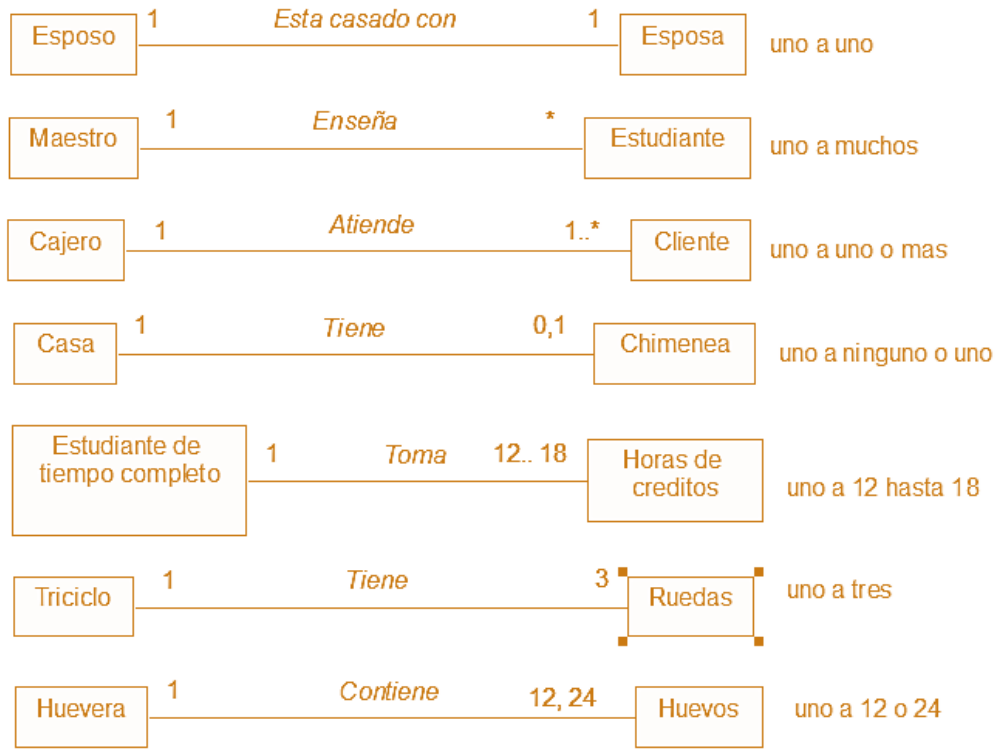
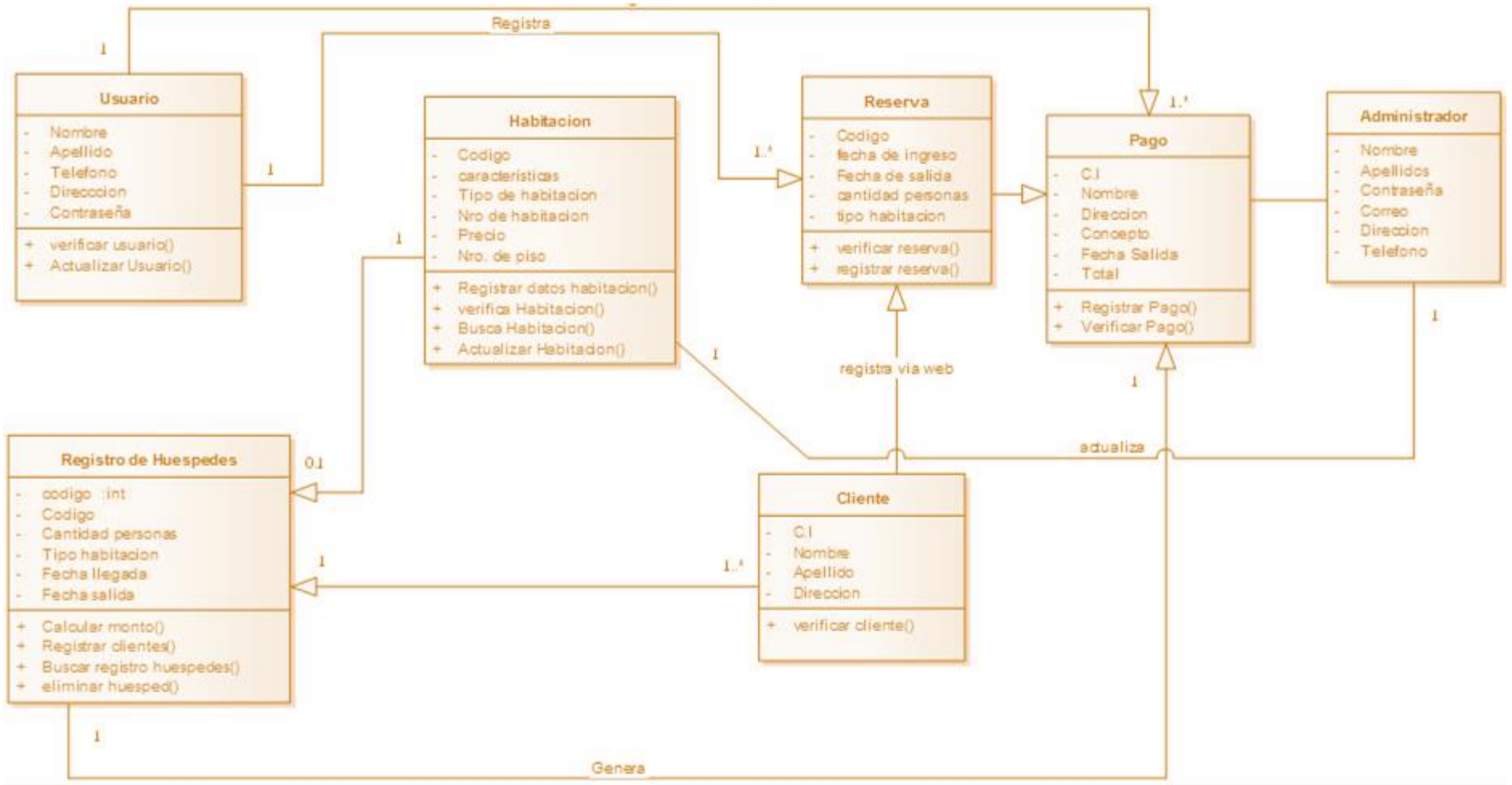


Diagrama de Clases

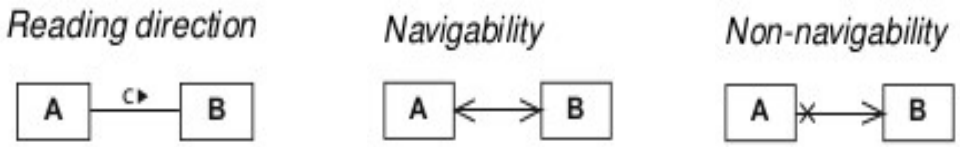
Un diagrama de clases o estructura estática muestra el conjunto de clases que forman parte de un sistema y visibilidad que tiene cada una de las clases, junto con las relaciones existentes entre clases.

Ejemplo de Diagrama de Clases

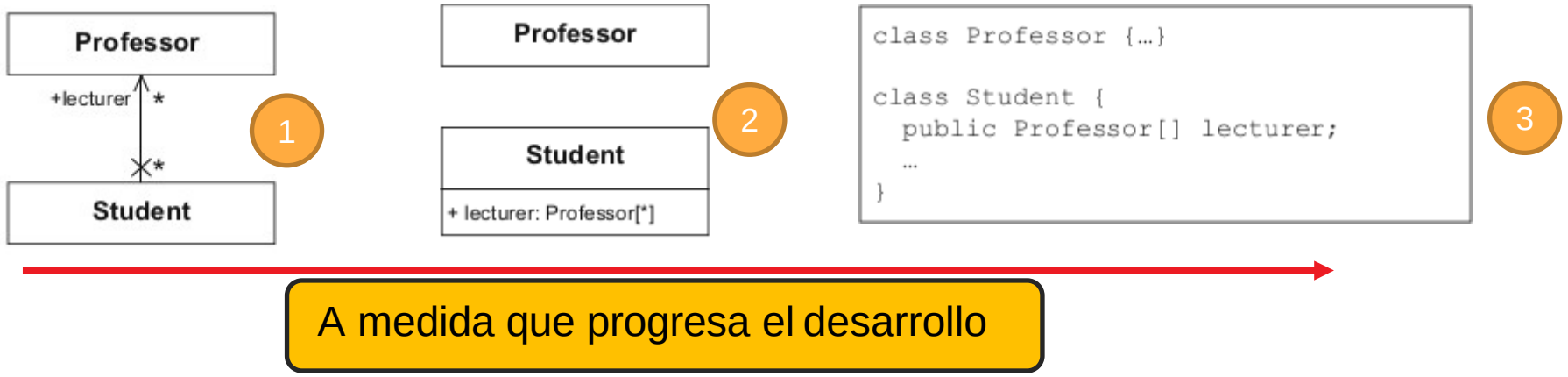


Asociaciones Binarias

- Las asociaciones entre clases posibilitan las relaciones conocidas como enlaces entre instancias de las clases.
- Asociación BINARIA: permite asociar las instancias de dos clases entre sí.

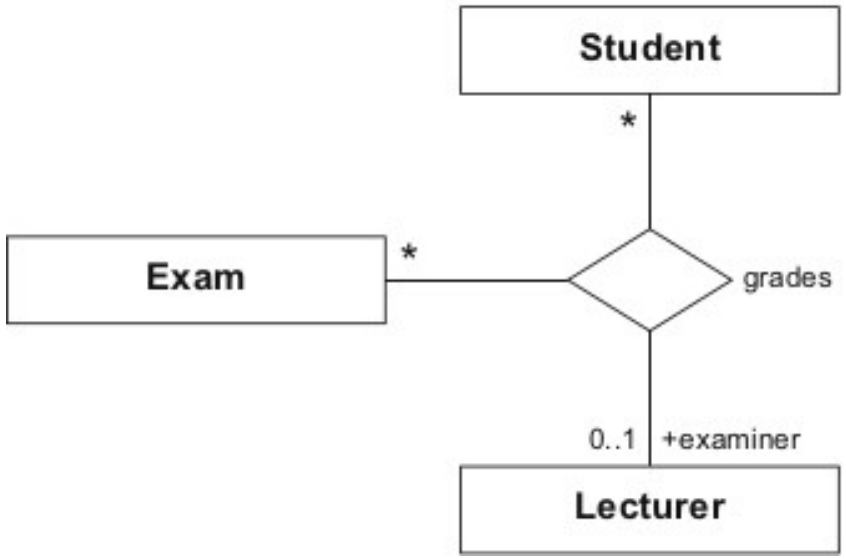


- Aparece el concepto de "**navegación**" entre clases



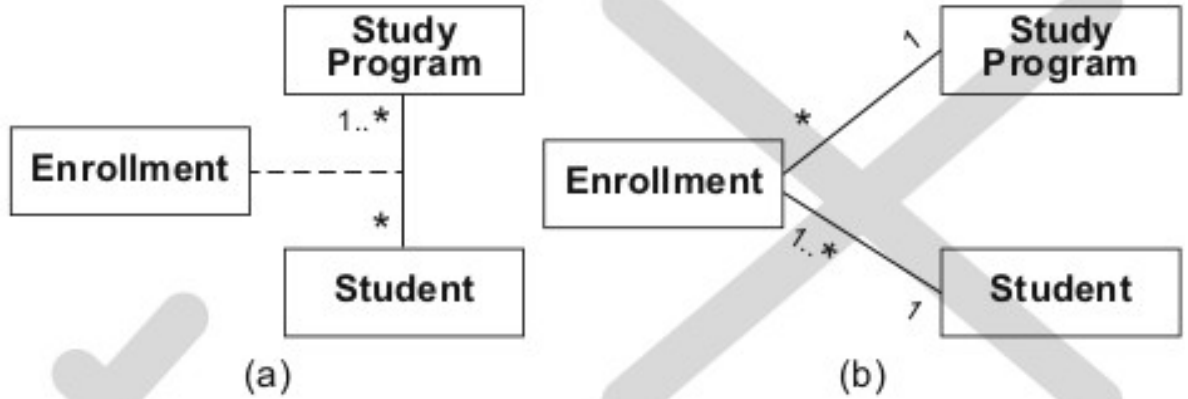
Asociaciones N-arias

- Cuando existen más de dos objetos socios en una relación.
- Se emplea un **rombo** en el centro.
- No posee dirección de navegación.



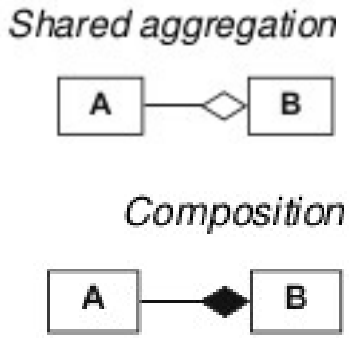
Clases Asociativas

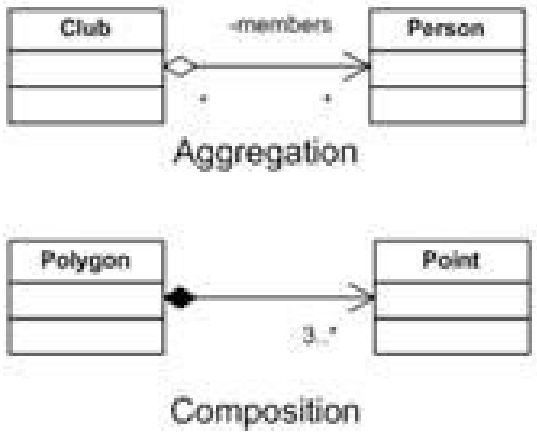
- Si desea asignar atributos u operaciones a la relación entre una o más clases en lugar de a una clase en sí, puede hacerlo usando una clase asociativa

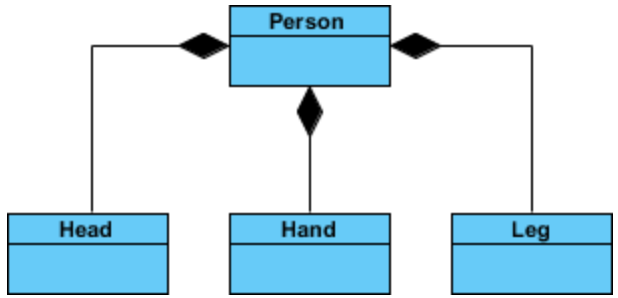


Agregaciones

- Una agregación es una forma especial de asociación que se emplea para expresar que una instancia de una clase es parte de la instancia de otra clase.
- UML diferencia entre "agregación compartida" y "composición". Ambas son transitivas.
  - La agregación compartida se representa por un rombo hueco. Expresa una pertenencia débil de las partes a un todo, lo que significa que las partes también existen independientemente del todo.
  - La composición por un rombo sólido. Expresa que una parte específica solo puede ser como máximo un objeto compuesto en un punto específico en el tiempo. Esto da como resultado una multiplicidad máxima de 1 en el extremo de agregación.





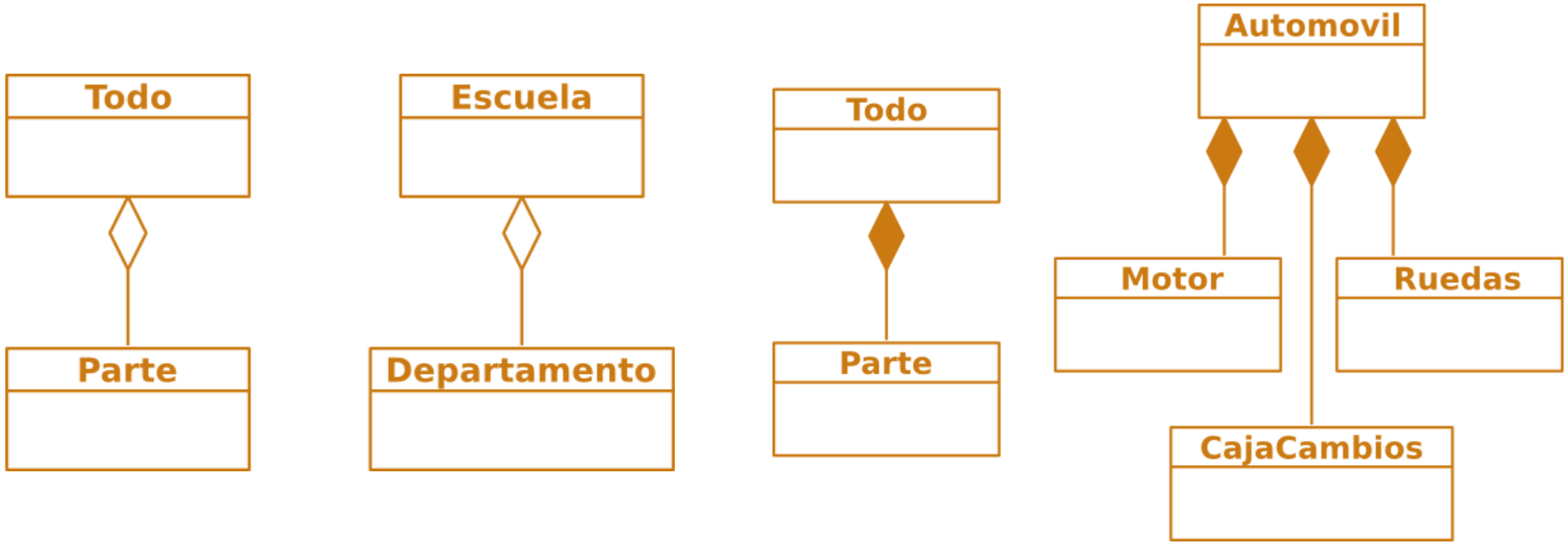


http://www.cems.uwe.ac.uk/~jsa/OODP(E-learning)/LectureMaterials/Unit4Output/unit4(1011-12)/page_14.htm

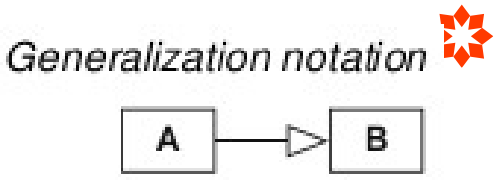
Agregación - Composición

Agregación: Es una relación en la que una de las clases representa un todo y la otra representa parte de ese todo. ¡Las partes pueden ser compartidas por varios todos!

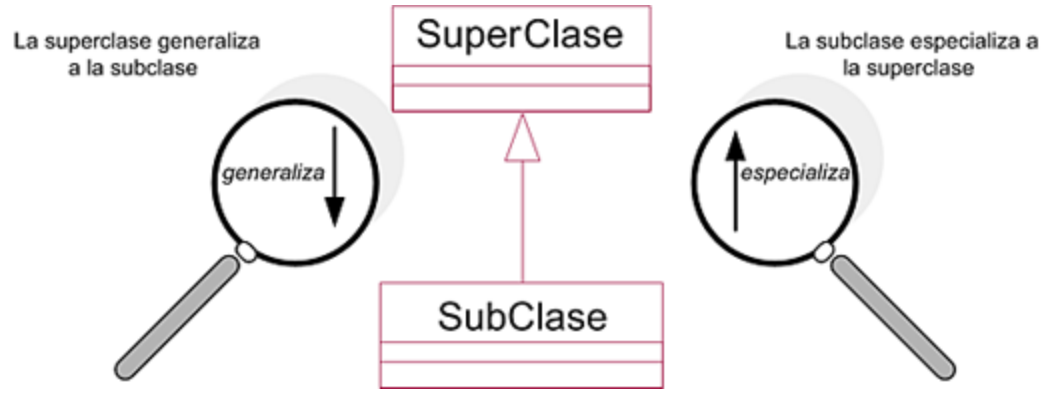
Composición: Es una forma más fuerte de la agregación, en la que el todo no puede existir sin sus partes. ¡Las partes NO pueden ser compartidas por varios todos!



Generalización

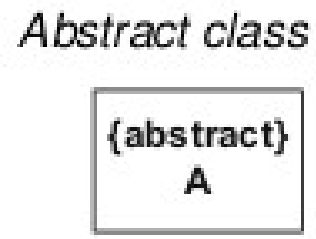


- Podemos usar una relación de generalización para resaltar puntos en común entre clases, lo que significa que ya no tenemos que definir estas características muchas varias veces.
- Podemos derivar clases más específicas a partir de las clases existentes.
  - ✓ Sinónimos: "herencia" o "relación IS-A"
- Subtype y supertype son equivalentes a subclass y superclass
- Los objetos se relacionan con sus clases mediante un "clasificador" llamado "InstanceOf" → UML permite clasificación múltiple.
- IMPORTANTE: UML tambien permite herencia múltiple



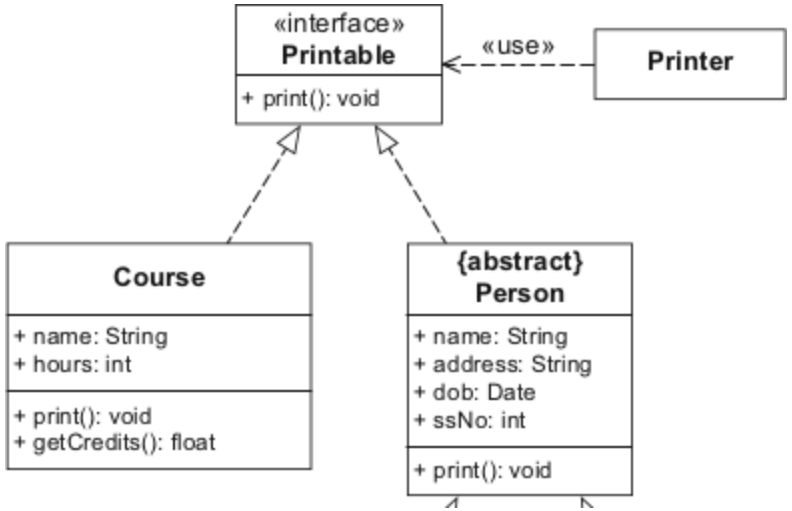
https://www.ediciones-eni.com/Open/download/351d5068-e3be-45af-b710-1f69c6199c0d/images/FIGUR63

Clases abstractas e Interfaces



- Las clases que no se pueden instanciar son **clases abstractas**.
- Se usan para describir características de las subclases: son útiles en el contexto de las relaciones de generalización.
- Una operación abstracta no tiene implementación por sí misma. Requiere la implementación en la subclase concreta

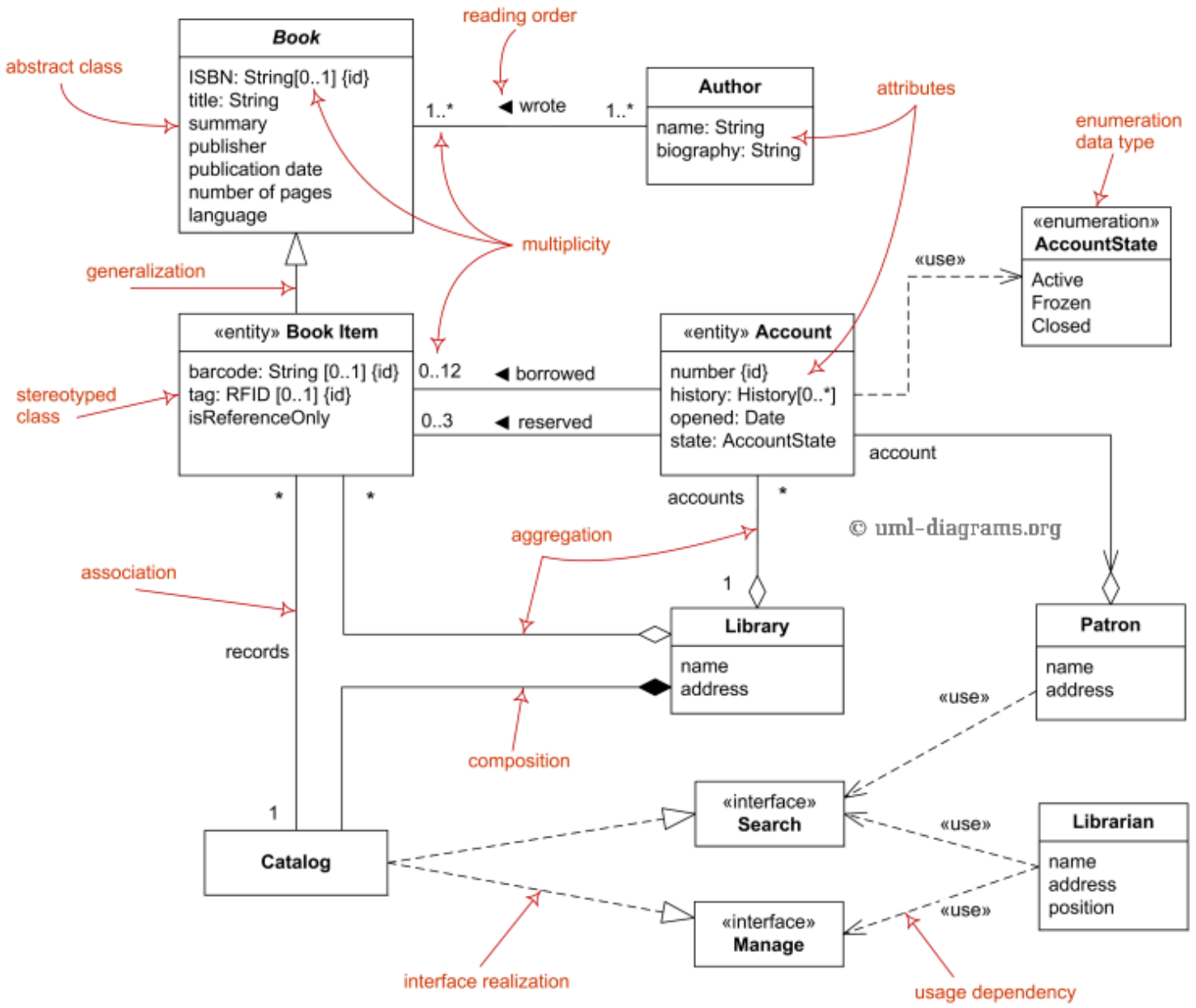
- Una **interface** tampoco tiene implementación ni instancias directas.
- La interface representa un "contrato": las clases que implementan la interface se obligan a cumplir con el comportamiento contratado.
- La interface es una relación "*LIKE A*"



Algunos tips

- A partir de los requerimientos, los nombres generalmente indican clases.
- Los valores de los atributos generalmente son adjetivos o también sustantivos.
- Las operaciones generalmente son verbos.
- Si los valores de un atributo pueden ser derivados a partir de otros atributos, entonces debe indicarse como atributo derivado.

Resumiendo...


Book
ISBN: String[0..1] {id}
title: String
summary
publisher
publication date
number of pages
language
abstract class
reading order
wrote
1..*
1..*
Author
name: String
biography: String
attributes
enumeration data type
«enumeration» AccountState
Active
Frozen
Closed
«use»
multiplicity
generalization
«entity» Book Item
barcode: String [0..1] {id}
tag: RFID [0..1] {id}
isReferenceOnly
stereotyped class
0..12
borrowed
0..3
reserved
«entity» Account
number {id}
history: History[0..*]
opened: Date
state: AccountState
account
accounts
*
*
*
1
aggregation
association
records
© uml-diagrams.org
Library
name
address
Patron
name
address
composition
«use»
1
Catalog
«interface» Search
«interface» Manage
«use»
«use»
Librarian
name
address
position
interface realization
usage dependency