Modelado de Software usando UML

Material Completo del Curso

Total de Unidades: 5

31 de December de 2025

Índice de Contenidos

Modelado de Software usando UML

Unidad 01: Fundamentos de UML

Notas del Presentador:
Bienvenidos a la primera unidad del curso de Modelado de Software usando UML. Esta unidad establece los fundamentos históricos y conceptuales que son esenciales para comprender el resto del curso.

Competencias y Resultados de Aprendizaje

Competencia Específica:

CE15: Gestionar estrategias de mantenimiento de software, aplicando los planes de mejora de éste de manera adecuada, siguiendo los principios de evolución con el fin de preservar la calidad a lo largo del tiempo.

Resultados de Aprendizaje:

- Comprender la historia y evolución de UML
- Identificar los diferentes tipos de diagramas UML
- Reconocer la importancia del modelado en el desarrollo de software
- Entender el contexto histórico del surgimiento de UML
- Clasificar los diagramas según su propósito

Notas del Presentador:
Estos resultados están alineados con el programa académico. Asegúrate de que los estudiantes comprendan que UML no es solo una herramienta, sino un lenguaje estándar para la comunicación en el desarrollo de software.

Agenda de la Unidad

1. Historia de UML
2. Orígenes y evolución
3. Los Tres Amigos: Booch, Rumbaugh y Jacobson
4. Estándares OMG (Object Management Group)
5. Tipos de diagramas UML
6. Clasificación: Estructurales vs. Comportamiento
7. Importancia del modelado en software
8. Práctica guiada

Notas del Presentador:
Duración estimada: 2 horas. La primera parte será teórica (60 min), luego práctica guiada (40 min), y finalmente evaluación (20 min).

Historia de UML: Orígenes

Antecedentes (Años 80-90)

- **Década de 1980:** Surgimiento de la programación orientada a objetos
- **Diferentes metodologías:** Cada metodólogo tenía su propia notación
- **Problema:** Falta de estandarización causaba confusión
- **Necesidad:** Un lenguaje común para modelar sistemas orientados a objetos

> **Contexto:** En los años 90, había más de 50 métodos de modelado diferentes, lo que dificultaba la comunicación entre equipos y organizaciones.

Notas del Presentador:
Es importante contextualizar que antes de UML, cada empresa y metodólogo usaba sus propias notaciones, lo que generaba problemas de comunicación.

Los Tres Amigos: Creadores de UML

Grady Booch

- Desarrolló el **Método Booch**
- Enfocado en diseño y arquitectura de sistemas
- Contribución: Diagramas de clases y objetos

James Rumbaugh

- Creador de **OMT (Object Modeling Technique)**
- Especialista en análisis de sistemas
- Contribución: Modelado de datos y comportamiento

Ivar Jacobson

- Desarrollador de **OOSE (Object-Oriented Software Engineering)**
- Pionero en casos de uso
- Contribución: Diagramas de casos de uso

> **1994-1995:** Los tres metodólogos se unen en Rational Software para crear un lenguaje unificado.

Evolución de UML

Hitos Importantes:

- **1995:** Primera propuesta de UML (versión 0.8)
- **1997:** UML 1.0 - Adoptado por OMG
- **1998:** UML 1.1 - Primera versión estándar oficial
- **2003:** UML 1.5 - Mejoras en extensiones
- **2005:** UML 2.0 - Revisión mayor con nuevos diagramas
- **2015:** UML 2.5 - Versión actual, simplificada y mejorada

> **OMG (Object Management Group):** Organización internacional que mantiene y actualiza el estándar UML desde 1997.

Notas del Presentador:
UML 2.0 fue una revisión importante que agregó nuevos tipos de diagramas y mejoró la semántica. UML 2.5 simplificó y corrigió problemas de versiones anteriores.

¿Qué es UML?

Definición:

UML (Unified Modeling Language) es un lenguaje de modelado visual estándar para especificar, visualizar, construir y documentar los artefactos de sistemas de software.

Características Principales:

- **Visual:** Usa diagramas y símbolos gráficos
- **Estándar:** Aceptado internacionalmente (ISO/IEC 19505)
- **Independiente de lenguaje:** No está atado a un lenguaje de programación
- **Orientado a objetos:** Basado en conceptos de POO
- **Completo:** Cubre todo el ciclo de vida del software

> **Importante:** UML es un LENGUAJE, no una metodología. Define CÓMO representar, no QUÉ proceso seguir.

Tipos de Diagramas UML

Clasificación Principal:

Diagramas Estructurales (6 tipos)

Representan la estructura estática del sistema

Diagramas de Comportamiento (7 tipos)

Representan el comportamiento dinámico del sistema

> **Total:** UML 2.5 define 14 tipos de diagramas oficiales, organizados en dos categorías principales.

Diagramas Estructurales

Representan la **estructura estática** del sistema: qué elementos existen y cómo se relacionan.

Tipos de Diagramas Estructurales:

1. **Diagrama de Clases:** Estructura de clases y sus relaciones
2. **Diagrama de Objetos:** Instancias específicas en un momento dado
3. **Diagrama de Componentes:** Componentes físicos del sistema
4. **Diagrama de Despliegue:** Arquitectura física y hardware
5. **Diagrama de Paquetes:** Organización lógica en paquetes
6. **Diagrama de Estructura Compuesta:** Estructura interna de clasificadores

Notas del Presentador:
Los diagramas estructurales son como "fotografías" del sistema en un momento dado. Muestran qué existe, no qué sucede.

Diagramas de Comportamiento

Representan el **comportamiento dinámico** del sistema: qué sucede y cuándo.

Tipos de Diagramas de Comportamiento:

Diagramas de Interacción (4 tipos):

- **Diagrama de Secuencia:** Interacciones ordenadas por tiempo
- **Diagrama de Comunicación:** Interacciones entre objetos
- **Diagrama de Tiempo:** Cambios de estado en el tiempo
- **Diagrama de Vista de Interacción:** Vista general de interacciones

Otros Diagramas de Comportamiento (3 tipos):

- **Diagrama de Casos de Uso:** Funcionalidades del sistema
- **Diagrama de Actividades:** Flujos de trabajo y procesos
- **Diagrama de Estados:** Estados y transiciones de objetos

Estructural vs. Comportamiento

Diagramas Estructurales

- Muestran **QUÉ** existe en el sistema
- Representan la **estructura estática**
- Son como un **mapa arquitectónico**
- Ejemplo: Diagrama de Clases muestra las clases y sus atributos

Diagramas de Comportamiento

- Muestran **CÓMO** funciona el sistema
- Representan el **comportamiento dinámico**
- Son como un **video del sistema**
- Ejemplo: Diagrama de Secuencia muestra las interacciones paso a paso

Analogía: Estructural = Plano de una casa | Comportamiento = Video de personas viviendo en la casa

Importancia del Modelado en Software

Beneficios del Modelado con UML:

- **Comunicación:** Lenguaje común entre desarrolladores, analistas y stakeholders
- **Visualización:** Facilita entender sistemas complejos
- **Documentación:** Crea documentación viva y actualizada
- **Diseño:** Permite diseñar antes de implementar
- **Análisis:** Identifica problemas temprano en el desarrollo
- **Mantenimiento:** Facilita el mantenimiento y evolución del software

> **Regla de oro:** "Un diagrama vale más que mil palabras" - especialmente cuando se trata de sistemas complejos.

UML en la Vida Real

Casos de Uso Comunes:

- **Análisis de Requisitos:** Diagramas de Casos de Uso para entender funcionalidades
- **Diseño de Arquitectura:** Diagramas de Componentes y Despliegue
- **Diseño de Base de Datos:** Diagramas de Clases para modelar entidades
- **Documentación de APIs:** Diagramas de Secuencia para documentar flujos
- **Planificación de Proyectos:** Diagramas de Actividades para procesos
- **Comunicación con Clientes:** Diagramas simples para explicar el sistema

> **Ejemplo Real:** Empresas como IBM, Microsoft, y Oracle usan UML extensivamente en sus procesos de desarrollo.

Herramientas para Modelado UML

Herramientas Comerciales:

- **Enterprise Architect:** Herramienta completa y profesional
- **IBM Rational Rose:** Clásica herramienta de IBM
- **Visual Paradigm:** Popular en la industria
- **MagicDraw:** Herramienta empresarial

Herramientas Open Source:

- **StarUML:** Gratuita y potente
- **PlantUML:** Basada en texto, muy práctica
- **ArgoUML:** Herramienta Java open source
- **Draw.io / diagrams.net:** Online y gratuita

> **Recomendación para estudiantes:** Empezar con Draw.io (gratis) o StarUML, luego avanzar a herramientas más profesionales.

Errores Comunes al Iniciar con UML

1. Modelar TODO

❌ Error: Intentar modelar cada detalle del sistema

✅ Correcto: Modelar solo lo que es relevante y útil

2. Diagramas demasiado complejos

❌ Error: Un diagrama con 50+ elementos

✅ Correcto: Dividir en múltiples diagramas más simples

3. No actualizar diagramas

❌ Error: Crear diagramas y olvidarlos

✅ Correcto: Mantener diagramas actualizados con el código

Buenas Prácticas en UML

- **Simplicidad:** Mantener diagramas simples y claros
- **Consistencia:** Usar notación estándar consistentemente
- **Propósito claro:** Cada diagrama debe tener un objetivo específico
- **Nombres descriptivos:** Usar nombres claros y significativos
- **Documentación:** Agregar notas cuando sea necesario
- **Iteración:** Refinar diagramas a medida que se aprende más

> **Principio KISS:** "Keep It Simple, Stupid" - Los mejores diagramas son los más simples que comunican efectivamente la idea.

Resumen de la Unidad

Puntos Clave:

1. **UML** surgió de la necesidad de estandarizar el modelado orientado a objetos
2. Fue creado por **Booch, Rumbaugh y Jacobson** en los años 90
3. Es mantenido por **OMG** como estándar internacional
4. Existen **14 tipos de diagramas** organizados en estructurales y de comportamiento
5. Los diagramas **estructurales** muestran QUÉ existe
6. Los diagramas de **comportamiento** muestran CÓMO funciona
7. UML es esencial para **comunicación, diseño y documentación** en desarrollo de software

> **Próxima Unidad:** Profundizaremos en los Diagramas Estructurales, empezando con Diagramas de Clases.

Práctica Guiada

Ejercicio 1: Identificar Tipos de Diagramas

Para cada escenario, identifica qué tipo de diagrama UML sería más apropiado:

1. Mostrar las clases de un sistema de biblioteca y sus relaciones
2. Documentar el proceso de registro de un usuario
3. Mostrar cómo interactúan objetos durante una transacción
4. Representar los estados de una orden de compra

Ejercicio 2: Investigación

Investiga y documenta:

- ¿Qué empresas importantes usan UML en su desarrollo?
- ¿Cuál es la diferencia entre UML 1.x y UML 2.x?
- ¿Existen alternativas a UML? ¿Cuáles?

Mini-Quiz

Pregunta 1:

¿Qué significa la sigla UML?

Respuesta: Unified Modeling Language (Lenguaje Unificado de Modelado)

Pregunta 2:

¿Cuántos tipos de diagramas define UML 2.5?

Respuesta: 14 tipos de diagramas (6 estructurales y 8 de comportamiento)

Pregunta 3:

¿Cuál es la diferencia principal entre diagramas estructurales y de comportamiento?

Respuesta: Los estructurales muestran QUÉ existe (estructura estática), mientras que los de comportamiento muestran CÓMO funciona (comportamiento dinámico).

Tarea para Casa

Actividad:

Crear un documento que incluya:

1. Una línea de tiempo detallada de la evolución de UML
2. Una tabla comparativa de los 14 tipos de diagramas UML
3. Un ejemplo de uso de UML en un proyecto real (investigación)
4. Una lista de al menos 5 herramientas UML con sus características

Fecha de Entrega:

Una semana después de esta clase

Criterios de Evaluación:

- Completitud de la información (40%)
- Claridad y organización (30%)
- Calidad de los ejemplos (30%)

Recursos Adicionales

📚 Material Complementario

Para ampliar tu conocimiento sobre esta unidad, visita nuestra página de recursos con:

- Enlaces a documentación oficial
- Videos educativos
- Herramientas recomendadas
- Libros y tutoriales
- Ejercicios prácticos

Ver Recursos Adicionales →

Recursos Adicionales - Unidad 01

Fundamentos de UML - Enlaces, videos y material complementario

📚 Documentación Oficial

[Especificación Oficial UML 2.5 - OMG Documentación]

Documentación completa y oficial del estándar UML mantenido por Object Management Group (OMG).

[UML Diagrams - Guía Completa Referencia]

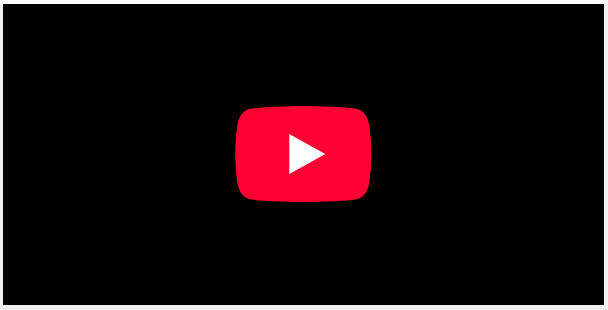
Guía exhaustiva de todos los tipos de diagramas UML con ejemplos y explicaciones detalladas.

[Sitio Oficial UML Oficial]

Portal oficial de UML con recursos, noticias y actualizaciones sobre el estándar.

🎥 Videos Educativos

Introducción a UML



Video introductorio sobre UML y su importancia en el desarrollo de software.

[Historia de UML - Los Tres Amigos Video]

Documental sobre la creación de UML por Booch, Rumbaugh y Jacobson.

[¿Qué es UML? Explicación Completa Video]

Explicación detallada de qué es UML y por qué es importante en el desarrollo de software.

[Tipos de Diagramas UML - Resumen Video]

Resumen visual de los 14 tipos de diagramas UML con ejemplos prácticos.

🛠️ Herramientas UML

[Draw.io (diagrams.net) Gratis]

Herramienta online gratuita para crear diagramas UML. No requiere instalación.

[StarUML Gratis]

Herramienta profesional de modelado UML, gratuita y de código abierto.

PlantUML Gratis

Herramienta basada en texto para crear diagramas UML mediante código.

Visual Paradigm Comercial

Herramienta profesional completa para modelado UML y desarrollo de software.

Lucidchart Freemium

Herramienta online para crear diagramas UML con colaboración en tiempo real.

📖 Libros Recomendados

UML y Patrones - Craig Larman

Libro clásico sobre UML y patrones de diseño orientado a objetos. Mencionado en el programa.

UML: Guía del Usuario - Booch, Rumbaugh, Jacobson

Escrito por los creadores de UML. Referencia esencial para entender el lenguaje.

Aprendiendo UML 2.0 - Russ Miles

Guía práctica y accesible para aprender UML 2.0 con ejemplos claros.

🌐 Artículos y Tutoriales

Tutorial UML - TutorialsPoint

Tutorial completo y gratuito sobre UML con ejemplos prácticos.

Tutorial UML - Guru99

Guía paso a paso para aprender UML desde cero.

Guía UML de IBM

Documentación técnica de IBM sobre UML y modelado de software.

💡 Recursos Interactivos

Ejemplos de Diagramas UML

Colección extensa de ejemplos de diagramas UML para diferentes casos de uso.

Plantillas UML - Creately

Plantillas listas para usar de diferentes tipos de diagramas UML.

🎓 Cursos Online

> [Cursos UML en Udemy]
>
> Variedad de cursos online sobre UML, desde principiantes hasta avanzados.

> [Cursos UML en Coursera]
>
> Cursos universitarios sobre UML y modelado de software.

💡 Consejo de Estudio

Te recomendamos empezar con los videos introductorios, luego practicar con Draw.io o StarUML, y finalmente profundizar con la documentación oficial y los libros recomendados. La práctica constante es clave para dominar UML.

Modelado de Software usando UML

Unidad 02: Diagramas Estructurales

Competencias y Resultados de Aprendizaje

Competencia Específica:

CE15: Gestionar estrategias de mantenimiento de software, aplicando los planes de mejora de éste de manera adecuada.

Resultados de Aprendizaje:

- Comprender los diagramas de clases y su notación
- Crear diagramas de objetos para instancias específicas
- Organizar sistemas usando diagramas de paquetes
- Aplicar diagramas estructurales en proyectos reales

Agenda de la Unidad

1. Introducción a Diagramas Estructurales
2. Diagramas de Clases
3. Diagramas de Objetos
4. Diagramas de Paquetes
5. Relaciones entre elementos
6. Ejemplos prácticos
7. Práctica guiada

Introducción a Diagramas Estructurales

Los diagramas estructurales representan la **estructura estática** del sistema: qué elementos existen y cómo se relacionan.

Características:

- Muestran la organización del sistema
- Representan elementos y sus relaciones
- No muestran comportamiento dinámico
- Son como "fotografías" del sistema

> **Tipos principales:** Diagramas de Clases, Objetos y Paquetes son los más utilizados en el desarrollo de software.

Diagramas de Clases

Definición:

Muestran las clases del sistema, sus atributos, operaciones y las relaciones entre ellas.

Componentes principales:

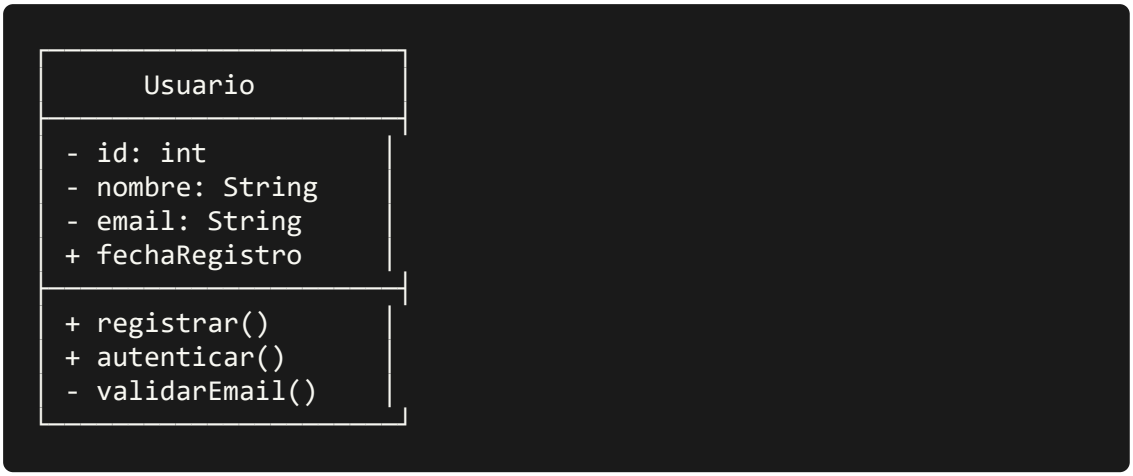
- **Clase:** Representada como un rectángulo con tres secciones (nombre, atributos, operaciones)
- **Atributos:** Propiedades o características de la clase
- **Operaciones:** Métodos o comportamientos de la clase
- **Relaciones:** Asociación, herencia, dependencia, etc.

Notación de Clases

Visibilidad:

- `+` Público (public)
- `-` Privado (private)
- `#` Protegido (protected)
- `~` Paquete (package)

Ejemplo:



Diagramas de Objetos

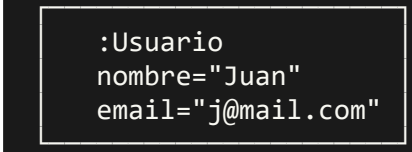
Definición:

Muestran instancias específicas de clases en un momento dado del sistema.

Diferencia con Diagramas de Clases:

- **Clases:** Plantillas o moldes (qué puede existir)
- **Objetos:** Instancias específicas (qué existe ahora)

Notación:



> **Uso:** Útiles para mostrar ejemplos concretos y validar el diseño de clases.

Diagramas de Paquetes

Definición:

Organizan y agrupan elementos relacionados del sistema en paquetes lógicos.

Propósito:

- Organizar grandes sistemas en módulos
- Mostrar dependencias entre módulos
- Facilitar la comprensión de la arquitectura
- Gestionar la complejidad

Notación:

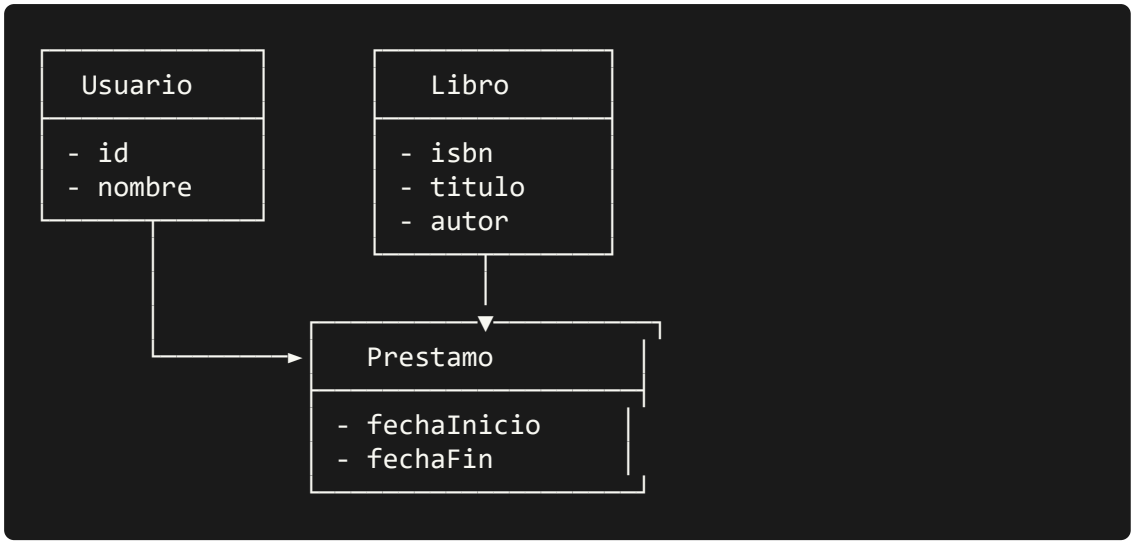
Relaciones en Diagramas Estructurales

Tipos de Relaciones:

- **Asociación:** Relación entre clases (línea simple)
- **Herencia/Generalización:** Relación "es-un" (triángulo)
- **Dependencia:** Una clase usa a otra (línea punteada con flecha)
- **Agregación:** Relación "tiene-un" (diamante vacío)
- **Composición:** Relación "parte-de" (diamante lleno)

> **Multiplicidad:** Indica cuántas instancias participan en la relación (1, *, 0..1, 1..*, etc.)

Ejemplo: Sistema de Biblioteca



Relaciones: Usuario y Libro están asociados a través de Prestamo (relación de asociación con clase intermedia).

Buenas Prácticas

- **Simplicidad:** No más de 7-9 clases por diagrama
- **Nombres claros:** Usar nombres descriptivos y consistentes
- **Agrupación:** Usar paquetes para organizar grandes sistemas
- **Relaciones:** Mostrar solo las relaciones relevantes
- **Documentación:** Agregar notas cuando sea necesario

> **Principio:** Un diagrama debe comunicar una idea clara. Si es demasiado complejo, divídelo.

Práctica Guiada

Ejercicio 1: Diagrama de Clases

Crear un diagrama de clases para un sistema de gestión de estudiantes que incluya:

- Clase Estudiante (id, nombre, email)
- Clase Curso (codigo, nombre, creditos)
- Clase Inscripcion (fecha, calificacion)
- Relaciones apropiadas entre las clases

Ejercicio 2: Diagrama de Paquetes

Organizar el sistema anterior en paquetes lógicos (Modelo, Vista, Controlador).

Resumen

Puntos Clave:

1. Los diagramas estructurales muestran la organización estática del sistema
2. Los diagramas de clases son los más utilizados en UML
3. Los diagramas de objetos muestran instancias específicas
4. Los diagramas de paquetes organizan sistemas grandes
5. Las relaciones definen cómo se conectan los elementos

> **Próxima Unidad:** Diagramas de Comportamiento e Interacción

Recursos Adicionales

📚 Material Complementario

Para ampliar tu conocimiento sobre esta unidad, visita nuestra página de recursos con:

- Videos tutoriales sobre diagramas estructurales
- Ejemplos prácticos
- Herramientas para crear diagramas
- Ejercicios adicionales

Ver Recursos Adicionales →

Recursos Adicionales - Unidad 02

Diagramas Estructurales - Enlaces, videos y material complementario

📚 Documentación Específica

Guía Completa de Diagramas de Clases Documentación

Documentación detallada sobre diagramas de clases con ejemplos y mejores prácticas.

Diagramas de Objetos - Referencia Referencia

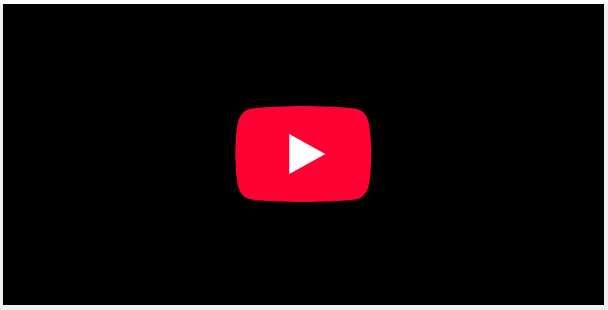
Guía completa sobre diagramas de objetos y cuándo usarlos.

Diagramas de Paquetes Referencia

Documentación sobre diagramas de paquetes y organización de sistemas.

🎥 Videos Educativos

Diagramas de Clases - Tutorial Completo



Tutorial completo sobre cómo crear y leer diagramas de clases en UML.

[Diagramas de Clases: Atributos y Operaciones Video]

Explicación detallada de cómo representar atributos y operaciones en diagramas de clases.

[Relaciones en Diagramas de Clases Video]

Video sobre los diferentes tipos de relaciones: asociación, herencia, dependencia, etc.

[Diagramas de Objetos vs Diagramas de Clases Video]

Explicación de las diferencias entre diagramas de objetos y diagramas de clases.

[Diagramas de Paquetes - Organización de Sistemas Video]

Cómo usar diagramas de paquetes para organizar sistemas grandes.

🛠️ Herramientas Específicas

[Visual Paradigm - Tutorial Diagramas de Clases Tutorial]

Tutorial interactivo sobre cómo crear diagramas de clases usando Visual Paradigm.

StarUML - Guía Diagramas de Clases Guía

Guía oficial de StarUML para trabajar con diagramas de clases.

📖 Ejemplos Prácticos

Ejemplos de Diagramas de Clases

Colección de ejemplos reales de diagramas de clases para diferentes dominios.

Ejemplos de Diagramas de Paquetes

Ejemplos prácticos de cómo organizar sistemas usando diagramas de paquetes.

Plantillas de Diagramas de Clases

Plantillas listas para usar de diagramas de clases para diferentes casos.

💡 Artículos y Tutoriales

Tutorial: Diagramas de Clases - TutorialsPoint

Tutorial completo sobre diagramas de clases con ejemplos paso a paso.

[Guía de Diagramas de Clases - Guru99]

Guía detallada sobre diagramas de clases con casos de uso prácticos.

[Diagramas de Clases - Documentación IBM]

Documentación técnica de IBM sobre diagramas de clases.

🎯 Ejercicios Prácticos

[Ejercicios de Diagramas de Clases]

Ejercicios prácticos para practicar la creación de diagramas de clases.

[Tutorial Interactivo UML 2.0]

Tutorial interactivo que incluye ejercicios sobre diagramas estructurales.

💡 Consejo de Estudio

Para dominar los diagramas estructurales, te recomendamos: 1) Estudiar los conceptos teóricos, 2) Practicar creando diagramas de sistemas que conozcas (biblioteca, tienda online, etc.), 3) Revisar ejemplos de la industria, y 4) Usar herramientas como Draw.io o StarUML para practicar. La práctica constante es esencial.

Modelado de Software usando UML

Unidad 03: Diagramas de Comportamiento e Interacción

Competencias y Resultados de Aprendizaje

Resultados de Aprendizaje:

- Comprender diagramas de casos de uso
- Crear diagramas de actividades
- Modelar estados con diagramas de estados
- Documentar interacciones con diagramas de secuencia

Agenda de la Unidad

1. Diagramas de Casos de Uso
2. Diagramas de Actividades
3. Diagramas de Estados
4. Diagramas de Secuencia
5. Comparación y uso práctico

Diagramas de Casos de Uso

Definición:

Muestran las funcionalidades del sistema desde la perspectiva del usuario.

Componentes:

- **Actor:** Usuario o sistema externo (hombrecito)
- **Caso de Uso:** Funcionalidad del sistema (óvalo)
- **Relaciones:** Asociación, inclusión, extensión

Diagramas de Actividades

Definición:

Representan flujos de trabajo, procesos de negocio o algoritmos.

Elementos principales:

- **Actividad:** Acción o tarea (rectángulo redondeado)
- **Decisión:** Punto de bifurcación (rombo)
- **Flujo:** Flechas que conectan actividades
- **Inicio/Fin:** Nodos inicial y final

> **Uso:** Ideal para modelar procesos de negocio y flujos de trabajo complejos.

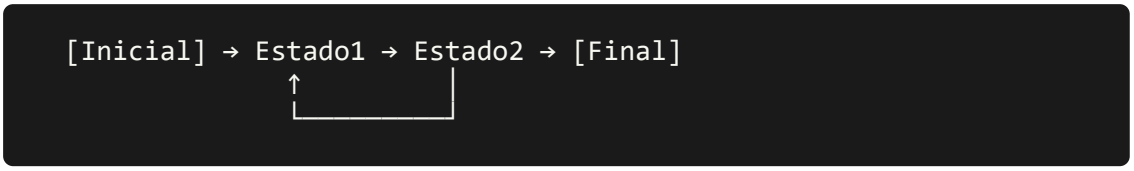
Diagramas de Estados

Definición:

Muestran los diferentes estados por los que pasa un objeto y las transiciones entre ellos.

Componentes:

- **Estado:** Condición del objeto (rectángulo redondeado)
- **Transición:** Cambio de estado (flecha)
- **Evento:** Disparador de transición
- **Estado inicial/final:** Puntos de entrada/salida



Diagramas de Secuencia

Definición:

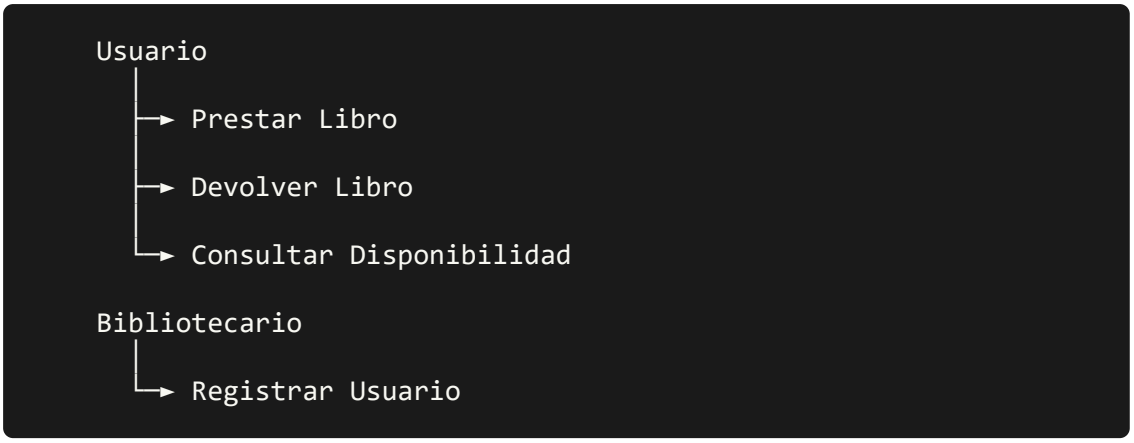
Muestran las interacciones entre objetos ordenadas por tiempo.

Elementos:

- **Línea de vida:** Representa un objeto (rectángulo vertical)
- **Mensaje:** Comunicación entre objetos (flecha horizontal)
- **Activación:** Período de ejecución (rectángulo delgado)
- **Fragmento:** Condiciones y bucles

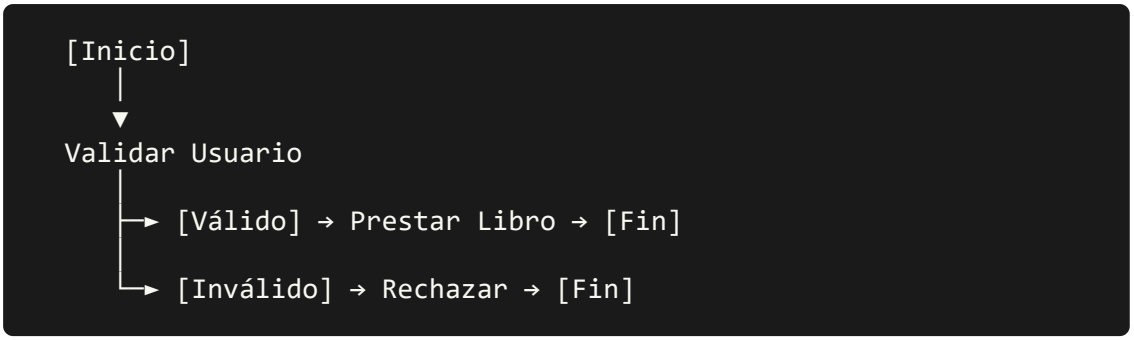
> **Uso:** Esencial para documentar cómo los objetos colaboran para realizar una funcionalidad.

Ejemplo: Caso de Uso - Sistema de Biblioteca



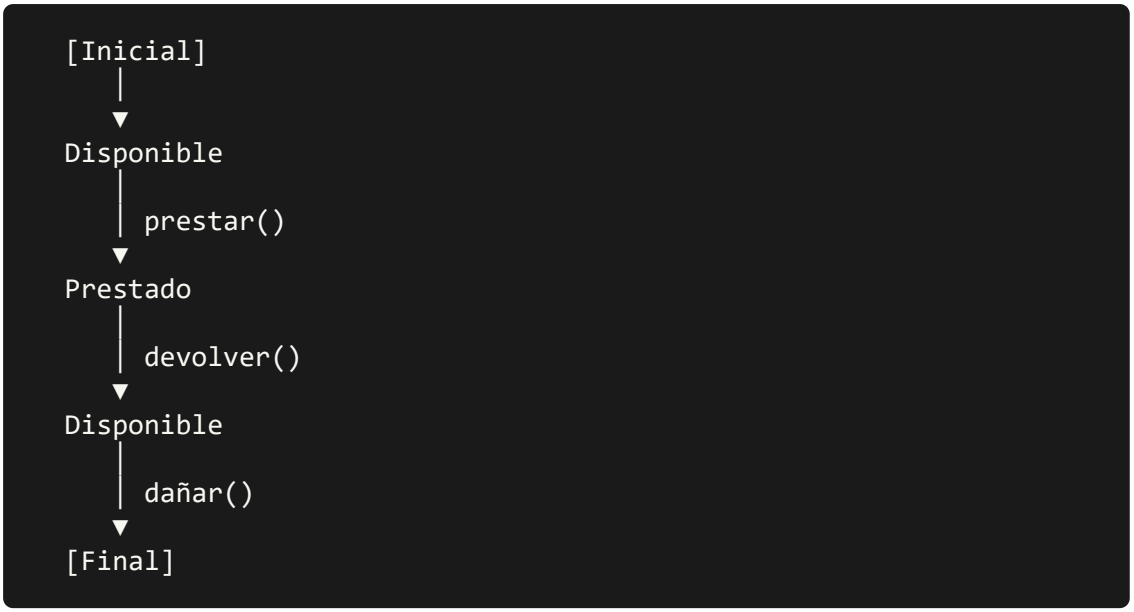
Relaciones: Los actores (Usuario, Bibliotecario) interactúan con casos de uso específicos del sistema.

Ejemplo: Diagrama de Actividades



Flujo: Muestra el proceso paso a paso con decisiones y actividades.

Ejemplo: Diagrama de Estados - Libro



Estados: Disponible, Prestado. **Transiciones:** prestar(), devolver(), dañar().

Cuándo Usar Cada Diagrama

- **Casos de Uso:** Análisis de requisitos, comunicación con stakeholders
- **Actividades:** Procesos de negocio, algoritmos complejos
- **Estados:** Objetos con comportamiento complejo basado en estados
- **Secuencia:** Diseño detallado de interacciones entre objetos

> **Regla:** Usa el diagrama más simple que comunique efectivamente la idea.

Práctica Guiada

Ejercicio: Sistema de E-commerce

Crear para un sistema de compras online:

1. Diagrama de Casos de Uso (Cliente, Administrador)
2. Diagrama de Actividades del proceso de compra
3. Diagrama de Estados de una Orden
4. Diagrama de Secuencia del proceso de pago

Resumen

Puntos Clave:

1. Los diagramas de comportamiento muestran CÓMO funciona el sistema
2. Casos de Uso: funcionalidades desde perspectiva del usuario
3. Actividades: flujos de trabajo y procesos
4. Estados: cambios de estado de objetos
5. Secuencia: interacciones ordenadas por tiempo

> **Próxima Unidad:** Agrupación de Objetos y Operaciones

Recursos Adicionales

📚 Material Complementario

Para ampliar tu conocimiento sobre esta unidad, visita nuestra página de recursos con:

- Videos sobre diagramas de comportamiento
- Ejemplos de casos de uso reales
- Tutoriales de diagramas de secuencia
- Plantillas y ejercicios

Ver Recursos Adicionales →

Recursos Adicionales - Unidad 03

Diagramas de Comportamiento e Interacción - Enlaces, videos y material complementario

📚 Documentación Específica

[Diagramas de Casos de Uso - Guía Completa Documentación]

Documentación detallada sobre diagramas de casos de uso con ejemplos y mejores prácticas.

[Diagramas de Actividades Referencia]

Guía completa sobre diagramas de actividades y flujos de trabajo.

[Diagramas de Estados Referencia]

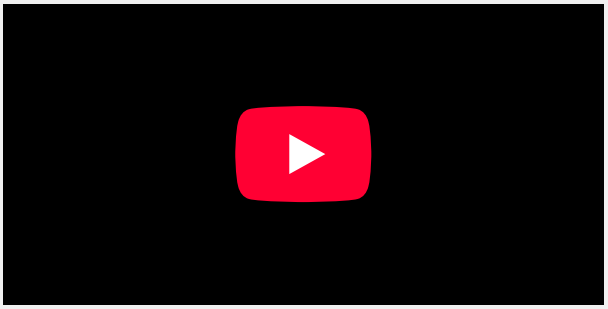
Documentación sobre diagramas de estados y máquinas de estado.

[Diagramas de Secuencia Referencia]

Guía completa sobre diagramas de secuencia e interacciones entre objetos.

🎥 Videos Educativos

Diagramas de Casos de Uso



Tutorial completo sobre cómo crear diagramas de casos de uso.

Diagramas de Actividades - Tutorial Video

Explicación detallada de diagramas de actividades con ejemplos prácticos.

Diagramas de Estados - Máquinas de Estado Video

Video sobre diagramas de estados y cómo modelar el comportamiento de objetos.

Diagramas de Secuencia - Interacciones Video

Tutorial sobre diagramas de secuencia y cómo documentar interacciones.

🛠️ Herramientas y Tutoriales

[Tutorial: Casos de Uso - TutorialsPoint Tutorial]

Tutorial paso a paso sobre diagramas de casos de uso.

[Guía: Diagramas de Actividades - Guru99 Guía]

Guía detallada sobre diagramas de actividades con casos de uso.

📖 Ejemplos Prácticos

[Ejemplos de Diagramas de Casos de Uso]

Colección de ejemplos reales de diagramas de casos de uso.

[Ejemplos de Diagramas de Actividades]

Ejemplos prácticos de diagramas de actividades para diferentes procesos.

[Ejemplos de Diagramas de Secuencia]

Ejemplos reales de diagramas de secuencia para diferentes escenarios.

💡 Artículos Especializados

[Casos de Uso - Documentación IBM]

> Documentación técnica de IBM sobre diagramas de casos de uso.

> [Diagramas de Actividades - Documentación IBM]
>
> Guía técnica sobre diagramas de actividades de IBM.

💡 Consejo de Estudio

Para dominar los diagramas de comportamiento, practica modelando procesos que conozcas: el proceso de compra en línea, el registro de usuario, el flujo de trabajo de un sistema que uses. Los diagramas de comportamiento son especialmente útiles para comunicar con stakeholders no técnicos, así que practica explicándolos a otras personas.

Modelado de Software usando UML

Unidad 04: Agrupación de Objetos y Operaciones

Competencias y Resultados de Aprendizaje

Resultados de Aprendizaje:

- Comprender los clasificadores en UML
- Definir atributos y operaciones correctamente
- Aplicar visibilidad y multiplicidad
- Utilizar abstracción en el modelado

Agenda de la Unidad

1. Clasificadores
2. Atributos
3. Operaciones
4. Visibilidad
5. Multiplicidad
6. Abstracción

Clasificadores

Definición:

Un clasificador es un mecanismo que describe características estructurales y de comportamiento.

Tipos de Clasificadores:

- **Clase:** Describe un conjunto de objetos
- **Interfaz:** Define un contrato de comportamiento
- **Tipo de Datos:** Especifica un dominio de valores
- **Componente:** Parte reutilizable del sistema

> **Propósito:** Los clasificadores agrupan elementos relacionados y definen su estructura común.

Atributos

Definición:

Un atributo es una propiedad o característica de un clasificador que describe su estado.

Sintaxis:

```
[visibilidad] nombre [: tipo] [multiplicidad] [= valorInicial]
```

Ejemplos:

- `- id: int` (privado, entero)
- `+ nombre: String` (público, cadena)
- `# saldo: double = 0.0` (protegido, decimal con valor inicial)
- `- telefonos: String[0..*]` (privado, lista de cadenas)

Operaciones

Definición:

Una operación es un comportamiento que puede ser invocado en instancias del clasificador.

Sintaxis:

```
[visibilidad] nombre [(parámetros)] [: tipoRetorno]
```

Ejemplos:

- `+ calcularTotal(): double` (público, retorna decimal)
- `- validarDatos(): boolean` (privado, retorna booleano)
- `+ agregarProducto(codigo: String, cantidad: int): void` (con parámetros)

> **Nota:** Las operaciones definen QUÉ puede hacer un objeto, no CÓMO lo hace.

Visibilidad

Niveles de Visibilidad:

- `+` **Público (Public):** Accesible desde cualquier lugar
- `-` **Privado (Private):** Solo accesible dentro de la clase
- `#` **Protegido (Protected):** Accesible en la clase y subclases
- `~` **Paquete (Package):** Accesible dentro del mismo paquete

Buenas Prácticas:

- Atributos generalmente privados
- Operaciones públicas para la interfaz
- Protegido para herencia

Multiplicidad

Definición:

Especifica cuántas instancias de un elemento pueden participar en una relación.

Notaciones Comunes:

- `1` Exactamente uno
- `0..1` Cero o uno (opcional)
- `1..*` Uno o más
- `*` Cero o más (muchos)
- `0..*` Cero o más (equivalente a *)
- `2..5` Entre dos y cinco

> **Ejemplo:** Un Usuario puede tener 0 o más Prestamos: `Usuario 1 ---- 0..* Prestamo`

Abstracción

Definición:

La abstracción es el proceso de identificar características esenciales y ocultar detalles irrelevantes.

Tipos de Abstracción:

- **Abstracción de Datos:** Ocultar detalles de implementación
- **Abstracción de Control:** Ocultar detalles de flujo de control
- **Abstracción de Proceso:** Ocultar detalles algorítmicos

En UML:

- Clases abstractas (nombre en cursiva)
- Interfaces (contratos sin implementación)
- Operaciones abstractas

Clases Abstractas

Definición:

Una clase que no puede ser instanciada directamente, solo a través de sus subclases.

Notación:

> **Uso:** Útil para definir comportamientos comunes que serán implementados por subclases específicas.

Interfaces

Definición:

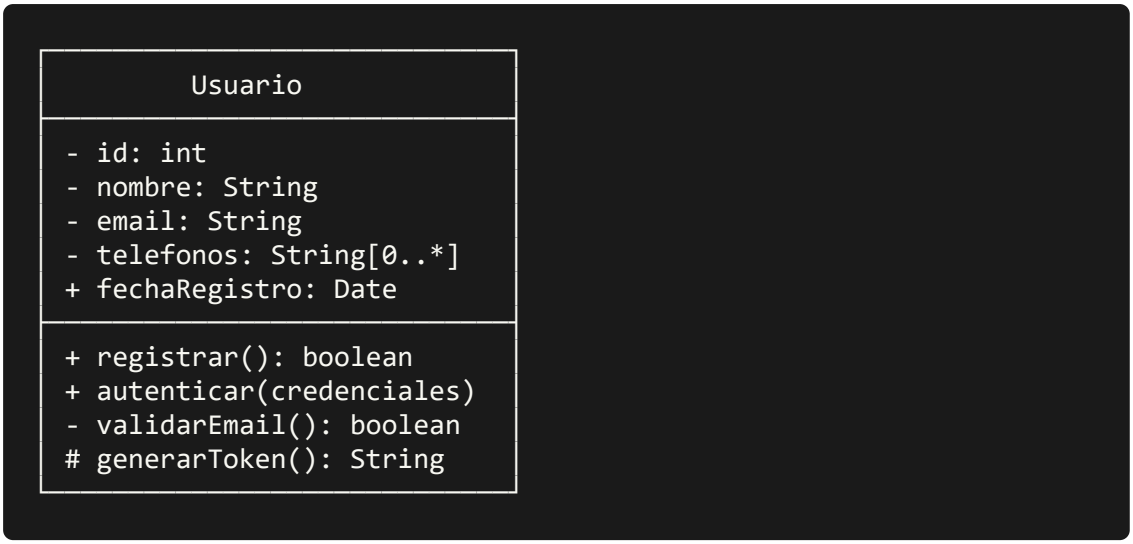
Define un contrato de comportamiento que las clases deben implementar.

Notación:

Características:

- Solo define operaciones (sin atributos)
- No tiene implementación
- Una clase puede implementar múltiples interfaces

Ejemplo Completo: Clase Usuario



Análisis: Atributos privados para encapsulación, operaciones públicas para interfaz, protegida para herencia.

Buenas Prácticas

- **Nombres descriptivos:** Usar nombres claros y significativos
- **Encapsulación:** Atributos privados, acceso a través de operaciones
- **Cohesión:** Agrupar elementos relacionados
- **Abstracción apropiada:** Ocultar detalles innecesarios
- **Documentación:** Agregar comentarios cuando sea necesario

> **Principio:** "Todo lo que puede ser privado, debe ser privado" - Minimizar la superficie pública de las clases.

Práctica Guiada

Ejercicio: Modelar una Clase Producto

Crear una clase Producto con:

- Atributos: id (privado), nombre (público), precio (privado), stock (privado)
- Operaciones: obtenerPrecio() (público), actualizarStock() (público), validarDisponibilidad() (privado)
- Aplicar visibilidad y multiplicidad apropiadas
- Considerar si necesita ser abstracta o implementar alguna interfaz

Resumen

Puntos Clave:

1. Los clasificadores agrupan elementos relacionados
2. Los atributos definen el estado de los objetos
3. Las operaciones definen el comportamiento
4. La visibilidad controla el acceso a elementos
5. La multiplicidad especifica cuántas instancias
6. La abstracción simplifica modelos complejos

> **Próxima Unidad:** Especificaciones de Clases

Recursos Adicionales

📚 Material Complementario

Para ampliar tu conocimiento sobre esta unidad, visita nuestra página de recursos con:

- Videos sobre atributos y operaciones
- Guías de visibilidad y multiplicidad
- Conceptos de abstracción
- Ejercicios prácticos

Ver Recursos Adicionales →

Recursos Adicionales - Unidad 04

Agrupación de Objetos y Operaciones - Enlaces, videos y material complementario

📚 Documentación Específica

Atributos en UML - Guía Completa Documentación

Documentación detallada sobre cómo definir y usar atributos en diagramas UML.

Operaciones en UML Referencia

Guía completa sobre operaciones, métodos y comportamientos en UML.

Visibilidad en UML Referencia

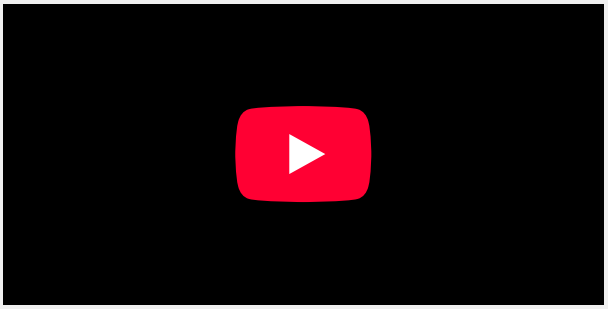
Documentación sobre niveles de visibilidad (público, privado, protegido).

Multiplicidad en UML Referencia

Guía sobre multiplicidad y cómo especificar relaciones entre elementos.

📹 Videos Educativos

Atributos y Operaciones en UML



Tutorial sobre cómo definir atributos y operaciones en clases UML.

[Visibilidad en Programación Orientada a Objetos Video]

Explicación de los niveles de visibilidad y su importancia en POO.

[Multiplicidad y Cardinalidad Video]

Video sobre multiplicidad y cómo especificar relaciones uno-a-muchos, muchos-a-muchos, etc.

[Abstracción en UML y POO Video]

Concepto de abstracción y cómo aplicarlo en el modelado UML.

💡 Artículos y Tutoriales

[Notaciones Básicas UML - TutorialsPoint]

Tutorial sobre las notaciones básicas incluyendo atributos, operaciones y visibilidad.

Clases, Atributos y Operaciones - Guru99

Guía detallada sobre cómo modelar clases con sus atributos y operaciones.

Especificación de Clases - Documentación IBM

Documentación técnica sobre especificación completa de clases en UML.

📖 Conceptos Avanzados

Clases Abstractas en UML

Documentación sobre clases abstractas y cuándo usarlas.

Interfaces en UML

Guía sobre interfaces y cómo implementarlas en UML.

Clasificadores en UML

Documentación sobre clasificadores y su uso en modelado.

🎯 Ejercicios Prácticos

Ejercicios de Atributos y Operaciones

> Ejercicios prácticos para practicar la definición de atributos y operaciones.

💡 Consejo de Estudio

Para dominar la agrupación de objetos y operaciones, practica modelando clases de sistemas reales. Piensa en las clases de un sistema de biblioteca, e-commerce, o red social, y define sus atributos (¿qué información guardan?) y operaciones (¿qué pueden hacer?). Recuerda: atributos privados para encapsulación, operaciones públicas para la interfaz.

Modelado de Software usando UML

Unidad 05: Especificaciones de Clases

Competencias y Resultados de Aprendizaje

Resultados de Aprendizaje:

- Identificar diferentes tipos de clases
- Comprender interfaces y su uso
- Definir relaciones entre clases
- Aplicar generalización e herencia
- Modelar asociaciones complejas

Agenda de la Unidad

1. Tipos de Clases
2. Interfaces
3. Tipos de Relaciones
4. Relaciones de Generalización
5. Relaciones de Asociación
6. Casos Prácticos

Tipos de Clases

Clases Concretas:

Pueden ser instanciadas directamente. Representan objetos reales del sistema.

Clases Abstractas:

No pueden ser instanciadas. Definen estructura común para subclases.

Clases de Utilidad:

Contienen solo operaciones estáticas. No tienen instancias.

Clases de Entidad:

Representan conceptos del dominio de negocio.

> **Notación:** Clases abstractas se marcan con `<<abstract>>` o nombre en cursiva.

Interfaces

Definición:

Define un contrato de comportamiento que las clases deben implementar.

Características:

- Solo define operaciones (sin atributos)
- No tiene implementación
- Una clase puede implementar múltiples interfaces
- Permite polimorfismo

Notación:

Tipos de Relaciones

Relaciones Principales:

1. **Asociación:** Relación estructural entre clases
2. **Generalización:** Relación "es-un" (herencia)
3. **Dependencia:** Una clase usa a otra
4. **Realización:** Una clase implementa una interfaz
5. **Agregación:** Relación "tiene-un" (parte-todo débil)
6. **Composición:** Relación "parte-de" (parte-todo fuerte)

> **Importante:** Cada tipo de relación tiene un propósito específico y notación diferente.

Relaciones de Generalización

Definición:

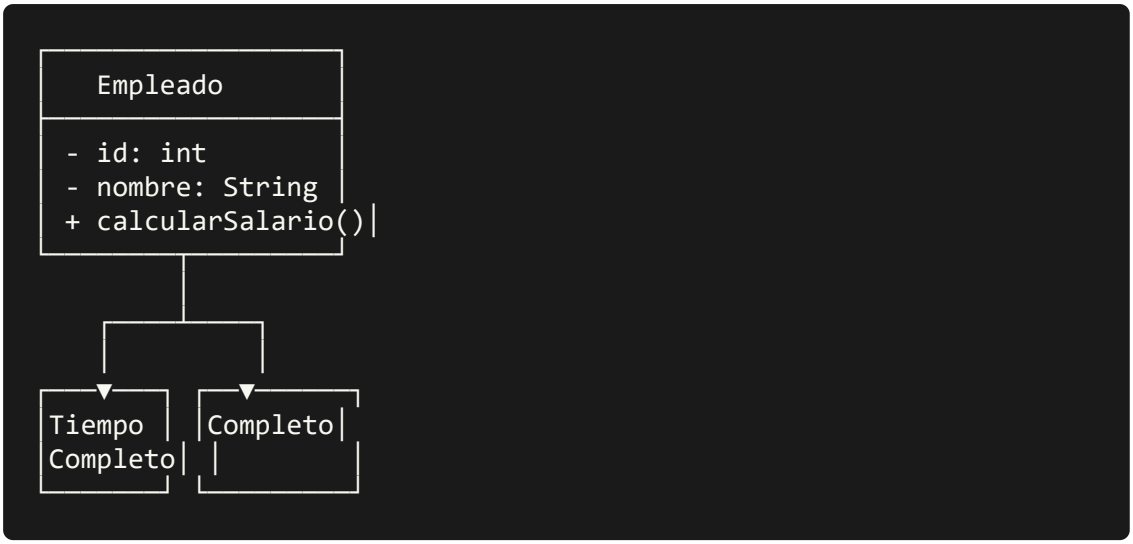
Representa herencia: una clase hija hereda características de una clase padre.

Notación:

Características:

- La clase hija hereda atributos y operaciones
- Puede sobrescribir operaciones (polimorfismo)
- Puede agregar nuevos atributos y operaciones
- Relación transitiva (herencia múltiple posible)

Ejemplo: Generalización



Análisis: Empleado es la clase padre. TiempoCompleto y EmpleadoCompleto heredan sus características y pueden tener implementaciones específicas de calcularSalario().

Relaciones de Asociación

Definición:

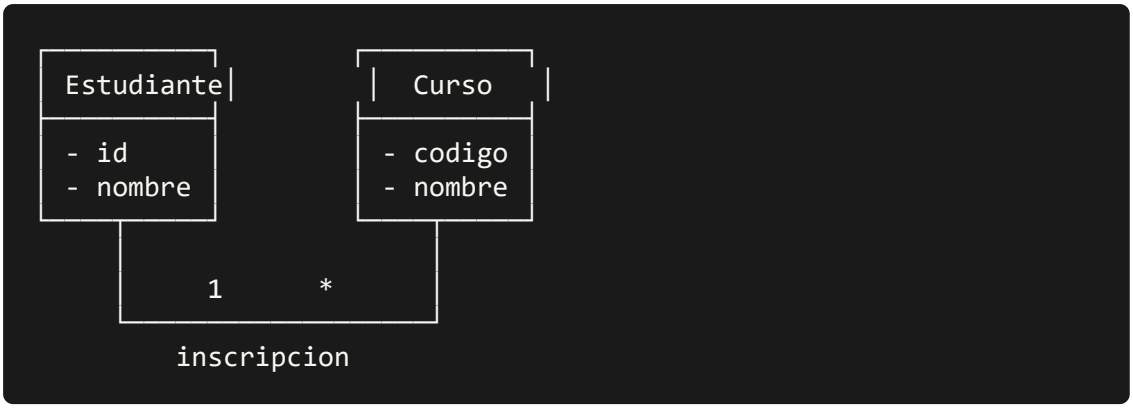
Representa una conexión estructural entre clases que permite que objetos de una clase se relacionen con objetos de otra.

Tipos:

- **Asociación Simple:** Relación básica entre clases
- **Asociación con Clase Asociativa:** Relación con atributos propios
- **Asociación Reflexiva:** Una clase se relaciona consigo misma
- **Asociación Calificada:** Con calificador para navegación

> **Notación:** Línea sólida entre clases, con multiplicidad en cada extremo.

Ejemplo: Asociación Simple

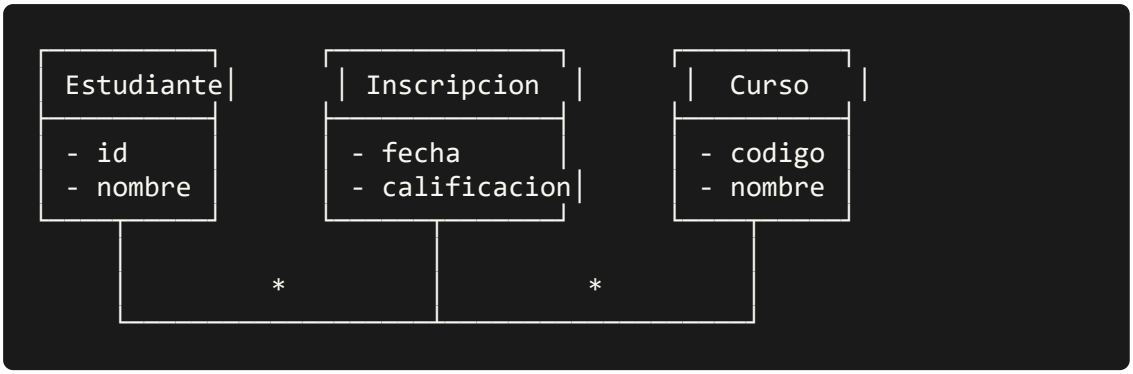


Interpretación: Un Estudiante puede estar inscrito en muchos Cursos (1..*), y un Curso puede tener muchos Estudiantes (*).

Asociación con Clase Asociativa

Cuándo usar:

Cuando la relación entre dos clases tiene atributos propios.



Ventaja: Permite almacenar información específica de la relación (fecha de inscripción, calificación).

Agregación vs. Composición

Agregación (Diamante Vacío):

- Relación "tiene-un"
- Parte puede existir sin el todo
- Relación débil
- Ejemplo: Universidad tiene Departamentos (pero un Departamento puede existir sin Universidad)

Composición (Diamante Lleno):

- Relación "parte-de"
- Parte no puede existir sin el todo
- Relación fuerte
- Ejemplo: Casa tiene Habitaciones (una Habitación no existe sin Casa)

Dependencia

Definición:

Una clase usa temporalmente a otra, pero no mantiene una relación estructural permanente.

Cuándo usar:

- Una clase usa operaciones de otra como parámetros
- Una clase crea instancias temporales de otra
- Una clase usa constantes o métodos estáticos de otra

Notación:

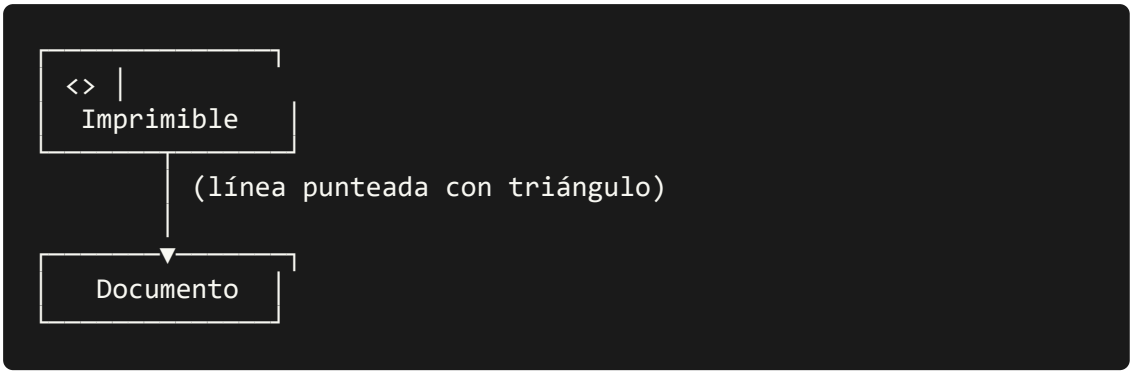
> **Diferencia con Asociación:** La dependencia es temporal, la asociación es estructural.

Realización (Implementación de Interfaces)

Definición:

Una clase implementa las operaciones definidas en una interfaz.

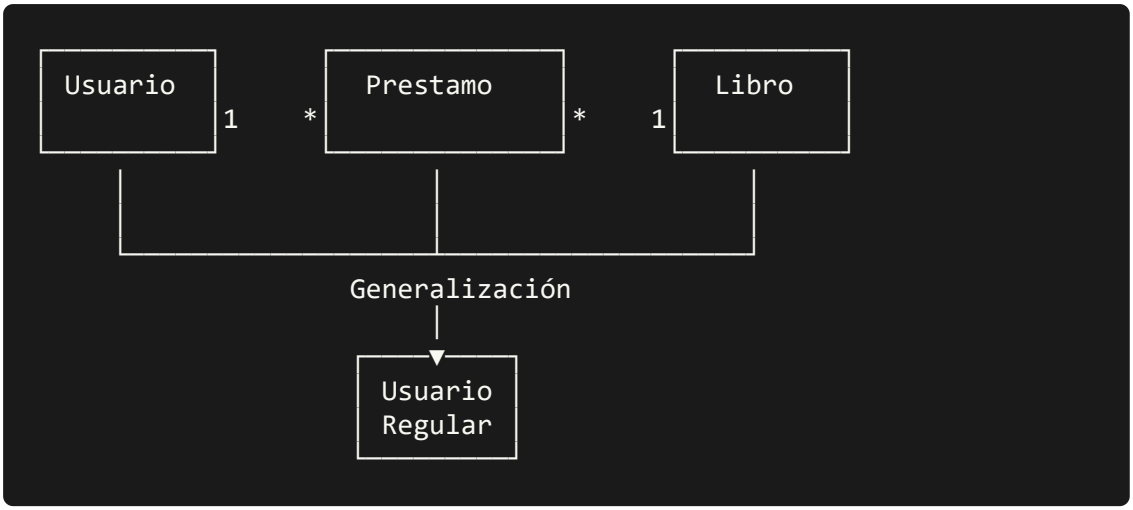
Notación:



Características:

- Una clase puede implementar múltiples interfaces
- Debe implementar todas las operaciones de la interfaz
- Permite polimorfismo

Ejemplo Completo: Sistema de Biblioteca



Análisis: Muestra asociación (Usuario-Libro a través de Prestamo) y generalización (Usuario Regular hereda de Usuario).

Buenas Prácticas en Relaciones

- **Usar herencia con cuidado:** Solo cuando hay relación "es-un" real
- **Preferir composición sobre herencia:** Más flexible
- **Interfaces para contratos:** Usar interfaces para definir comportamientos
- **Multiplicidad clara:** Especificar siempre la multiplicidad
- **Nombres descriptivos:** Nombrar roles en asociaciones

> **Principio:** "Favor composition over inheritance" - La composición es más flexible que la herencia.

Práctica Guiada

Ejercicio: Modelar Sistema de E-commerce

Crear un diagrama de clases que incluya:

1. Clase Producto (abstracta) con subclases ProductoFisico y ProductoDigital
2. Clase Cliente con relación de asociación a Orden
3. Clase Orden con composición de ItemOrden
4. Interfaz Pagable implementada por Orden
5. Relaciones apropiadas con multiplicidad

Resumen del Curso

Conceptos Clave Aprendidos:

1. Fundamentos de UML y su historia
2. Diagramas estructurales (Clases, Objetos, Paquetes)
3. Diagramas de comportamiento (Casos de Uso, Actividades, Estados, Secuencia)
4. Clasificadores, atributos y operaciones
5. Visibilidad, multiplicidad y abstracción
6. Tipos de clases e interfaces
7. Relaciones: Generalización, Asociación, Dependencia, Realización

> **¡Felicidades!** Has completado el curso de Modelado de Software usando UML. Continúa practicando y aplicando estos conceptos en proyectos reales.

Recursos Adicionales

📚 Material Complementario

Para ampliar tu conocimiento sobre esta unidad y el curso completo, visita nuestra página de recursos con:

- Videos sobre relaciones UML
- Guías de herencia vs composición
- Patrones de diseño
- Recursos avanzados

Ver Recursos Adicionales →

Recursos Adicionales - Unidad 05

Especificaciones de Clases - Enlaces, videos y material complementario

📚 Documentación Específica

Relaciones en UML - Guía Completa Documentación

Documentación detallada sobre todos los tipos de relaciones en diagramas de clases.

Generalización e Herencia Referencia

Guía completa sobre relaciones de generalización y herencia en UML.

Asociaciones en UML Referencia

Documentación sobre asociaciones simples, calificadas y reflexivas.

Interfaces y Realización Referencia

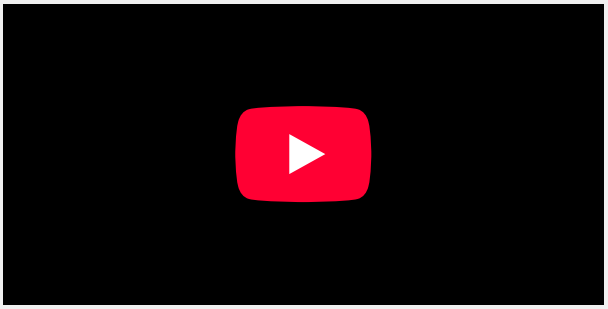
Guía sobre interfaces y la relación de realización en UML.

Agregación y Composición Referencia

Documentación sobre agregación vs composición y cuándo usar cada una.

🎥 Videos Educativos

Relaciones en UML - Tutorial Completo



Tutorial completo sobre todos los tipos de relaciones en UML.

Herencia vs Composición - ¿Cuándo usar cada una? Video

Video explicando cuándo usar herencia y cuándo composición en diseño orientado a objetos.

Interfaces en UML y POO Video

Explicación sobre interfaces, su uso y la relación de realización.

Agregación vs Composición - Diferencias Video

Video sobre las diferencias entre agregación y composición con ejemplos prácticos.

Asociaciones en UML - Tipos y Uso Video

Tutorial sobre diferentes tipos de asociaciones y cómo modelarlas correctamente.

💡 Artículos y Tutoriales

[Relaciones entre Clases - TutorialsPoint]

Tutorial completo sobre las relaciones entre clases en UML.

[Tipos de Relaciones UML - Guru99]

Guía detallada sobre todos los tipos de relaciones con ejemplos.

[Especificación de Clases - Documentación IBM]

Documentación técnica completa sobre especificación de clases y relaciones.

📖 Conceptos Avanzados

[Dependencias en UML]

Documentación sobre relaciones de dependencia y cuándo usarlas.

[Realización de Interfaces]

Guía sobre la relación de realización entre clases e interfaces.

[Patrones de Diseño - Refactoring Guru]

Recurso excelente sobre patrones de diseño que utilizan relaciones UML.

🎯 Ejercicios y Casos Prácticos

Ejercicios de Relaciones UML

Ejercicios prácticos para practicar diferentes tipos de relaciones.

Ejemplos de Diagramas con Relaciones

Ejemplos reales de diagramas de clases con diferentes tipos de relaciones.

📚 Libros Especializados

Design Patterns - Gang of Four

Libro clásico sobre patrones de diseño que utiliza extensivamente relaciones UML.

Clean Architecture - Robert C. Martin

Libro sobre arquitectura de software que profundiza en relaciones y dependencias.

💡 Consejo de Estudio

Para dominar las especificaciones de clases y relaciones, practica modelando sistemas completos. Intenta crear diagramas de clases para sistemas como: red social, sistema bancario, e-commerce, sistema de reservas. Presta especial atención a cuándo usar herencia vs composición, y recuerda: "Favor composition over inheritance". Practica identificando relaciones en código existente y modelándolas en UML.

🎉 ¡Felicitaciones!

Has completado todas las unidades del curso de Modelado de Software usando UML. Continúa practicando y aplicando estos conceptos en proyectos reales. El modelado UML es una habilidad que mejora con la práctica constante.
