UML, lenguaje de modelado gráfico

El lenguaje de modelado unificado (UML) es un estándar para la representación visual de objetos, estados y procesos dentro de un sistema. Por un lado, el lenguaje de modelado puede servir de modelo para un proyecto y garantizar así una arquitectura de información estructurada; por el otro, ayuda a los desarrolladores a presentar la descripción del sistema de una manera que sea comprensible para quienes están fuera del campo. UML se utiliza principalmente en el desarrollo de software orientado a objetos. Al ampliar el estándar en la versión 2.0, también es adecuado para visualizar procesos empresariales.

UML: conceptos básicos

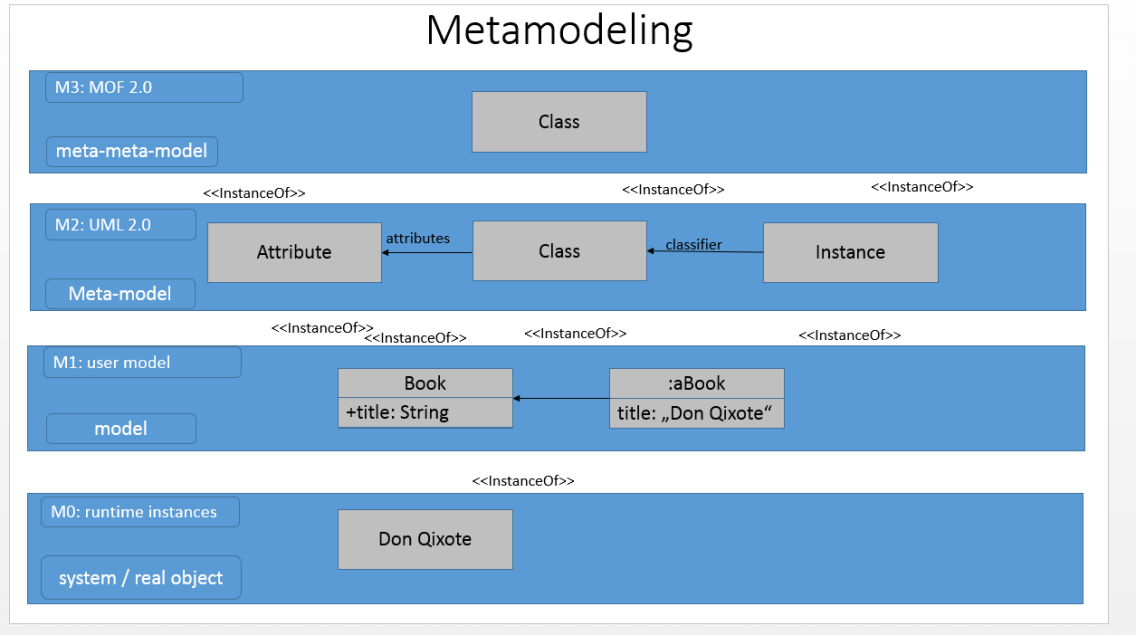
El Lenguaje Unificado de Modelado es referido por algunos como la *lingua franca* entre los lenguajes de modelado. Como se mencionó al principio, el UML visualiza los estados y las interacciones entre objetos dentro de un sistema. Su extensa popularidad se debe probablemente a la fuerte influencia que ejercen los miembros del OMG (IBM, Microsoft y HP entre otros). La semántica estructurada hace el resto. Los diagramas UML se utilizan para representar los siguientes componentes del sistema:

- Objetos individuales (elementos básicos)
- Clases (combina elementos con las mismas propiedades)
- Relaciones entre objetos (jerarquía y comportamiento/comunicación entre objetos)
- Actividad (combinación compleja de acciones/módulos de comportamiento)
- Interacciones entre objetos e interfaces

Metamodelismo

UML 2.0 define **unidades de lenguaje** que operan en diferentes niveles. Se utilizan para expresar la estructura y el comportamiento de un sistema. Algunos elementos utilizan el lenguaje de modelado para definirse. La **metamodelación** incluye todos los elementos de UML, incluyendo aquellos que describen el propio UML. Para ello utiliza cuatro niveles dispuestos jerárquicamente (M0 a M3).

El nivel meta-meta M3 especifica los metadatos del lenguaje de modelado y sus relaciones usando la **Meta Object Facility** (MOF). Define el metamodelo. También le permite transferir metadatos. El **formato XMI** definido por el OMG es una herramienta para compartir datos orientados a objetos a nivel meta-meta entre herramientas de desarrollo. El **Object Constraint Language** (OCL), un lenguaje de programación declarativo, complementa el UML y regula las condiciones de contorno de los respectivos modelos. Como lenguaje de texto, sin embargo, solo tiene un efecto de apoyo, en lugar de estar disponible para el modelado en sí mismo.



El metamodelado muestra la relación jerárquica entre los niveles de lenguaje.

El gráfico superior muestra el **metamodelado** de UML 2.0. El nivel M0 es el nivel básico. Representa **objetos reales** y concretos y registros de datos individuales, por ejemplo, un objeto o un componente. El nivel M1 comprende todos los modelos que describen y estructuran los datos del nivel M0. Estos son diagramas UML como el Diagrama de Actividad o el Diagrama de Paquete (explicado a continuación). Para definir la estructura de estos modelos, los metamodelos de nivel M2 definen las **especificaciones y la semántica** de los elementos del modelo.

Si deseas crear un diagrama UML comprensible, necesitas conocer el metamodelo UML y sus reglas. El nivel más alto, M3, es un **metamodelo del metamodelo**. El mencionado Meta Object Facility trabaja a un nivel abstracto que define los metamodelos. Este nivel se define a sí mismo, porque de lo contrario surgirían metaniveles más altos.

Unidades lingüísticas

UML (nivel M2) define las reglas de su propia semántica. Las unidades de lenguaje son términos definidos en la superestructura UML 2.0. Esto permite una representación formal que todos los participantes pueden entender. Las unidades lingüísticas, en inglés *language units*, abstraen objetos y procesos de estructura y funcionamiento similares y les dan una forma visualmente representable. Según el nivel jerárquico dentro del modelo, los elementos asumen tareas más especializadas o definen más estrechamente otros elementos.

Clase: como unidad lingüística, las clases son un aspecto central de UML. Definen lo que constituye una clase y cómo las clases interactúan entre sí. Esta *language unit* tiene cuatro niveles, que van desde elementos simples hasta relaciones más complejas:

- Núcleo (describe elementos de la infraestructura UML 2.0 como paquetes, espacios de nombres, atributos, etc.)
- AssociationClasses (define clases de asociación)
- Interfaces (define las interfaces)
- Powertypes (clase cuyas instancias son subclases dentro de esta clase)

Componente: los componentes son elementos que **separan su contenido del sistema externo**. Solo existe una conexión con el exterior a través de interfaces o puertos. Un conector de composición establece una conexión con otro componente a través de la interfaz. El conector de delegación une los elementos interiores con una interfaz en el borde exterior. Los componentes son modulares e intercambiables.

Estructura de la composición: la estructura de la composición de la unidad de lenguaje describe los elementos, que están blindados como componentes hacia adentro y hacia afuera. Solo los puertos conectan el contenido con el sistema externo. Los llamados **clasificadores encapsulados** consisten en elementos llamados **partes**. Las piezas se comunican a través de conectores.

Perfil: un perfil configura UML 2.0 para necesidades específicas. Los términos abstractos como actividad u objeto deben ser especificados para algunos proyectos con el fin de aumentar la comprensión. La semántica y las notaciones que están colocadas en lugares sueltos se pueden adaptar con un perfil.

Modelo: el modelo comprende todos los elementos necesarios para presentar una visión específica de la estructura o el comportamiento de un sistema. Esto también incluye las influencias externas, como los actores.

Acción: cuando se trata de representar el comportamiento, las acciones son de importancia central. Los valores se aceptan a través de los pines de entrada y se envían a los pines de salida. Estos son los grupos temáticos que UML define para las acciones:

- Manipular objetos
- Manipular relaciones de objetos
- Manipular características estructurales
- Acciones de llamada
- Generar valores
- Acciones sobre objetos
- Recibir eventos

Comportamiento: la unidad de lenguaje Comportamiento se refiere a la modelización de aspectos dinámicos dentro de un sistema. Consta de tres especificaciones:

- **Actividad**: las acciones interactúan a través flujos de datos y control. Esto resulta en un sistema complejo de comportamientos, las actividades.
- **Interacción:** este metamodelo describe cómo se intercambian los flujos de mensajes entre objetos, cuándo se envía un mensaje a qué objeto y qué otros elementos se ven afectados por él.
- **Estado de las máquinas:** en un diagrama de estado, este modelo de metamodelo muestra tanto estados (situaciones con propiedades inmutables) como semiestados (estados sin asignación de valor) y sus transiciones. Los objetos de un estado pueden asignarse a acciones o actividades.

Distribución: una red está formada por objetos que están conectados entre sí en mallas. Se da un caso especial de uso si estos elementos representan software ejecutable o **artefactos**. Estos artefactos se ejecutan en entornos de ejecución o dispositivos clasificados como nodos por UML 2.0. Por lo tanto, el artefacto depende del nodo. La distribución representa esta **relación de dependencia** que surge durante la instalación.

Aplicación: el caso de uso (como unidad de idioma) representa los requisitos del sistema. El actor (una persona o un sistema) es un elemento que describe quién o qué debe realizar una actividad concreta utilizando el sistema. El sistema también puede ser una

clase o un componente y, por lo tanto, se describe como un tema. El caso de uso (como elemento modelo) simplemente indica que se espera un comportamiento que sea visible para el mundo exterior, pero no suele representar qué acciones concretamente. Dentro de una descripción de comportamiento, el modelado asigna los requisitos detallados al caso de uso.

Flujos de información: esta unidad de lenguaje UML describe los elementos unidad de información y flujo de información. Estos elementos del modelo son técnicas abstractas de descripción del comportamiento que pueden estar muy orientadas al detalle, como actividades o interacciones. Esta representación simplificada permite el uso universal de estos elementos de modelado en todos los tipos de diagramas UML. Por lo tanto, a diferencia de la clase, la unidad de información nunca se describe con más detalle por atributos ni se incluye en los métodos. Por lo tanto, el flujo de información también conecta todos los elementos posibles que intercambian cualquier tipo de información entre sí.

Diagramas UML: campos de aplicación y breve introducción

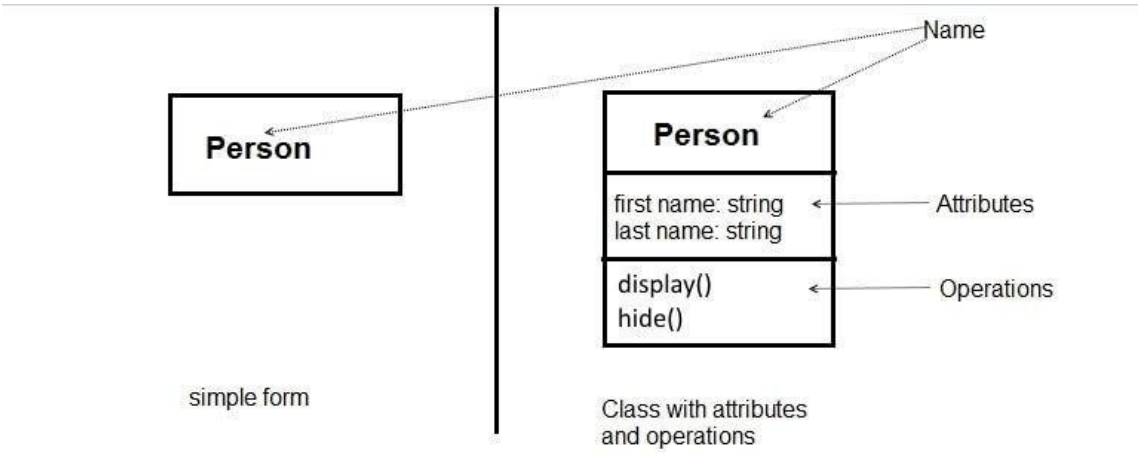
El lenguaje de modelado define 14 tipos de diagramas que se dividen en dos categorías. Las principales categorías de **estructura** y **comportamiento** representan los conceptos básicos representados por los diagramas UML. Dentro del grupo de diagramas de comportamiento, UML especifica la subcategoría de **diagramas de interacción**. Existe una cuarta especificación parcial desde UML 2.0. Determina el **diseño** de los diagramas del modelo.

Diagramas de estructura

Los diagramas de estructura representan los elementos individuales de un sistema. Por lo tanto, son especialmente adecuados para la representación de la arquitectura de software. La representación estática no representa un cambio, sino estados y dependencias en un momento determinado. Los elementos individuales u objetos están relacionados entre sí. Por ejemplo, un objeto pertenece a una clase. Otros componentes son nodos de ordenador o artefactos –un artefacto representa un resultado, por ejemplo, un archivo de script terminado.

Los diagramas UML de esta categoría representan un sistema completo o una subestructura. Esto último ayuda, por ejemplo, a clarificar la estructura en detalle. En UML 2.x, el lenguaje asigna siete tipos de diagramas a la categoría Estructura:

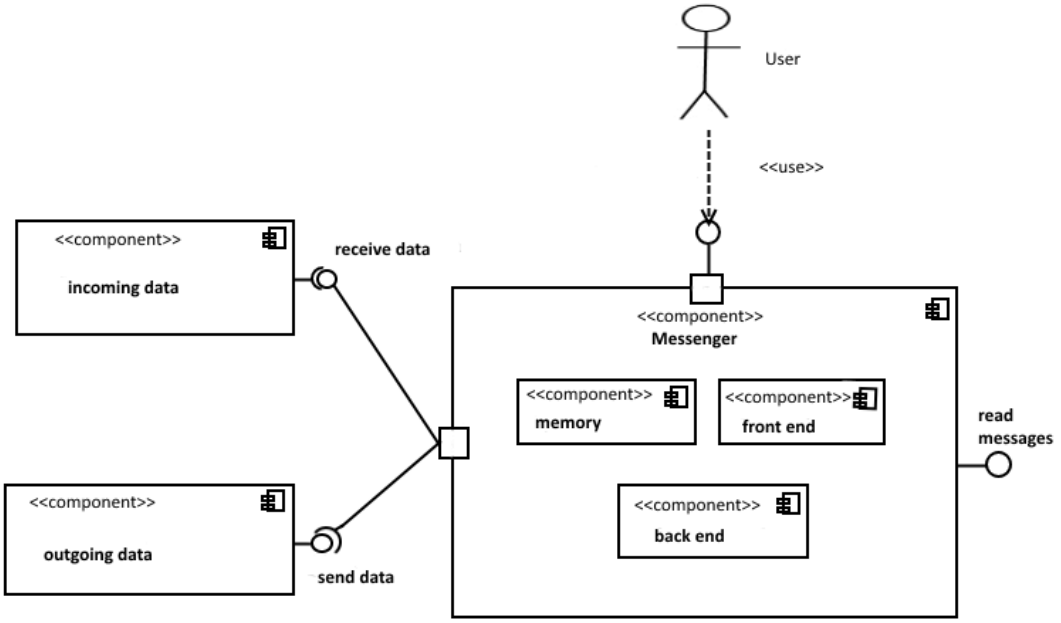
- **Diagrama de clases**: si los objetos tienen un comportamiento común o la misma estructura, pueden clasificarse (asignarse a una clase). La clase es, por tanto, un elemento simplificador (abstracción) para la representación visual. Las clases y los objetos están conectados entre sí mediante **interfaces**. El diagrama de clase muestra todos estos componentes y sus interrelaciones. Una clase representa el diagrama con un rectángulo. Contiene el nombre de la clase en negrita, como se muestra a continuación.



La clase "Persona" tiene los atributos nombre y apellido. Las operaciones determinan si la instancia se muestra (display) u oculta (hide).

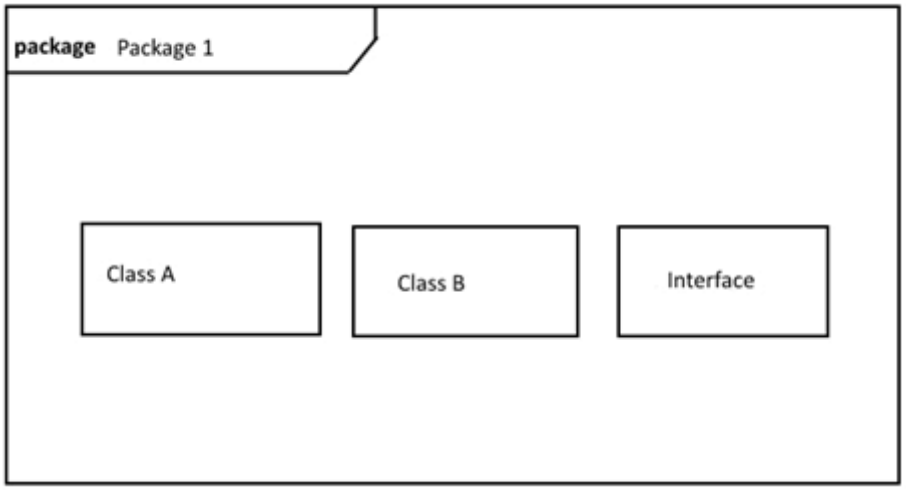
- **Diagrama de objetos**: el diagrama de objetos tiene una estructura similar a la del diagrama de clases. Cuando el nombre aparece en el diagrama de clase (ver Persona arriba), el diagrama de objeto especifica el nombre de la instancia junto con el nombre del clasificador/categoría. De acuerdo con el pliego de condiciones, se subraya lo siguiente (por ejemplo: Helga:Persona)

- **Diagrama de componentes**: un componente es un módulo que está aislado del sistema externo e interactúa con otros componentes mediante interfaces definidas. Es un subformulario de la clase. Por lo tanto, las características estructurales, como las operaciones y los atributos, definen el componente con mayor precisión. El componente contiene varios componentes. Estos pueden ser de nuevo componentes, pero también clases, subsistemas o partes. Hay dos opciones de visualización diferentes para el modelado: **Black Box** o caja negra (el contenido está oculto) y **White Box** o caja blanca (el contenido es visible).



Vista de caja blanca en un componente. Las interfaces necesarias se muestran como cajas ("Socket") y se ofrecen como círculos ("Lollipop").

- **Diagrama de estructura compositiva**: los objetos pertenecen a clases. Estos, a su vez, también pueden clasificarse. Estas metaclases se denominan **clasificadores** en UML. El diagrama de estructura de la composición representa las partes y conectores de un clasificador. Las partes son siempre parte del todo, incluso si no son necesarias para completar el clasificador. Los **conectores** son las conexiones entre las partes. Las características o servicios que requieren componentes externos al clasificador envían las piezas a través de una interfaz.
- **Diagrama de paquete**: un paquete agrupa elementos como interfaces o clases en un espacio de nombres. Los paquetes (packages) también pueden fusionarse con otros paquetes (fusión de paquetes), importarlos (importación de paquetes) o contener otros paquetes (subpaquetes). Los paquetes **estructuran el contenido del diagrama jerárquicamente** como en un diagrama de árbol. El diagrama de paquete se utiliza, por ejemplo, en el metamodelo de UML 2. En los sistemas de software representa las subáreas de forma **modular**. Según la especificación, un paquete consta de una cabecera y un área de contenido



En la cabecera el paquete de palabras clave describe el elemento, "paquete" es el nombre. En el área de contenido, el paquete incluye una interfaz y dos clases.

Nota

Un espacio de nombres es un elemento del metamodelo UML 2.
Los componentes deben tener un nombre y uno de los siguientes atributos de visibilidad: *package*, *private*, *protected* o *public*. El paquete es un caso especial del espacio de nombres.

- **Diagrama de distribución**: el diagrama de distribución modela la distribución física de los artefactos en nodos. Los **nodos** pueden ser hardware (device nodes), que puede proporcionar memoria, o software (execution environment nodes), que proporciona un entorno para ejecutar procesos. Se representan como cuboides tridimensionales. Los **artefactos** se dibujan como rectángulos conteniendo el nombre del archivo. Para distinguirlo de una clase, añade el estereotipo <<artefact>>. El diagrama es adecuado para visualizar dependencias entre nodos y artefactos, las llamadas **relaciones de distribución**.
- **Gráfica de perfil**: los diagramas de perfil se utilizan a nivel de **metamodelo**. Se utilizan para asignar un **estereotipo** a las clases o un **perfil a los paquetes**. En el metanivel tienes la posibilidad de adaptar el modelo para otra plataforma o dominio. Por ejemplo, si restringes la semántica UML dentro de un perfil, transfieres las especificaciones a las clases subordinadas.

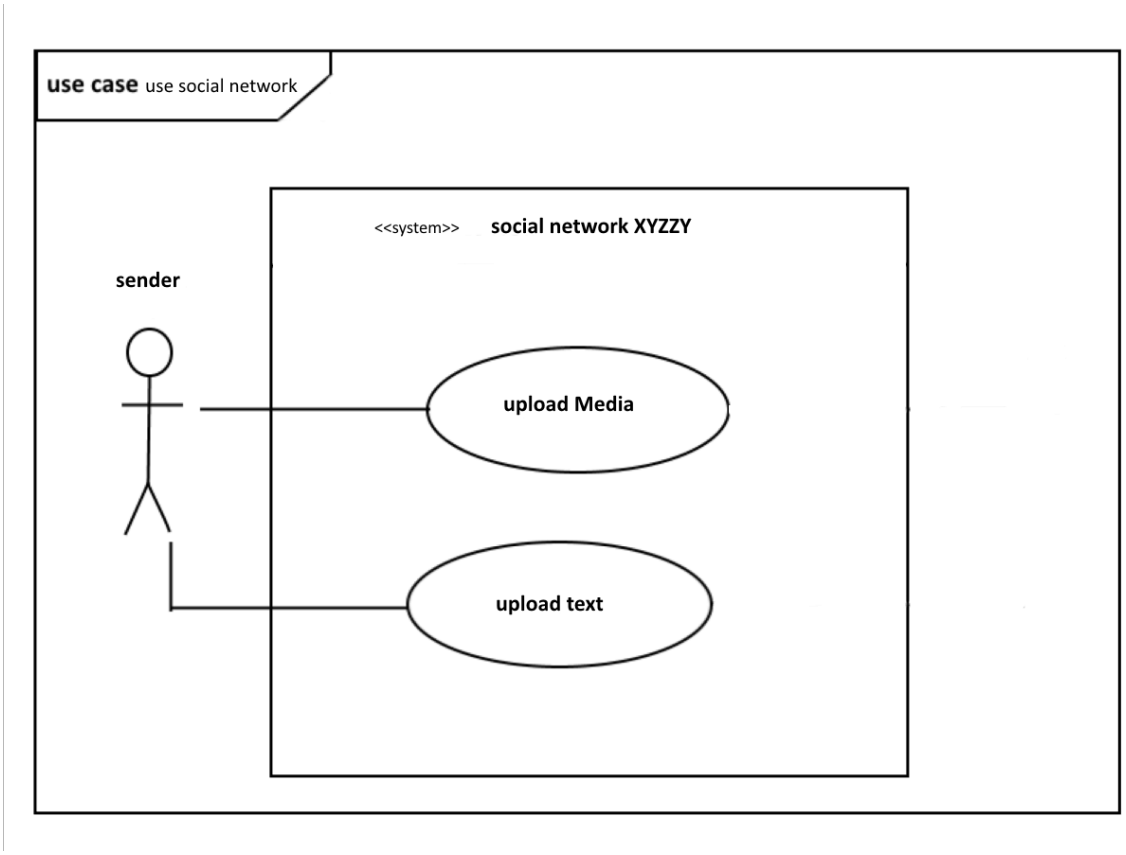
Diagramas de comportamiento

Los diagramas de comportamiento cubren las especificaciones restantes bajo UML. A diferencia de los diagramas estructurales, no son estáticos, sino que **representan procesos y situaciones dinámicas**. Los diagramas de comportamiento también incluyen los diagramas de interacción (ver abajo).

Diagrama de casos de uso: el diagrama de casos de uso muestra el comportamiento que se esperará de un sistema más adelante. Este modelo no solo es adecuado para sistemas de software, sino también, por ejemplo, para procesos esperados en las relaciones comerciales. El caso de uso involucra a un **actor** (humano o sistema) con un **objetivo**. El

diagrama normalmente tiene como nombre el objetivo. Los diferentes casos de uso dentro del sistema cumplen el objetivo del actor.

El diagrama del caso de uso representa a UML con un rectángulo con la etiqueta *use case*. El remitente es el actor (esto se representa como una figura de palo, como en el caso anterior, incluso si se trata de un sistema). El actor está conectado por una **relación de dependencia** (representada como un guion) con el **caso de uso** (elipse con etiqueta) dentro de un **sistema** (rectángulo con etiqueta <<sistema>> y nombre del sistema).



El diagrama de casos de uso en el ejemplo tiene un actor que puede esperar dos casos de uso dentro del sistema “red social XYZZY”

- **Diagrama de actividades:** las actividades consisten en una red de acciones que se relacionan entre sí mediante flujos de datos y de control. Mientras que el Diagrama de casos de uso muestra los requisitos del sistema, el Diagrama de actividades muestra **cómo funcionan estos casos de uso**. En este tipo de diagrama, por ejemplo, el **token** juega un papel importante: en los procesos paralelos, es un marcador para el cual se priorizan los procesos y, por lo tanto, se reciben los recursos (por ejemplo, la memoria de trabajo).
- **Diagrama de máquina de estados:** una máquina de estados, también llamada autómata finito, representa un conjunto finito de estados en un sistema. Si se cumple una condición **definida en el sistema** (se detona un desencadenante), se produce una **situación** correspondiente. Esto puede incluir actividades o interacciones. Bajo UML 2.0, un estado representa esta **situación**. Los estados se consideran como nodos (inglés: vértices) y se muestran como rectángulos con esquinas redondeadas. Además, el diagrama de máquina de estados modela las transiciones de un estado (nodo fuente) a otro (nodo destino). Los modelos UML

presentan las transiciones de estado como bordes. Las acciones son la última piedra angular. Se asignan especificaciones de comportamiento al estado.

El comportamiento está ligado a ciertas situaciones o eventos. El diagrama de máquina de estados UML tiene 4 especificaciones:

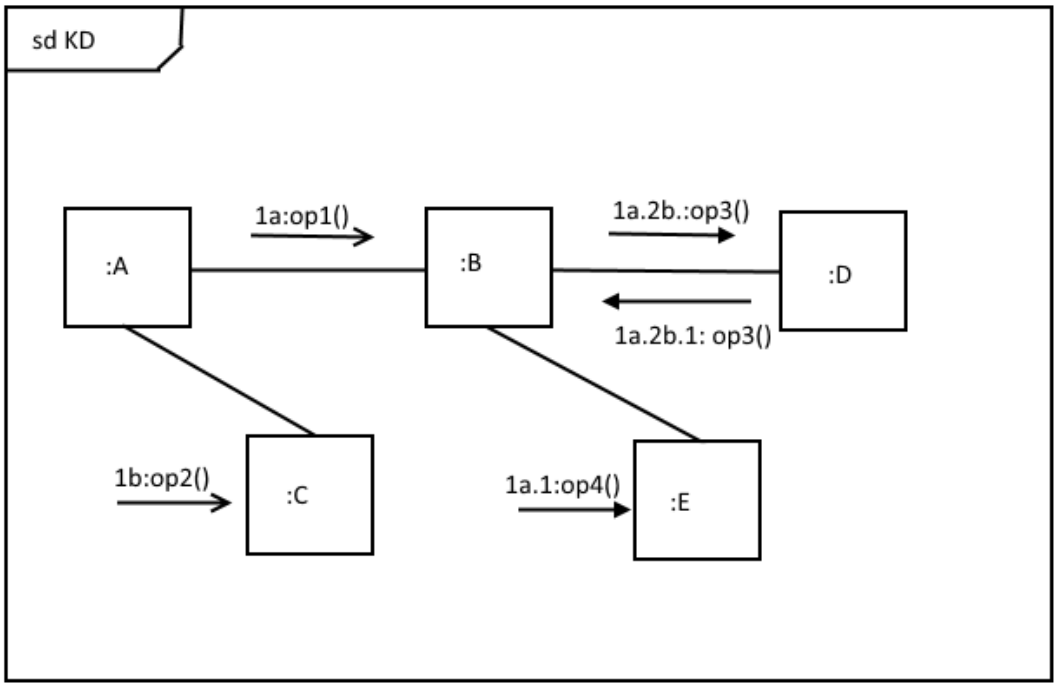
- Comportamiento de entrada (*entry behavior*)
- Comportamiento en el estado (*doActivity*)
- Comportamiento que ocurre en caso de un evento (*event behavior*)
- Comportamiento de salida (*exit behavior*)

Además, un estado puede ser interrumpido y continuado en el mismo punto. Esta propiedad se llama **estado de la historia**.

Diagramas de interacción

Los diagramas de interacción son un subtipo de los diagramas de comportamiento. Por lo tanto, también representan situaciones dinámicas. En particular, son adecuados para modelar el comportamiento en el que los elementos **intercambian información**. Los diagramas definen el papel de los objetos implicados. También nombran y priorizan los mensajes que se envían de un lado a otro entre los objetos. Los diagramas de interacción también muestran cómo estos mensajes afectan a los elementos del comportamiento. Por ejemplo, puede iniciar o detener actividades.

- **Diagramas de secuencia**: como diagrama de interacción, el diagrama de secuencia representa el intercambio de mensajes entre objetos. El tipo de diagrama UML modela estos objetos como las llamadas **líneas de vida**. En este sentido, es similar a otros diagramas de comportamiento como el diagrama de actividad. Sin embargo, a diferencia de estos, el diagrama de secuencia no se utiliza para obtener una visión general del comportamiento de un sistema, sino para presentar un **posible comportamiento entre muchos otros en detalle**. Prescribe una cronología. Una línea discontinua representa el curso del tiempo.
  UML 2.0 muestra mensajes síncronos (UML: flecha con la punta llena) y **mensajes asíncronos** (UML: flecha con la punta abierta). Los mensajes síncronos son aquellos que bloquean un canal hasta que reciben una respuesta del objeto de destino. Determinan las características de comportamiento en forma de operaciones sincrónicas. Los mensajes asincrónicos controlan el objeto fuente de llamada. Éstas incluyen tanto las operaciones asíncronas como las señales (paquetes de datos enviados entre acciones).
- **Diagrama de comunicación**: al igual que el diagrama de secuencia, el diagrama de comunicación modela una **transferencia de mensajes**. Sin embargo, este diagrama UML no utiliza líneas discontinuas para la secuencia de tiempo, sino que numera las secuencias con números y letras. Estos llamados **términos de secuencia** se encuentran encima de una flecha con la punta apuntando hacia el receptor. Los números representan el orden en que se envían los mensajes, las letras representan el nivel jerárquico (como se muestra en la figura siguiente).



Los mensajes asíncronos 1a y 1b se envían en paralelo. El objeto :B envía el mensaje con la etiqueta 1a.1 antes del mensaje 1a.2b. El objeto :D envía el mensaje 1a.2b.1 solo después de recibir el mensaje 1a.2b.

- **Diagrama de tiempos**: el diagrama de tiempos permite mostrar el comportamiento de los sistemas en detalle en función de la **secuenciación temporal**. Los sistemas en tiempo real, por ejemplo, tienen que manejar ciertos procesos dentro de un cierto período de tiempo. UML 2.0 modela el diagrama de temporización como un **diagrama bidimensional** con un eje x y un eje y para representar mejor el plano temporal. Con este subformulario del diagrama de secuencia, los **estados de los objetos** se encuentran en el eje *y* y las **secuencias de tiempo** asignadas a ellos se ejecutan a lo largo del eje y.

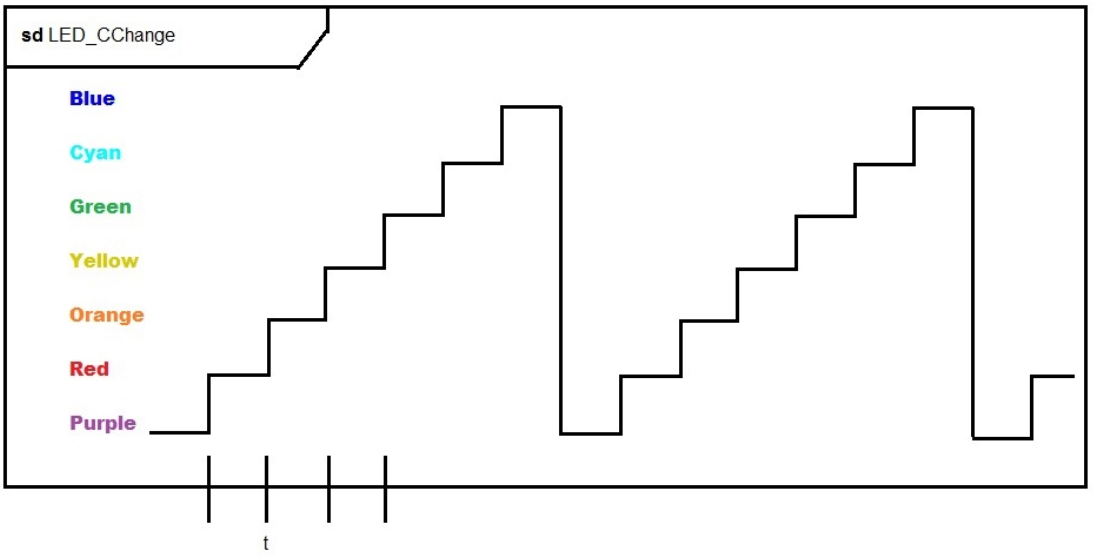


Diagrama de temporización de una lámpara LED con cambiador de color: las instancias de color se alternan y están activas durante el mismo intervalo de tiempo.

- **Diagrama de interacción**: el nuevo diagrama de interacción añadido a UML 2.0 ayuda a mostrar un sistema muy complejo primero en un esquema aproximado, si un diagrama de interacción normal se vuelve demasiado confuso. Un diagrama de secuencia, por ejemplo, es adecuado para la visualización detallada. El diagrama UML es similar al diagrama de actividad con nodos. Representa los **flujos de control entre interacciones**. La diferencia con el diagrama de actividad es que un diagrama de interacción completo puede anidarse dentro de nodos que representan actividades. Estos nidos se pueden mostrar directamente en el diagrama (en línea) o se puede hacer referencia (palabra clave: ref, del inglés *reference*) al modelo, que se muestra en detalle en otra parte.

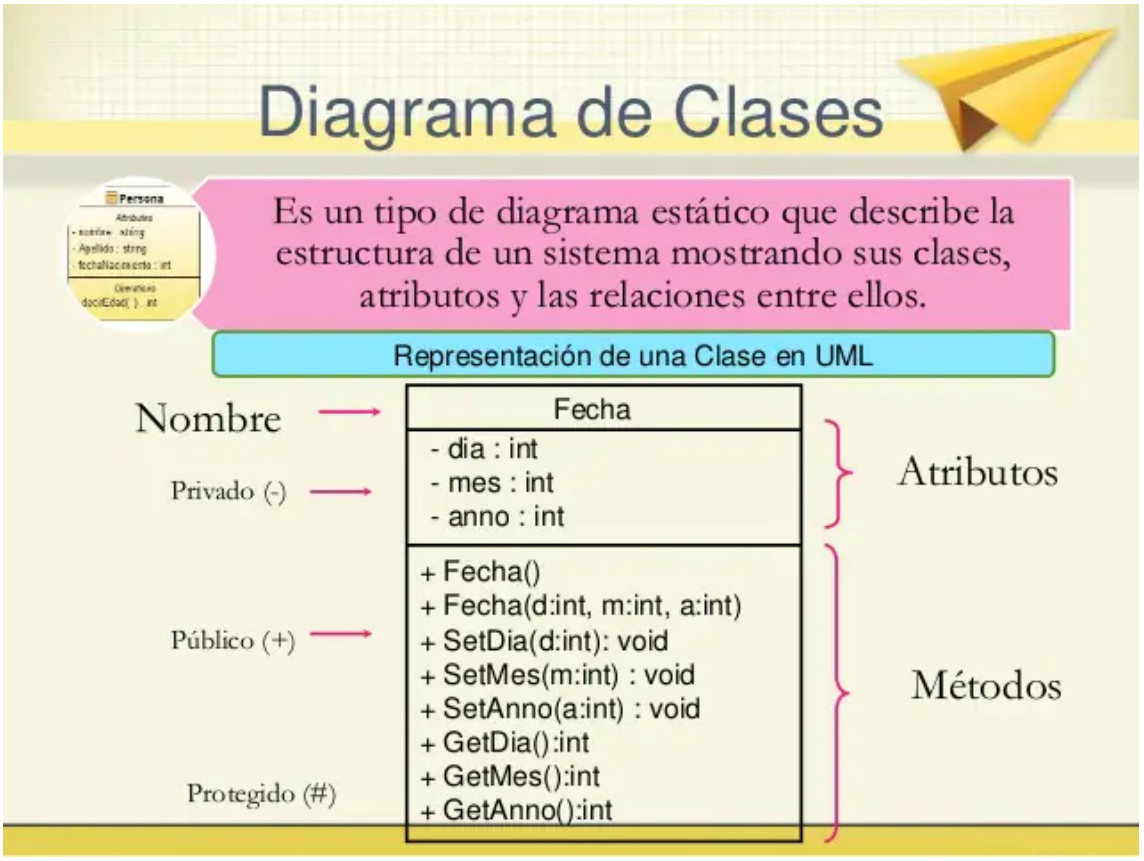
Diagramas UML: una visión general

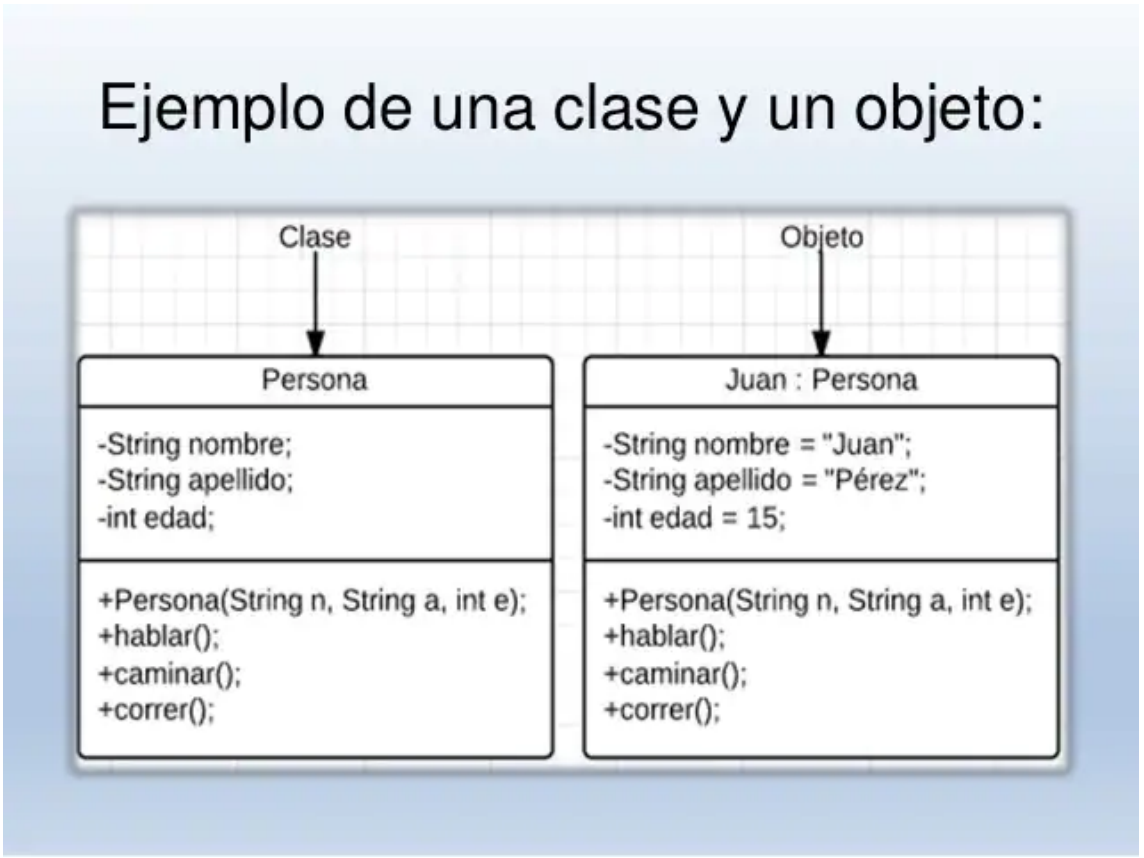
El siguiente resumen muestra las categorías superiores y las posibles aplicaciones de los tipos de diagrama individuales en forma abreviada. Si deseas representar visualmente un sistema de software orientado a modelos, un caso de uso en la economía, etc., debes seleccionar uno de los tipos de diagrama UML de antemano de acuerdo con la recomendación del Grupo de Trabajo de UML. Solo entonces vale la pena elegir una de las muchas herramientas UML, ya que estos a menudo prescriben un cierto método. A continuación, crea el diagrama UML.

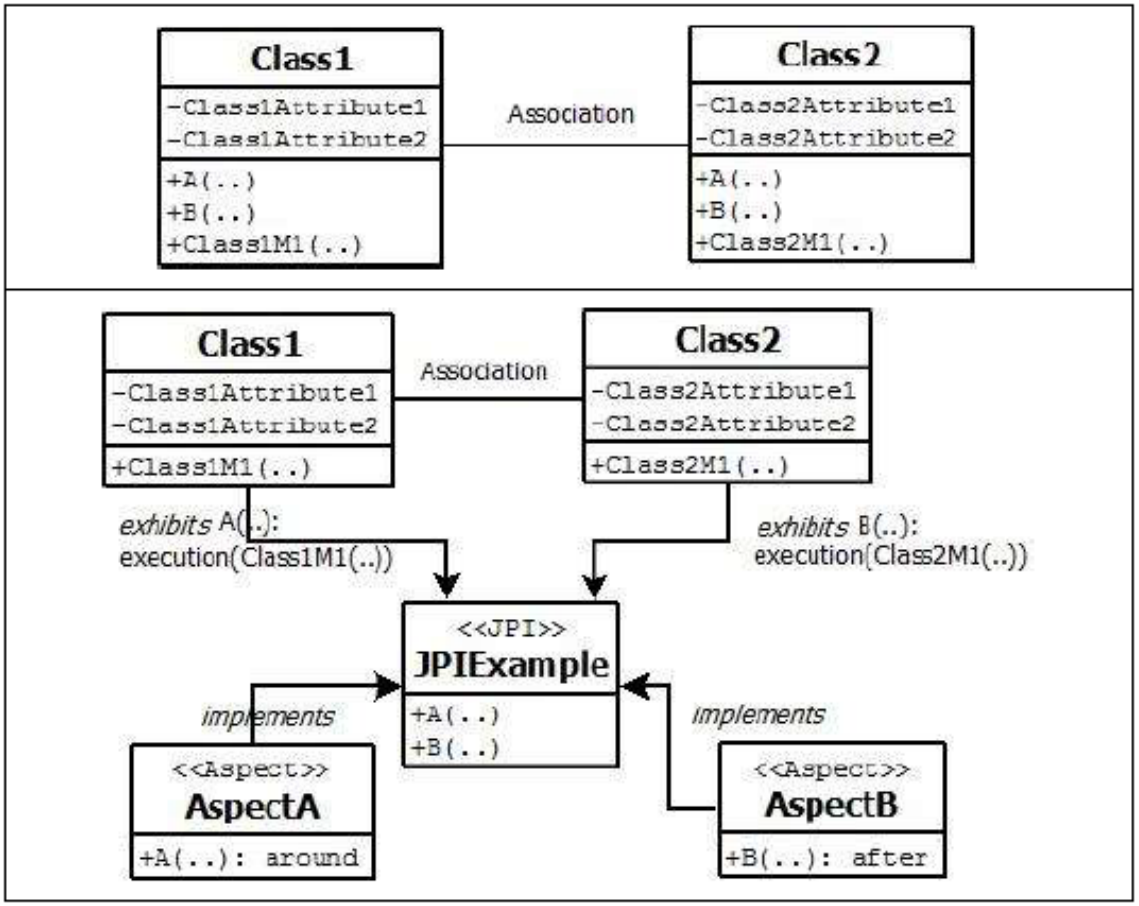
Categoría	Tipo de diagrama	Aplicación
Estructura	Diagrama de clases	Visualizar clases
	Diagrama de objetos	Estado del sistema en un momento dado
	Diagrama de componentes	Estructurar componentes y mostrar relaciones
	Diagrama de estructura compositiva	Divide los componentes o clases en sus componentes y aclara sus relaciones.
	Diagrama de paquete	Agrupa las clases en paquetes, muestra la jerarquía y la estructura de los paquetes.
	Diagrama de distribución	Distribución de componentes a los nodos informáticos
	Gráfica de perfil	Ilustra contextos de uso a través de estereotipos, condiciones límite, etc.
Comportamiento	Diagrama de casos de uso	Representa varias aplicaciones
	Diagrama de actividades	Describe el comportamiento de diferentes procesos (paralelos) en un sistema.
	Diagrama de máquina de estados	Documenta cómo un objeto es movido de un estado a otro por un evento.
Comportamiento : interacción	Diagrama secuencial	Secuencia temporal de las interacciones entre objetos
	Diagrama de comunicación	Distribución de roles de los objetos dentro de una interacción
	Diagrama de tiempos	Limitación de tiempo para los acontecimientos que conducen a un cambio de estado
	Diagrama de interacción	Secuencias y actividades interactivas

Fuente: https://www.ionos.es/

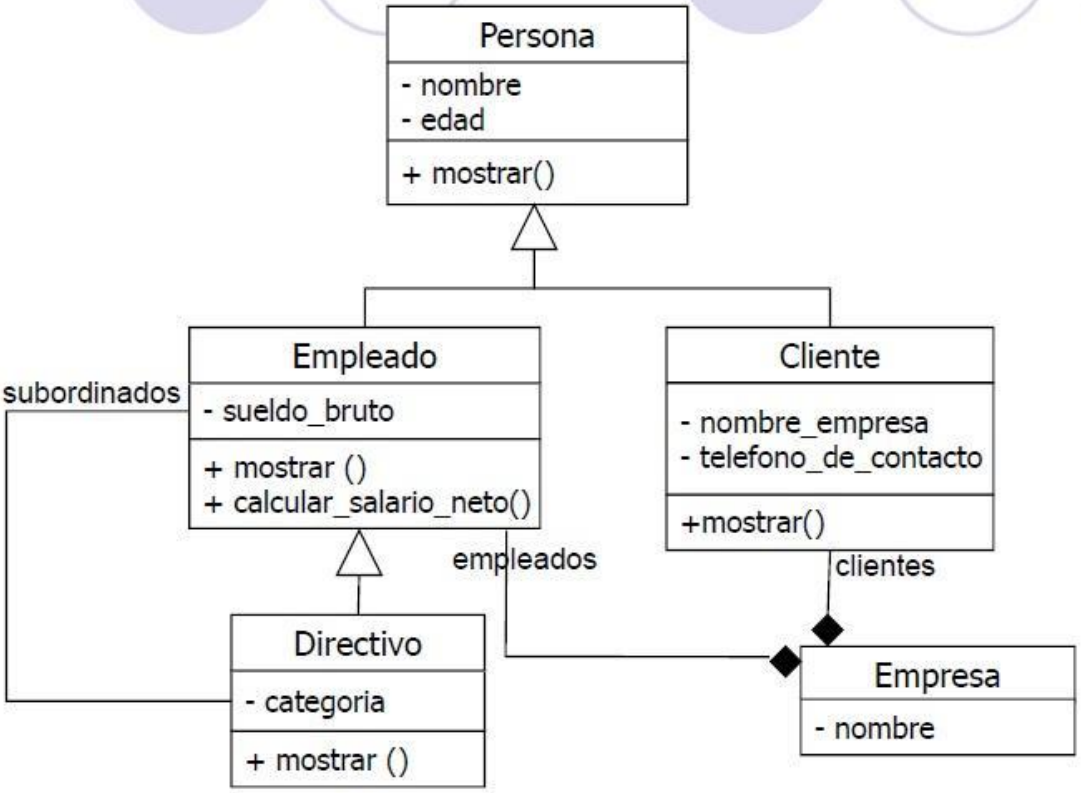
Oficial: https://www.uml.org/resource-hub.htm

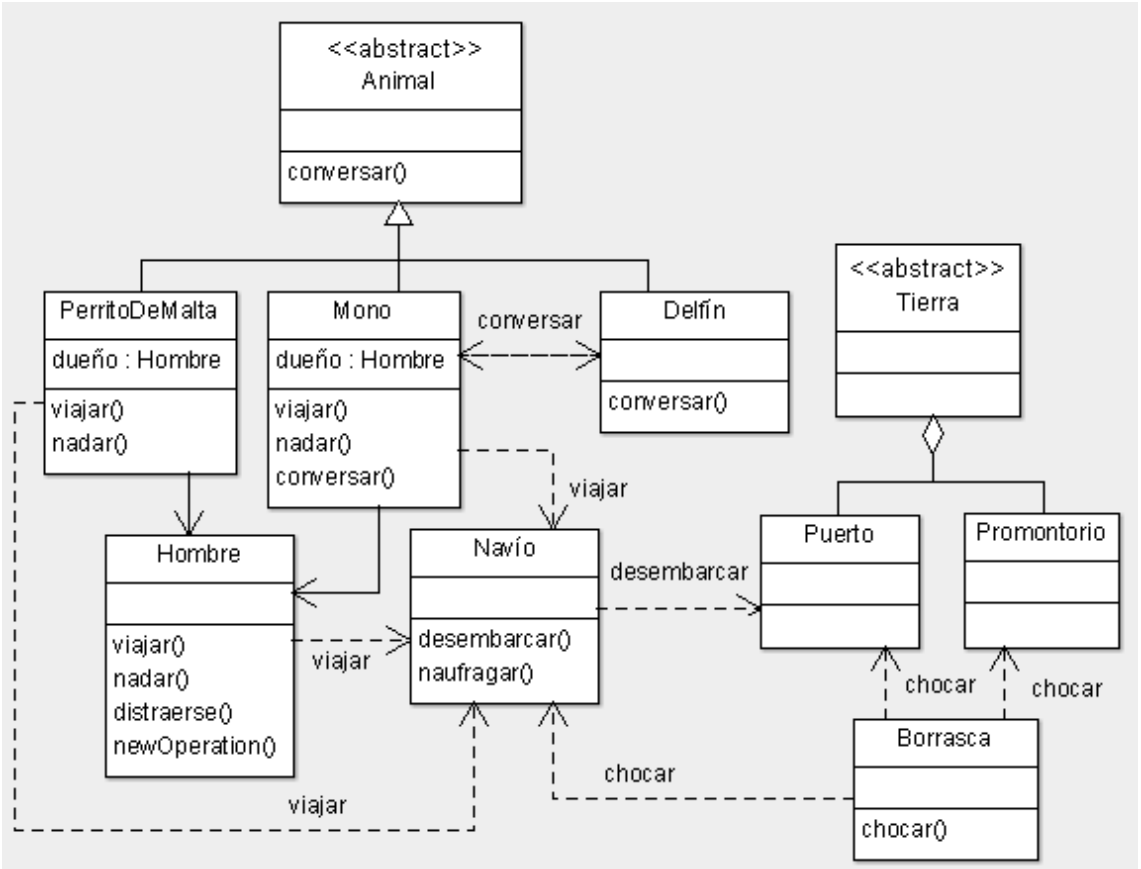





Class1
-Class1Attribute1
-Class1Attribute2
+A(..)
+B(..)
+Class1M1(..)
Association
Class2
-Class2Attribute1
-Class2Attribute2
+A(..)
+B(..)
+Class2M1(..)
Class1
-Class1Attribute1
-Class1Attribute2
+Class1M1(..)
Association
Class2
-Class2Attribute1
-Class2Attribute2
+Class2M1(..)
exhibits A(..):
execution(Class1M1(..))
exhibits B(..):
execution(Class2M1(..))
<<JPI>>
JPIExample
+A(..)
+B(..)
implements
implements
<<Aspect>>
AspectA
+A(..): around
<<Aspect>>
AspectB
+B(..): after

Ejemplo


Persona
- nombre
- edad
+ mostrar()
Empleado
- sueldo_bruto
+ mostrar ()
+ calcular_salario_neto()
Cliente
- nombre_empresa
- telefono_de_contacto
+mostrar()
subordinados
empleados
clientes
Directivo
- categoria
+ mostrar ()
Empresa
- nombre


<<abstract>>
Animal
conversar()
PerritoDeMalta
dueño : Hombre
viajar()
nadar()
Mono
dueño : Hombre
viajar()
nadar()
conversar()
conversar
Delfín
conversar()
<<abstract>>
Tierra
viajar
Hombre
viajar()
nadar()
distraerse()
newOperation()
Navío
desembarcar()
naufragar()
desembarcar
Puerto
Promontorio
viajar
chocar
chocar
Borrasca
chocar()
chocar
viajar