

UNIDAD II

Sistemas de Información

PARTE 2

Ciclo de vida del software



Ingeniería de requerimientos o proceso de requisitos.

Los requisitos se clasifican comúnmente en una jerarquía de tres niveles, que va desde los objetivos de alto nivel hasta las especificaciones técnicas detalladas:

- ▶ **Requisitos de Negocio (o Comerciales).**
- ▶ **Requisitos de Usuario (o del Stakeholder).**
- ▶ **Requisitos del Sistema (o Técnicos)**

Una clasificación de requisitos más importantes en la ingeniería de software:

- ▶ **Requisitos funcionales**
- ▶ **Requisitos no funcionales((RNF)**

RF vs RNF

Característica	Requisitos Funcionales	Requisitos No Funcionales (RNF)
Definición	Describen lo que el sistema debe hacer (su comportamiento y funcionalidades).	Describen cómo debe ser el sistema (sus cualidades, atributos y restricciones).
Orientación	Orientados a las funciones y el comportamiento del sistema.	Orientados a la calidad, el rendimiento, la seguridad y las características del sistema.
Medición	Se miden por su capacidad para cumplir con escenarios y casos de uso específicos.	Se miden por atributos como velocidad, seguridad, usabilidad y disponibilidad.

RF vs RNF

Característica	Requisitos Funcionales	Requisitos No Funcionales (RNF)
Ejemplo	Un usuario debe poder buscar un producto por nombre y categoría.	La página de resultados debe cargar en menos de 2 segundos.
Impacto	Afectan directamente la interacción del usuario con el sistema, satisfaciendo sus necesidades explícitas.	Afectan indirectamente la experiencia del usuario, influyendo en su satisfacción y la percepción de la calidad del sistema.
Documentación	Se traducen en funcionalidades específicas del sistema.	Se traducen en criterios de rendimiento, seguridad y usabilidad.
Aceptación	Son los que más influyen en la aceptación del software por parte del usuario final.	Son clave para la competitividad, la seguridad y el crecimiento a largo plazo de la empresa.

RNF



Rendimiento: Establecen las expectativas de velocidad y capacidad.



Seguridad: Definen los mecanismos para proteger el sistema y sus datos.



Usabilidad: Describen cómo debe ser la experiencia del usuario.



Disponibilidad: Determinan cuánto tiempo el sistema debe estar operativo.

Compatibilidad: La plataforma debe ser totalmente funcional y renderizarse correctamente en las 3 últimas versiones de los navegadores web populares.

Fiabilidad: en caso de falla en la red, tiempo de recuperación y reanudación de servicios.

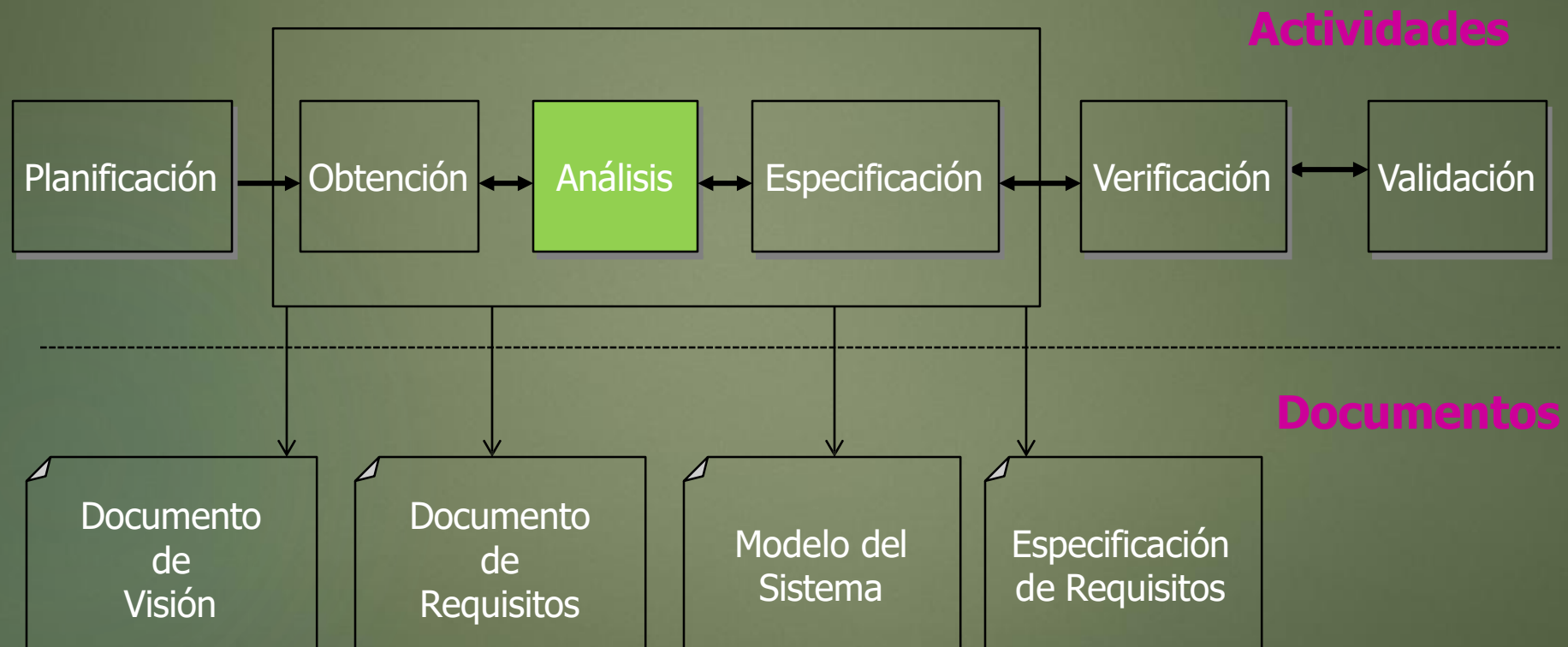
Internacionalización: soporte de idiomas adicionales, por ejemplo.

Ejercitación de Requerimientos:

Actividad 1: Caso de Estudio: Sistema de Cajero Automático

- ▶ **Objetivo**: Desarrollar habilidades de análisis (elicitación) y clasificación de requisitos.
- ▶ **Escenario**: Imagine que su equipo de desarrollo ha sido contratado para crear el software para un simulador de Cajero Automático (ATM) en una computadora personal. El sistema debe permitir a los usuarios verificar saldos, retirar efectivo y depositar fondos, interactuando con una base de datos bancaria ficticia.
- ▶ **Instrucciones**:
 - ▶ **Elicitación Guiada**: Identifiquen y documenten al menos 15 requisitos iniciales para el sistema, pensando en las necesidades del banco (negocio), del usuario que opera el cajero (cliente) y del propio sistema (técnico).
 - ▶ **Clasificación de Requisitos**: Una vez que tengan la lista, clasifiquen cada requisito de dos maneras:
 - ▶ Primero, en una de las tres jerarquías: Requisito de Negocio, Requisito de Usuario o Requisito del Sistema.
 - ▶ Segundo, como un Requisito Funcional o un Requisito No Funcional.

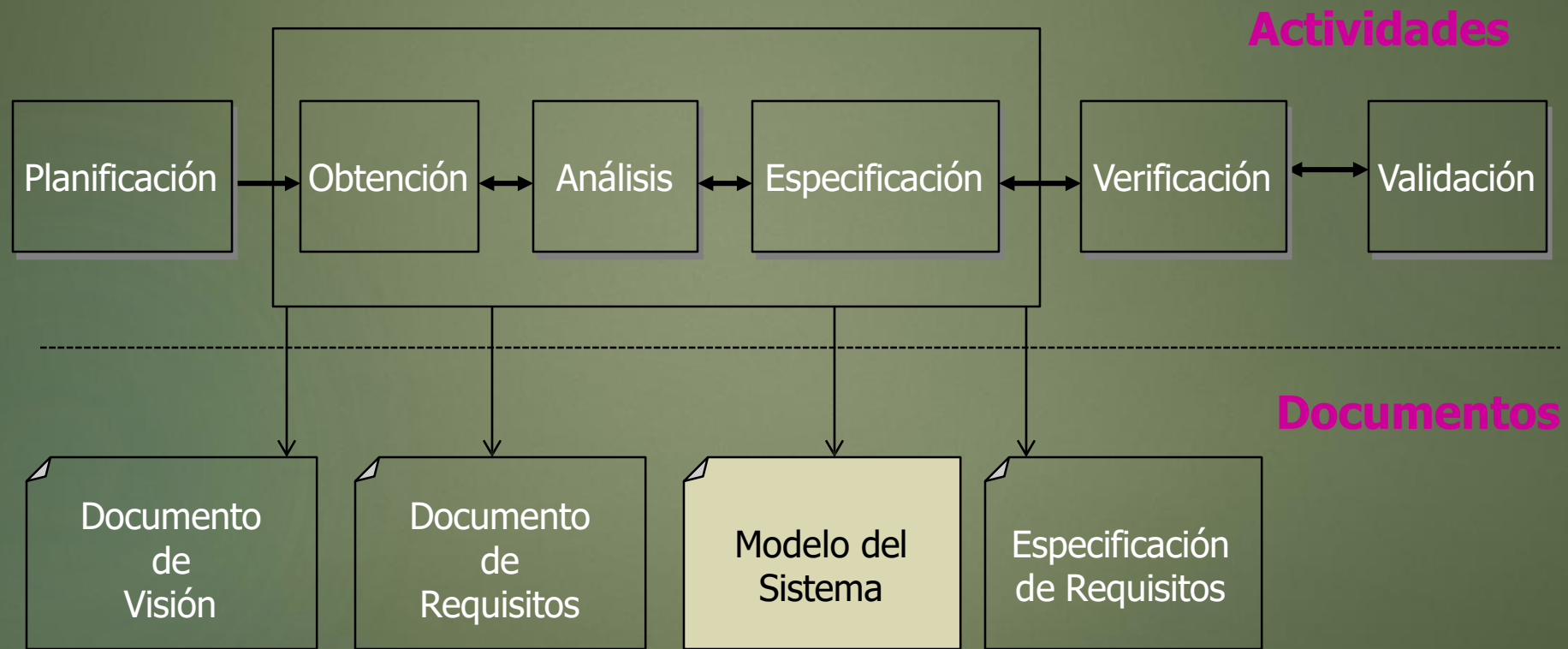
Proceso de Requerimientos



Proceso de Requerimientos

- Analizar stakeholders / clientes / usuarios
- Crear vistas
- Detallar
- Negociar prioridades
- Buscar requerimientos que faltan
- Evaluar factibilidad técnica - Prototipos
- Evaluar riesgos de requerimientos – En el Plan

Proceso de Requerimientos



Proceso de Requerimientos

MODELOS O VISTAS DEL SISTEMA

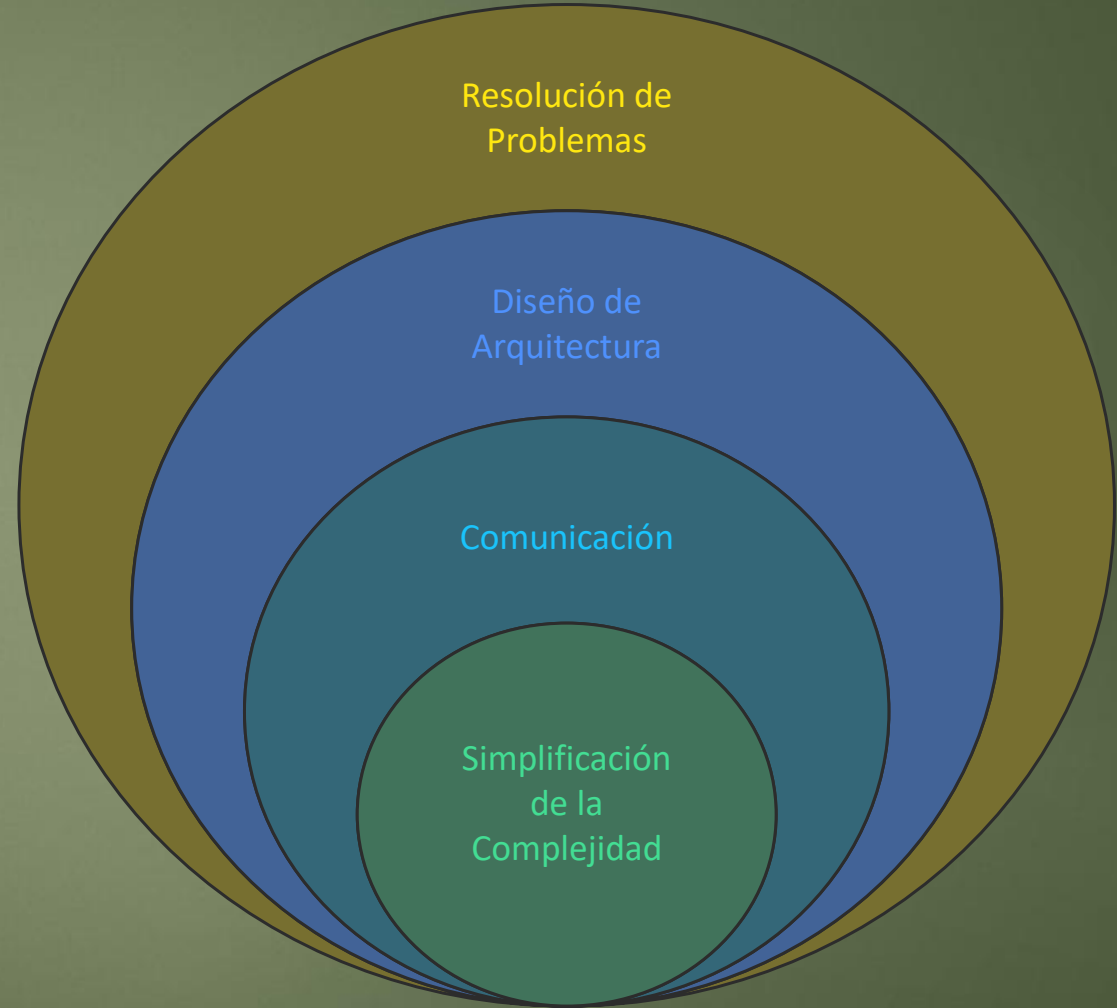
- ← Glosario
- ← Modelos gráficos
- ← Prototipos de interfaz gráfica
- ← ...
- ← ...

DIAGRAMAS

- ← BPM
- ← **UML**
 - Diagramas de Casos de Uso
 - Diagramas de Actividad
 - Diagramas de Estado
 - Modelo de dominio

Modelado Visual:

- ❑ Permite la identificación temprana de problemas.
- ❑ Ayuda a estructurar la arquitectura del sistema.
- ❑ Facilita la comprensión entre las partes interesadas.
- ❑ Simplifica sistemas complejos en visualizaciones claras.



UML: Lenguaje Unificado de Modelado

UML, por sus siglas en inglés,
Unified Modeling Language

- ▶ Es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad.
- ▶ Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema.

UML

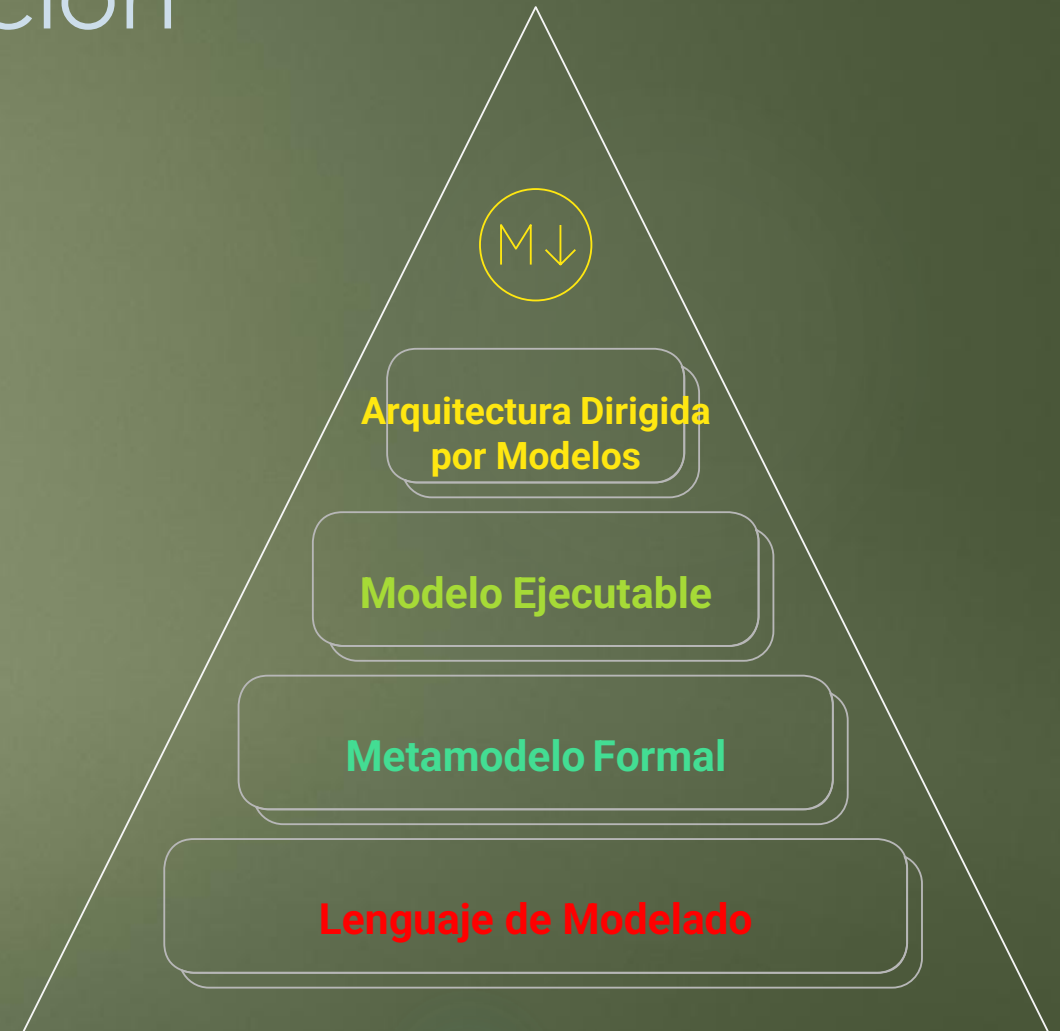
- ▶ UML ofrece un estándar para describir un "plano" del sistema (modelo)
- ▶ Es importante remarcar que UML es un "lenguaje de modelado" para especificar o para describir métodos o procesos.

UML: Un lenguaje No un lenguaje de Programación

FUNCIÓN: visualizar un sistema.

No es programación, sólo se diagrama la realidad de una utilización en un requerimiento. Mientras que, programación estructurada, es una forma de programar como lo es la orientación a objetos.

La programación orientada a objetos viene siendo un complemento perfecto de UML

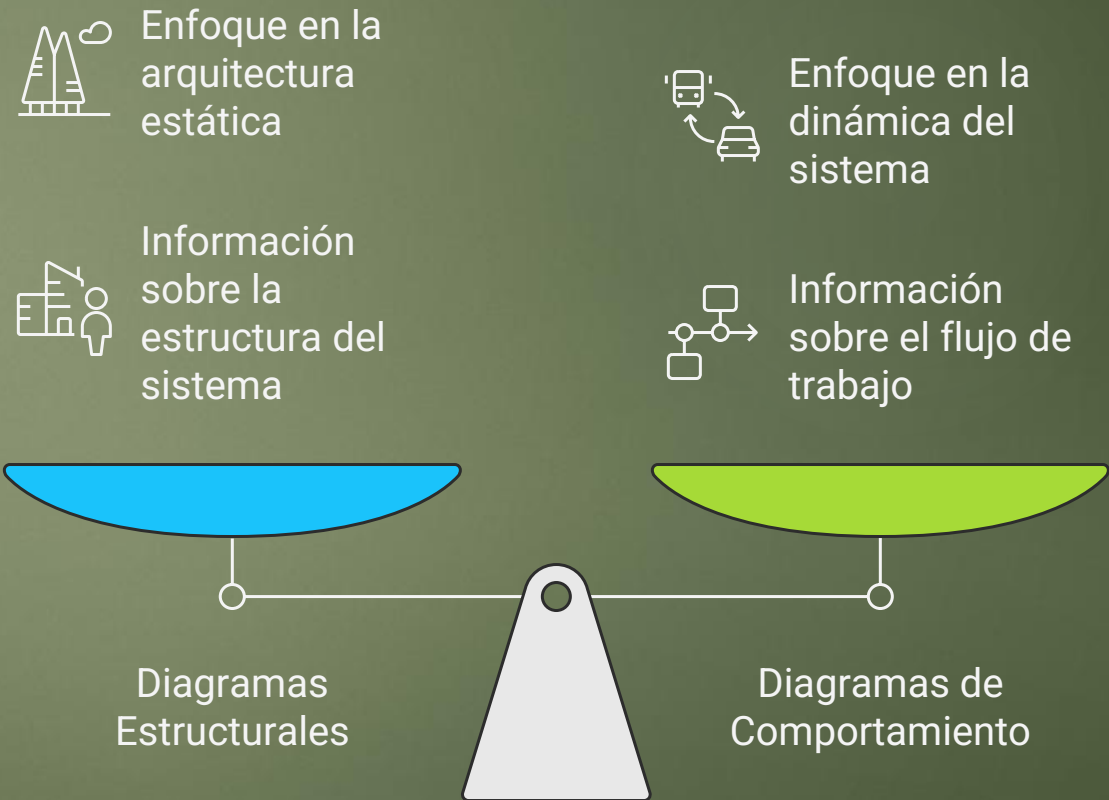


UML

- ▶ UML deriva y unifica las tres metodologías de análisis y diseño más extendidas:
 - ▶ metodología de Grady Booch para la descripción de conjuntos de objetos y sus relaciones,
 - ▶ técnica de modelado orientada a objetos de James Rum baugh (OMT: Object - Modelling Technique) y
 - ▶ La metodología de casos de uso Ivar Jacobson (OOSE: Object- Oriented Software Engineering).
- ▶ En 1997 UML 1.1 fue aprobada por la OMG convirtiéndose en la notación estándar de facto para el análisis y el diseño orientado a objetos.
- ▶ Desde el año 2005, UML es un estándar aprobado por la ISO como ISO /IEC 19501: 2005 Information technology, Open Distributed Processing Unified Modeling Language (UML).








Comparando la Estructura Estática y la Dinámica del Sistema

- UML organiza sus 15 tipos de diagramas en dos grandes categorías.



Estructura de los Diagramas UML

2.x

Categoría	Diagrama	Propósito
 Estructurales (Estáticos)	Diagrama de Clases	Muestra la estructura estática del sistema
	Diagrama de Componentes	Visualiza la organización del sistema en componentes
	Diagrama de Estructura Compuesta	Detalla la estructura interna de una clase
	Diagrama de Despliegue	Representa la disposición física de los nodos
	Diagrama de Objetos	Muestra instancias de clases en un momento
	Diagrama de Paquetes	Agrupar elementos de modelado en paquetes
	Diagrama de Perfiles	Extiende UML a dominios específicos

Estructura estática

- ▶ Los conceptos de la aplicación son modelados como CLASES, cada una de las cuales describe un conjunto de objetos que almacenan información y se comunican para implementar un comportamiento.
- ▶ La información que almacena es modelada como atributos; La estructura estática se expresa con diagramas de clases y puede usarse para generar la mayoría de las declaraciones de estructuras de datos en un programa.

Estructura de los Diagramas UML

2.x

Categoría	Diagrama	Propósito
 Comportamental (Dinámico)	Diagrama de Actividades	Describe el flujo de trabajo de un proceso
	Diagrama de Máquinas de Estado	Modela el ciclo de vida del objeto a través de estados
	Diagrama de Casos de Uso	Muestra la funcionalidad del sistema en términos
	Diagrama de Interacción	Modela el control y el flujo de datos
	Diagrama de Secuencia	Muestra la interacción entre objetos
	Diagrama de Comunicación	Muestra la colaboración entre objetos
	Diagrama de Tiempos	Muestra los cambios de estado de los objetos a lo largo del tiempo
	Diagrama de Visión General de Interacción	Combina diagramas de actividad y de secuencia

Comportamiento dinámico

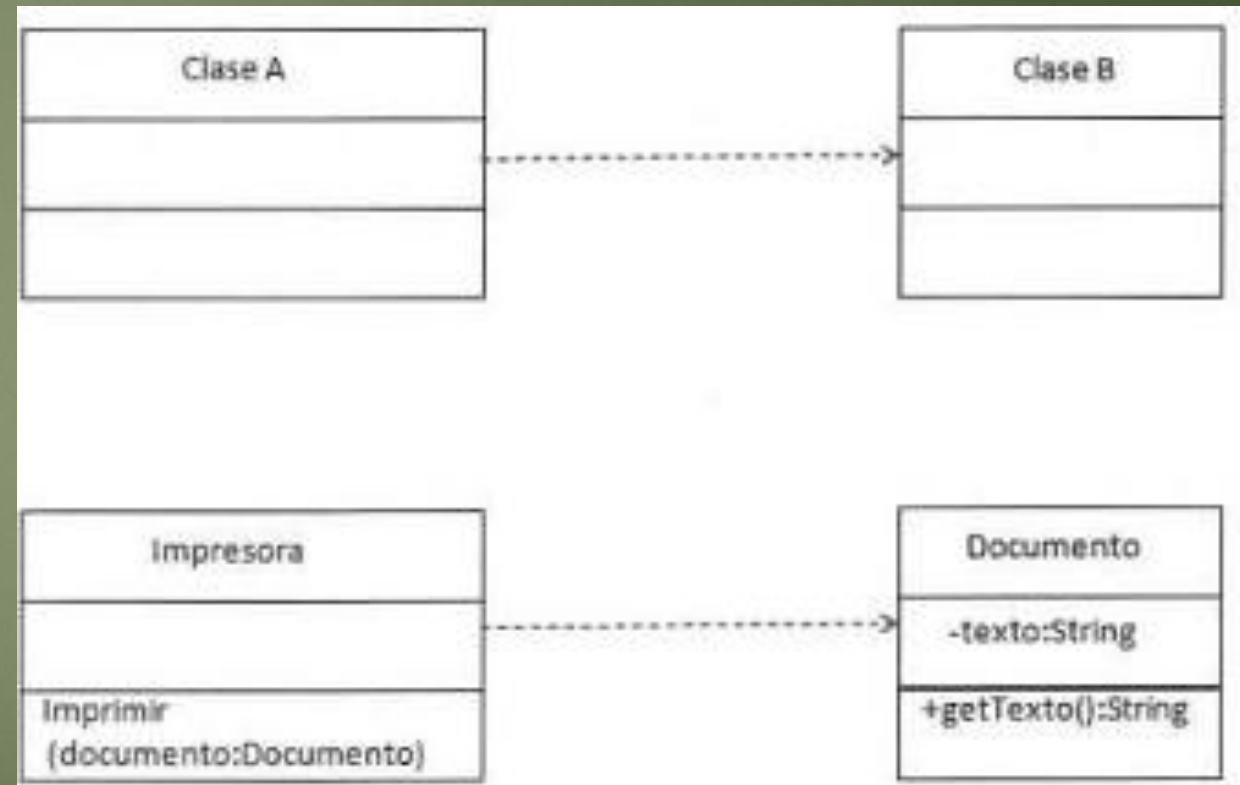
- ▶ Hay dos formas de modelar el comportamiento, una es la historia de la vida de un objeto y la forma como interactúa con el resto del mundo y la otra es por los patrones de comunicación de un conjunto de objetos conectados, es decir la forma en que interactúan entre sí.
- ▶ La visión de un objeto aislado es una máquina de estados que muestra la forma en que el objeto responde a los eventos en función de su estado actual. La visión de la interacción de los objetos se representa con los enlaces entre objetos junto con el flujo de mensajes y los enlaces entre ellos. Este punto de vista unifica la estructura de los datos, el control de flujo y el flujo de datos.

Organización del modelo

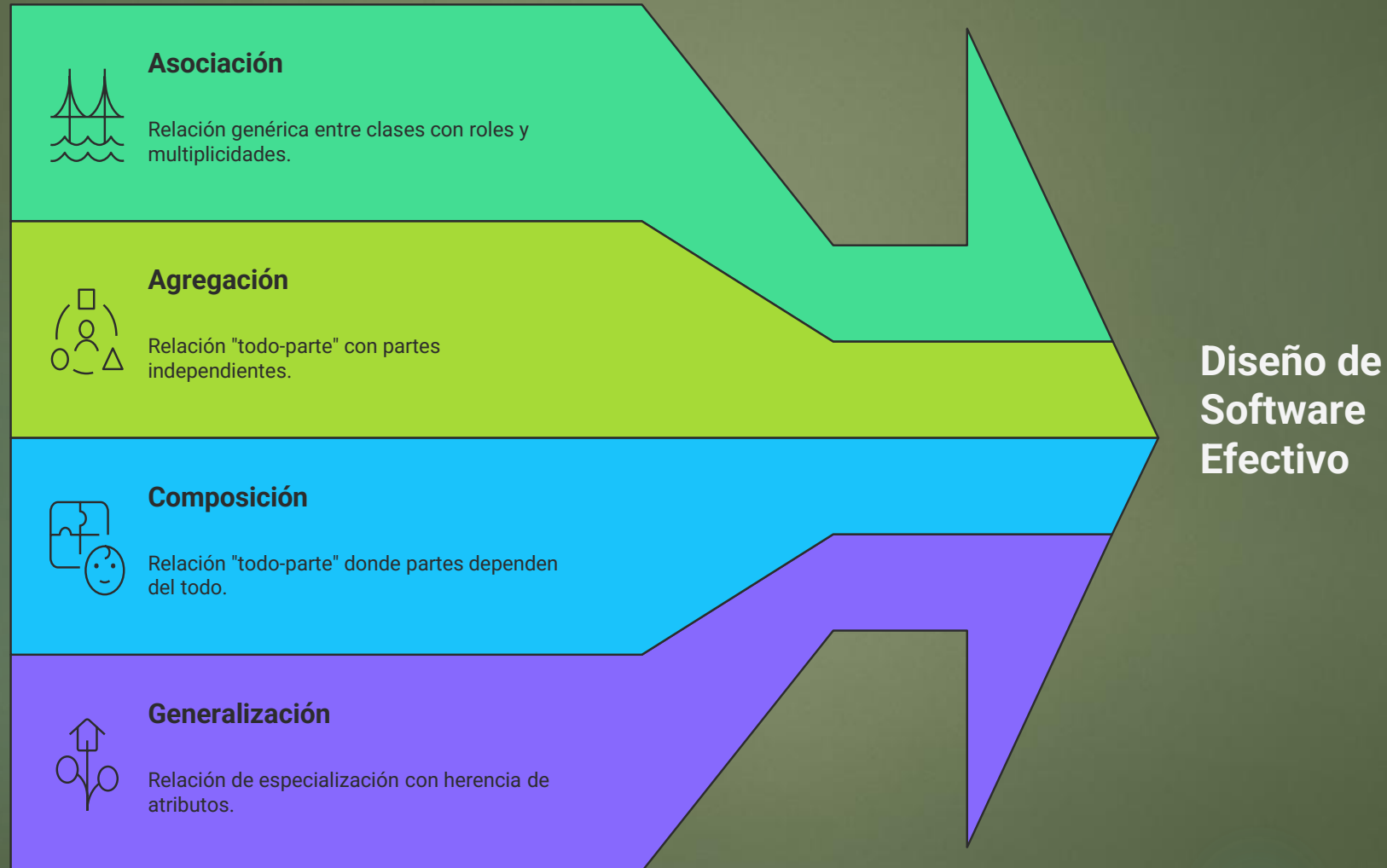
- ▶ La información del modelo debe ser dividida en piezas coherentes, para que los equipos puedan trabajar en las diferentes partes de forma concurrente.
- ▶ El conocimiento humano requiere que se organice el contenido del modelo en paquetes de tamaño modesto.
- ▶ Los paquetes son unidades organizativas, jerárquicas y de propósito general de los modelos de UML. Pueden usarse para almacenamiento, control de acceso, gestión de la configuración y construcción de bibliotecas que contengan fragmentos de código reutilizable.

Relaciones: Dependencias

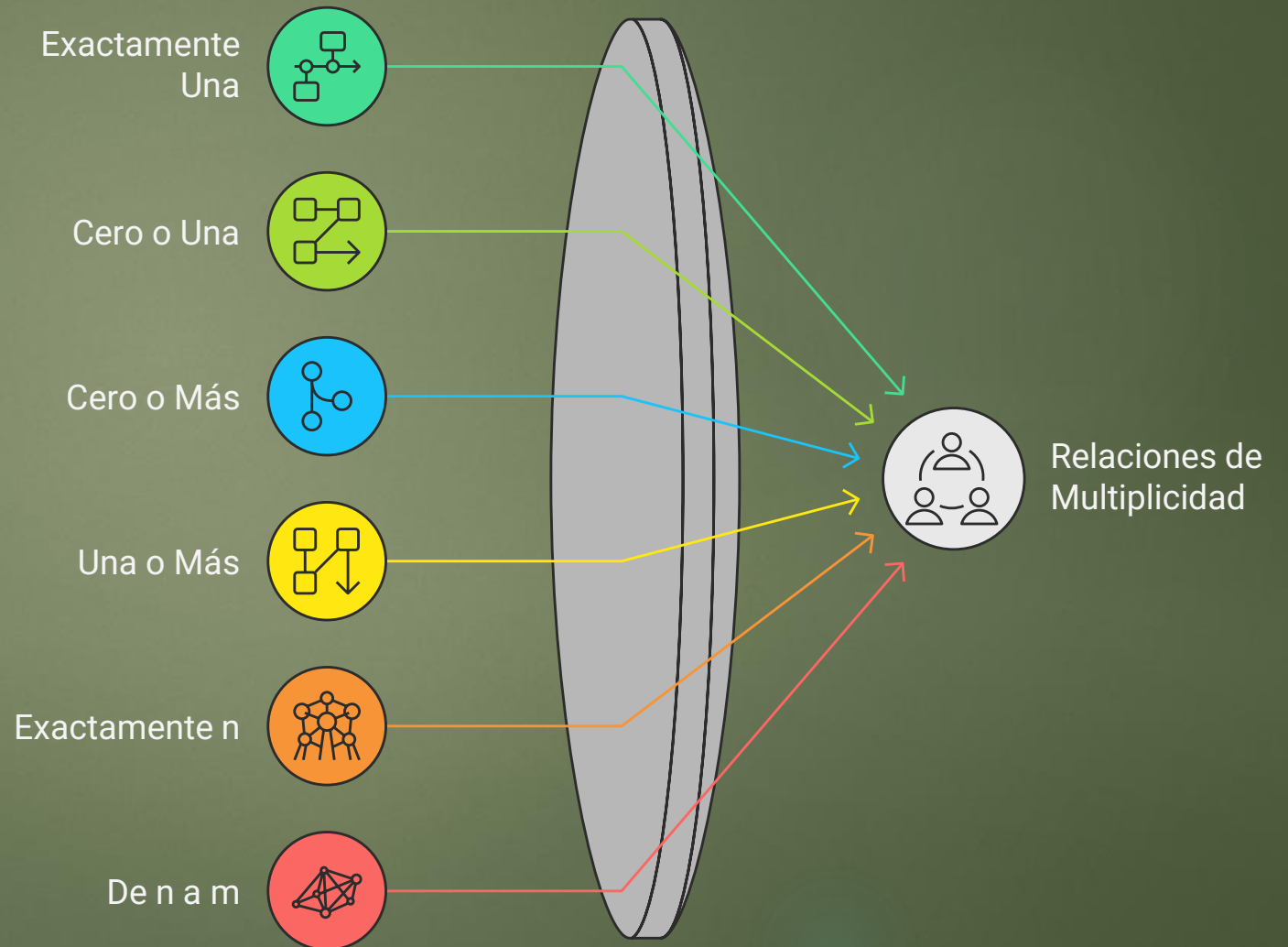
- ▶ Es una relación semántica entre dos elementos en la cual un cambio a un elemento (el elemento independiente) puede afectar a la semántica del otro elemento (elemento dependiente).
- ▶ Se representa como una línea discontinua, posiblemente dirigida, que a veces incluye una etiqueta.
- ▶ En el diagrama se puede ver una relación de dependencia entre dos clases. La clase A usa la clase B. La clase impresora usa la clase documento.



Relaciones:

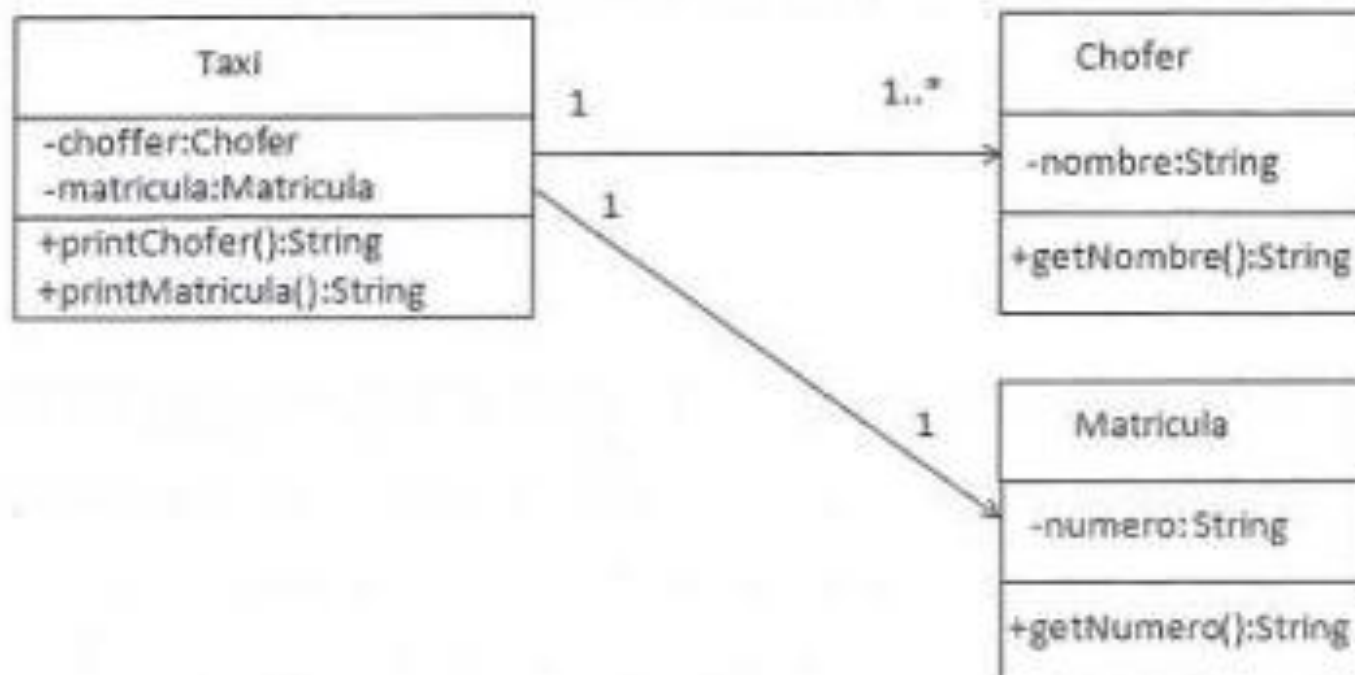
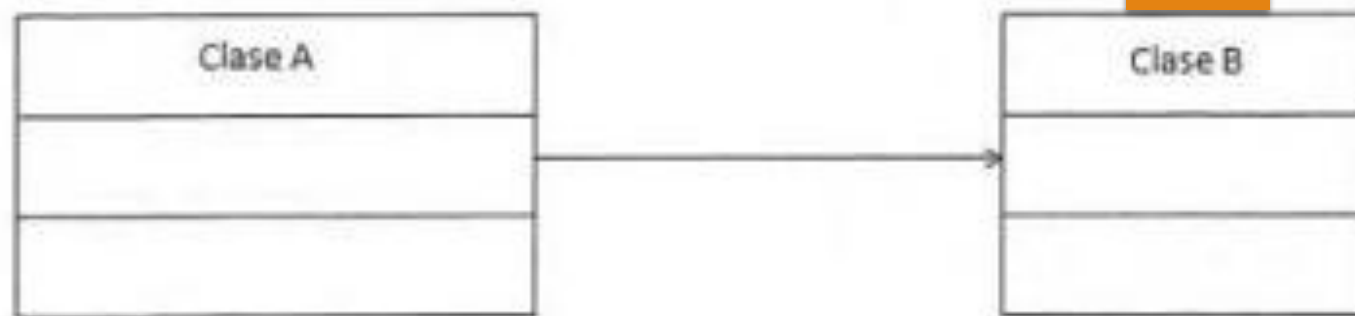


Multiplicidad en Relaciones



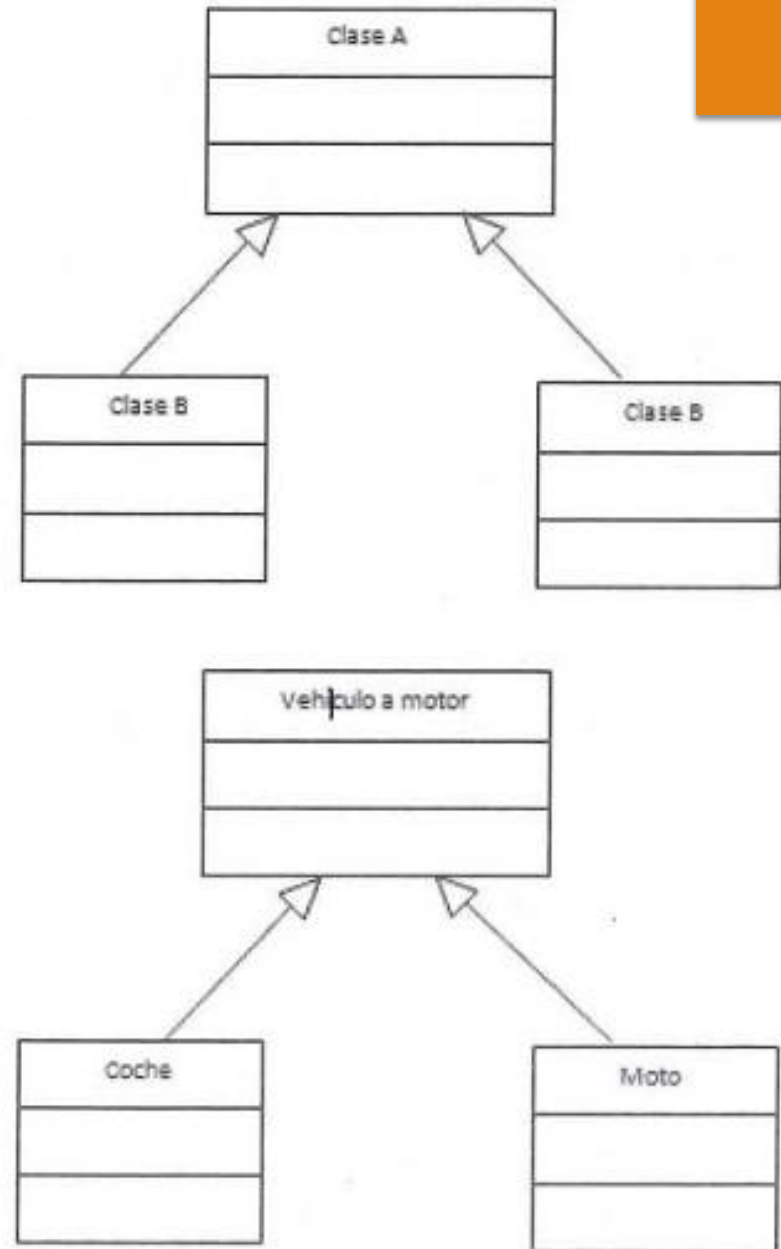
Relaciones: Asociativa

- ▶ La clase A está asociada a la clase B.
- ▶ La clase A necesita la clase B.
- ▶ La clase Taxi necesita la clase Chofer y la clase Matrícula. Necesita 1 o varios chóferes y un sola matrícula.



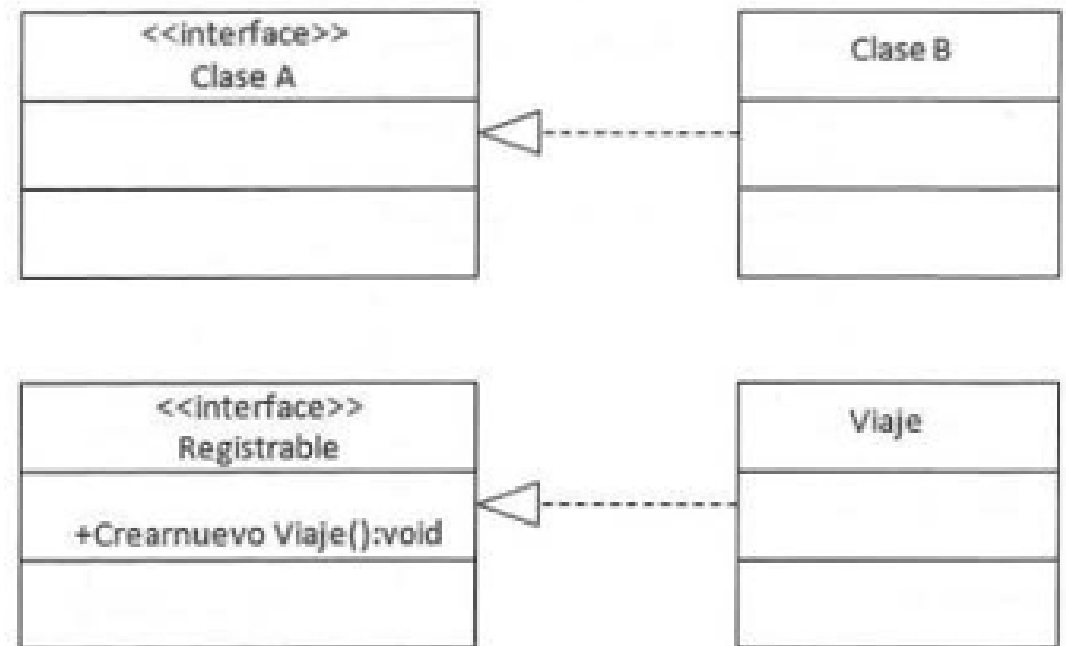
Relaciones: Generalización

- ▶ Es una relación de especialización / generalización en la cual los objetos del elemento especializado (el hijo) pueden sustituir a los objetos del elemento general (el padre).
- ▶ De esta forma, el hijo comparte la estructura y el comportamiento del padre.
- ▶ Gráficamente, la generalización se representa con una línea con punta de flecha vacía.



Relaciones de Realización

- ▶ Es una relación semántica entre clasificadores, donde un clasificador especifica un contrato que otro clasificador garantiza que cumplirá.
- ▶ Se pueden encontrar relaciones de realización en dos sitios: entre interfaces y las clases y componentes que las realizan, y entre los casos de uso y las colaboraciones que los realizan.
- ▶ La realización se representa como una mezcla entre la generalización y la dependencia, esto es, una línea discontinua con una punta de flecha vacía.

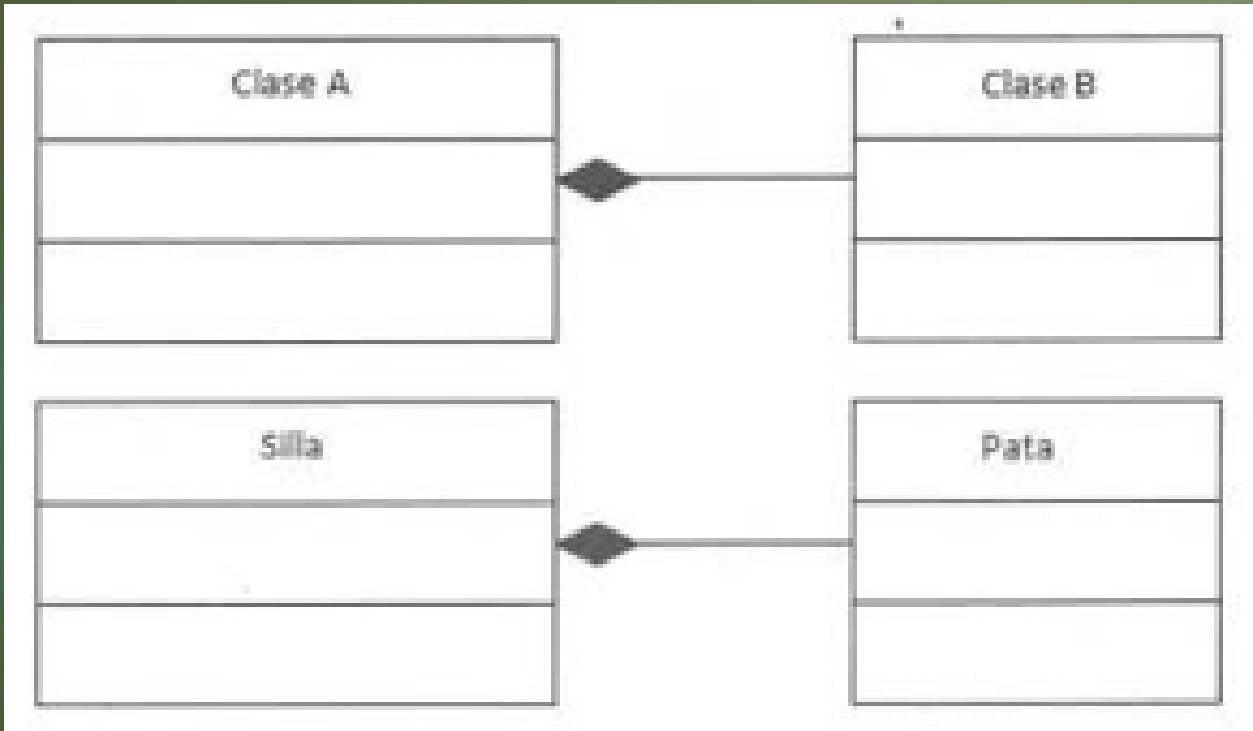


Relaciones: Agregación

- ▶ Es muy similar a la relación de Asociación solo varía en la multiplicidad ya que en lugar de ser una relación "uno a uno" es de "uno a muchos".
- ▶ Se representa con una flecha que parte de una clase a otra en cuya base hay un rombo de color blanco.
- ▶ En la figura se puede ver una relación de agrupación. La Clase A agrupa varios elementos del tipo Clase B. La clase Agenda agrupa a varios Contactos.



Relaciones: Composición



- ▶ Similar a la relación de Agregación solo que la Composición es una relación mas fuerte.
- ▶ Aporta documentación conceptual ya que es una "relación de vida", es decir, el tiempo de vida de un objeto está condicionado por el tiempo de vida del objeto que lo incluye.
- ▶ En la figura se puede ver una relación de composición o agrupación.



“

DIAGRAMA DE CASOS DE USOS

”

MODELO ESTÁTICO:
FUNDAMENTOS DEL DISEÑO DE SOFTWARE

Diagrama de Casos de Uso

- ▶ Diagramas de casos de uso ilustran la funcionalidad proporcionada por una sistema.
- ▶ Los diagramas de casos de uso describen las relaciones y las de dependencias entre un grupo de casos de uso y los actores participantes en el proceso.

Diagrama de Casos de Uso

- Los diagramas de casos de uso sirven para facilitar la comunicación con los futuros usuarios del sistema, y con el cliente, y resultan especialmente útiles para determinar las características necesarias que tendrá el sistema. En otras palabras, los diagramas de casos de uso describen **qué** es lo que debe hacer el sistema, pero no cómo.

Diagrama de Casos de Uso

- ▶ Un actor es una entidad externa (de fuera del sistema) que interacciona con el sistema participando (y normalmente iniciando) en un caso de uso.
- ▶ Los actores pueden ser gente real (por ejemplo, usuarios del sistema), otros sistemas o eventos.
- ▶ Los actores no representan a personas físicas o a sistemas, sino su rol.

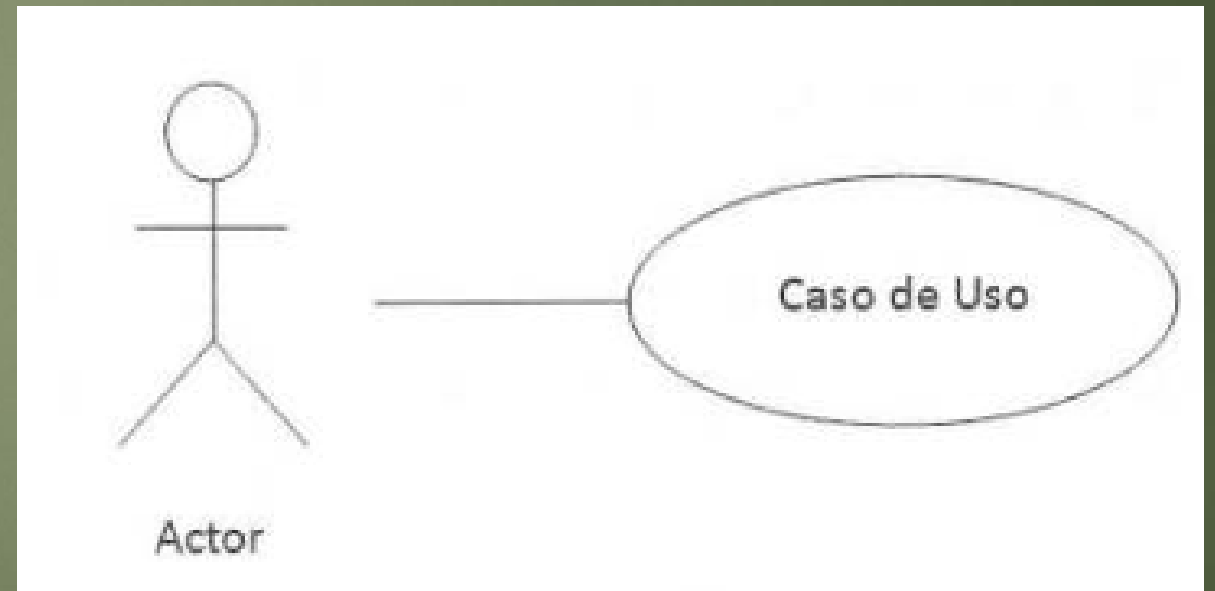


Diagrama de Casos de Uso

- En la figura se puede ver un diagrama de casos de uso de un cajero automático con dos actores: un cliente y un empleado de banco.

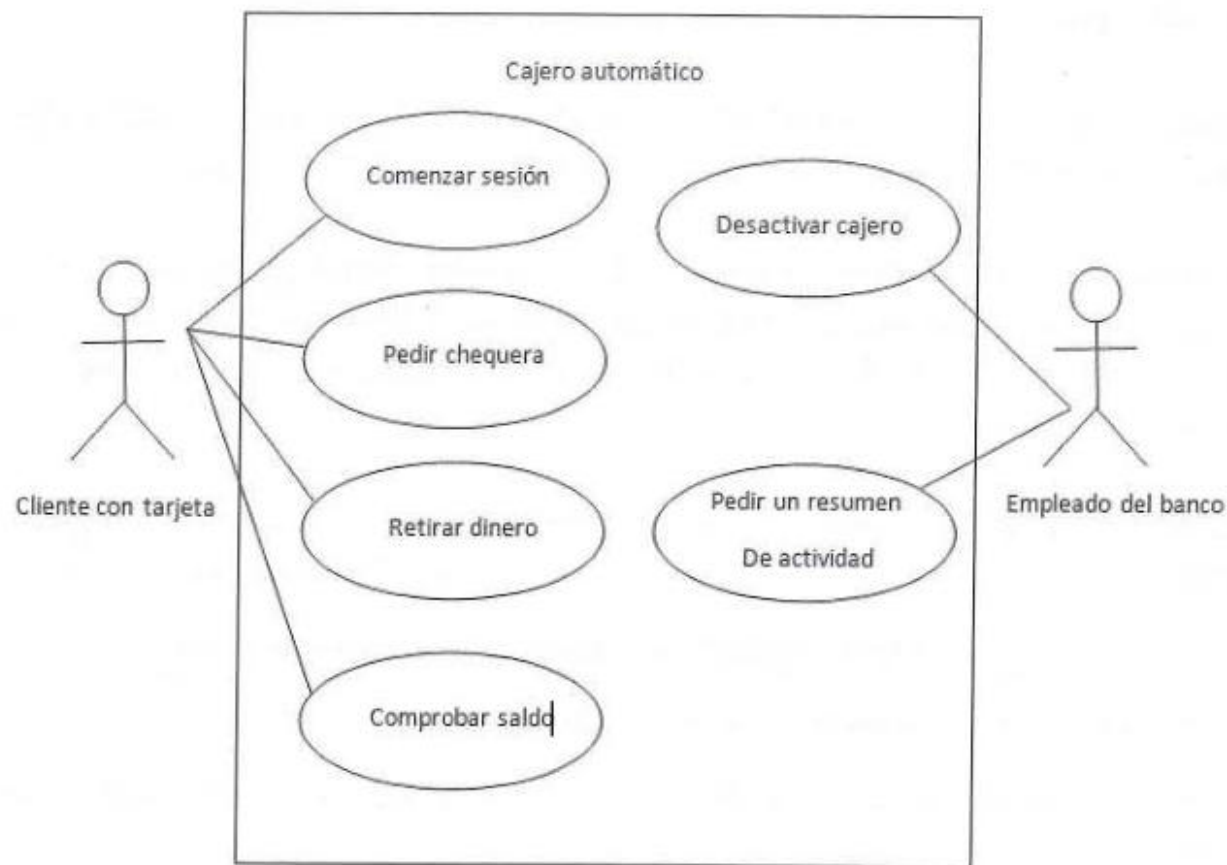
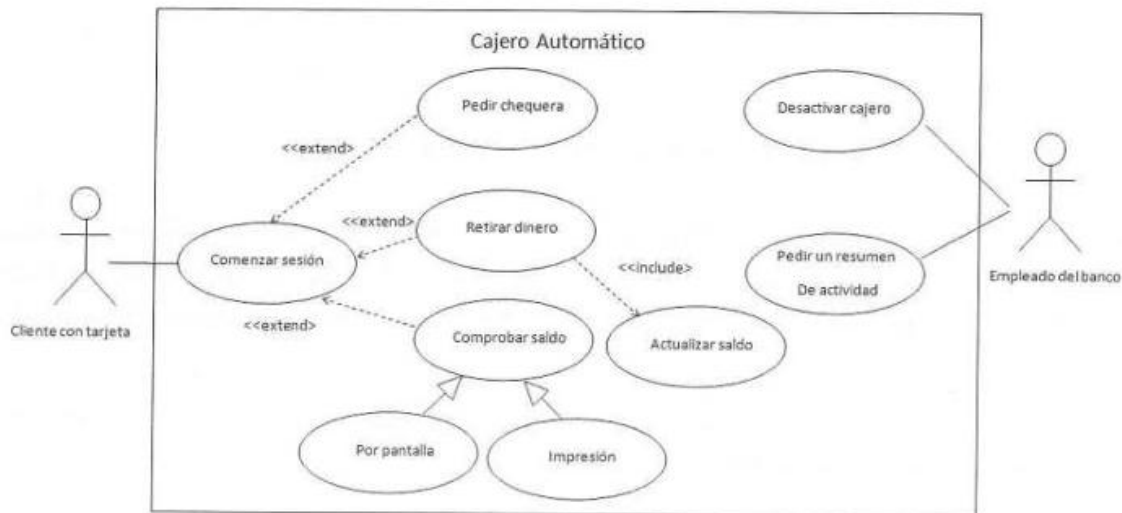


Diagrama de Casos de Uso



- ▶ Los tres tipos de relaciones más comunes entre casos de uso son:
- ▶ «include» que especifica una situación en la que un caso de uso tiene lugar dentro de otro caso de uso.
- ▶ «extends» que especifica que en ciertas situaciones, o en algún punto (llamado punto de extensión) un caso de uso será extendido por otro.
- ▶ Generalización que especifica que un caso de uso hereda las características del "super" caso de uso, y puede volver a especificar algunas o todas ellas de una forma muy similar a las herencias entre clases.

Diagrama de Casos de Uso

- ▶ Las descripciones de casos de uso son reseñas textuales del caso de uso.
- ▶ Normalmente tienen el formato de una nota o un documento relacionado de alguna manera con el caso de uso, y explica los procesos o actividades que tienen lugar en el caso de uso.

Título:	
Actores involucrados:	
Objetivo:	
Precondición:	
<u>Poscondición:</u>	
Escenario Principal	Flujo Alternativo
1. El Usuario	1.1



“

DIAGRAMA DE CLASES

”

MODELO ESTÁTICO:
FUNDAMENTOS DEL DISEÑO DE SOFTWARE

Diagrama de Clase

- ▶ Los diagramas de clases muestran las diferentes clases que componen un sistema y cómo se relacionan unas con otras.
- ▶ Se dice que los diagramas de clases son diagramas estáticos porque muestran las clases, junto con sus métodos y atributos, así como las relaciones estáticas entre ellas: **qué** clases conocen a qué otras clases o qué clases son parte de otras clases, pero no muestran los métodos mediante los que se invocan entre ellas.

DIAGRAMA DE CLASES

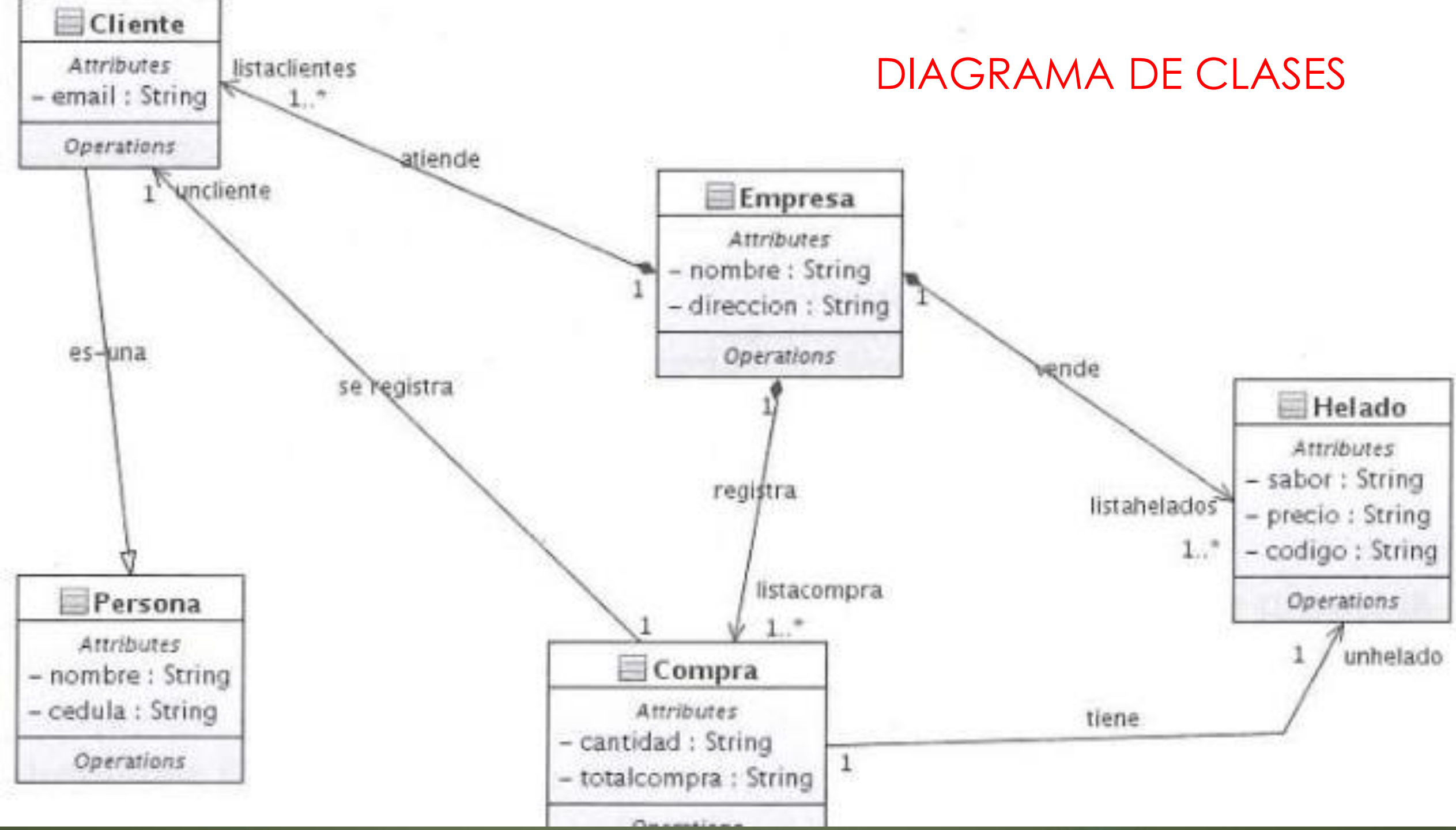


Diagrama de Clases: CLASES

- ▶ Una clase define los atributos y los métodos de una serie de objetos. Todos los objetos de esta clase (instancias de esa clase) tienen el mismo comportamiento y el mismo conjunto de atributos (cada objeto tiene el suyo propio).
- ▶ En las clases están representadas por rectángulos, con el nombre de la clase, y también pueden mostrar atributos y operaciones de la clase en otros dos «compartimentos» dentro del rectángulo.

CLASES

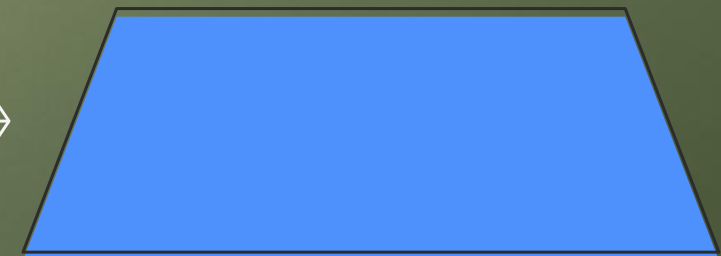
Operaciones
Métodos o funciones de la clase



Atributos
Visibilidad, nombre y tipo de atributos



Nombre de la clase
Nombre en negrita y camel case



CLASES

- ▶ La clase esta formada por atributos y operaciones o métodos • Atributos.
- ▶ En UML, los ATRIBUTOS se muestran al menos con su nombre, y también pueden mostrar su tipo, valor inicial y otras propiedades.
- ▶ Los atributos aparecen calificados en el diagrama dependiendo de su acceso como:
 - ▶ + Indica atributos públicos
 - ▶ # Indica atributos protegidos
 - ▶ - Indica atributos privados

CLASES

OPERACIONES

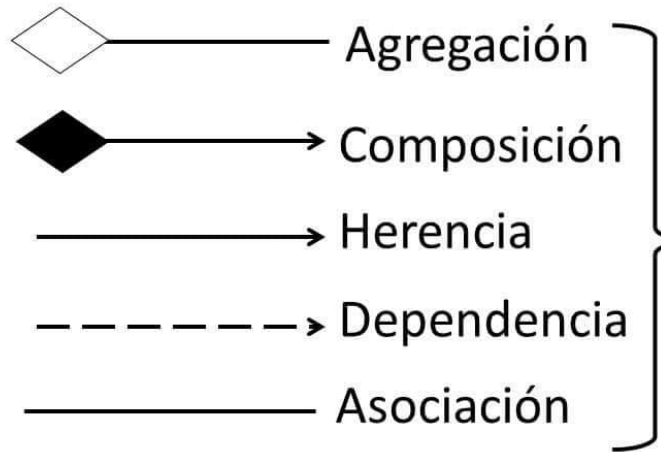
- ▶ Las operaciones (métodos) también se muestran al menos con su nombre, y pueden mostrar sus parámetros y valores de retorno.
- ▶ Las operaciones, al igual que los atributos, aparecen calificados en el diagrama dependiendo de su acceso como:
 - ▶ + Indica operaciones públicas
 - ▶ # Indica operaciones protegidas
 - ▶ - Indica operaciones privadas

Asociaciones de clases

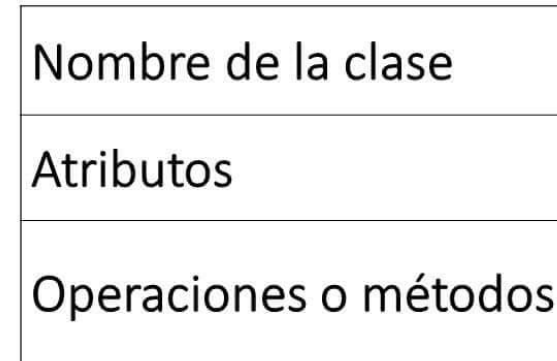
Las clases se puede relacionar, (estar asociadas), con otras de las siguientes posibles formas:

- ▶ Generalización
- ▶ Asociación
- ▶ Realización
- ▶ Agregación
- ▶ Composición

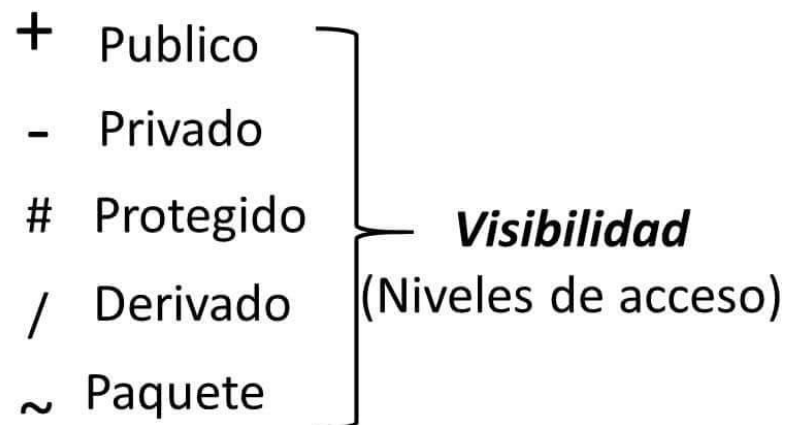
Elementos y símbolos en los diagramas de clases UML



Interacciones

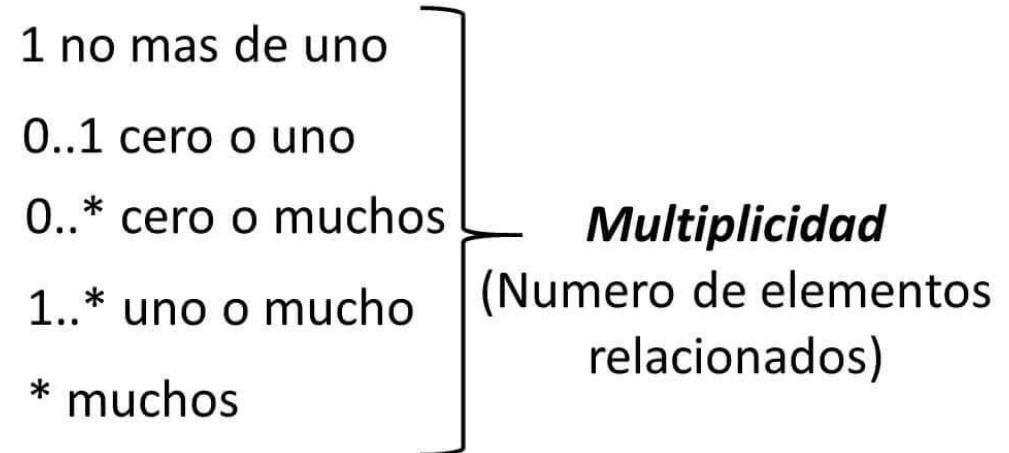


Clase



Visibilidad

(Niveles de acceso)



Multiplicidad

(Numero de elementos relacionados)

Diagrama de clases

Los diagramas de clases pueden contener más componentes aparte de clases. Estos pueden ser:

► Interfaces

Las interfaces son clases abstractas, esto es, instancias que no pueden ser creadas directamente a partir de ellas. Pueden contener operaciones, pero no atributos. Las clases pueden heredarse de las interfaces pudiendo así realizarse instancias a partir de estos diagramas.

► Tipo de datos

Los tipos de datos son primitivas incluidas en algunos lenguajes de programación. Algunos ejemplos son: bool y float. No pueden tener relación con clases, pero las clases sí pueden relacionarse con ellos.

► Enumeraciones

Las enumeraciones son simples listas de valores. Un ejemplo típico de esto sería una enumeración de los días de la semana. Al igual que los tipos de datos, no pueden relacionarse con las clases, pero las clases sí pueden hacerlo con ellos.

► Paquetes

Los paquetes, en lenguajes de programación, representan un espacio de nombres en un diagrama se emplean para representar partes del sistema que contienen más de una clase, incluso cientos de ellas.



BUENA SEMANA...