UML

El Lenguaje Unificado de Modelado

Ingeniería de Software II

El Lenguaje Unificado de Modelado

- **¿Qué es UML?**
- Diagrama de Casos de Uso
- Diagrama de Clases
- Notas
- Diagrama de Paquetes
- Diagrama de Objetos
- Diagrama de Secuencia
- Diagrama de Actividad
- Diagrama de Interacción en Visión General
- Diagrama de Comunicación
- Diagrama de Estados
- Diagrama de Componentes
- Diagrama de Despliegue

¿Qué es UML?

- UML es el Lenguaje Unificado de Modelado (*Unified Modeling Language*)
- Es un lenguaje gráfico capaz de expresar
    - Requisitos de Software
    - Arquitectura del Software
    - Diseño del Software
- Que sirve para
    - Comunicarse entre desarrolladores
    - Comunicarse con los clientes
    - Usar herramientas de generación automática de código
- http://www.uml.org

- OMG
    - Object Managment Group
    - Es un consorcio internacional de estandarización de tecnologías orientadas a objetos
    - http://www.omg.org/
    - Entre otras cosas estandariza el estándar UML

- Consiste en un conjunto integrado de diagramas definidos para ayudar a los desarrolladores de software y de sistemas a realizar las tareas de análisis y diseño:
    - Especificación
    - Visualización
    - Diseño Arquitectónico
    - Construcción
    - Simulación y pruebas
    - Documentación

Abstracción

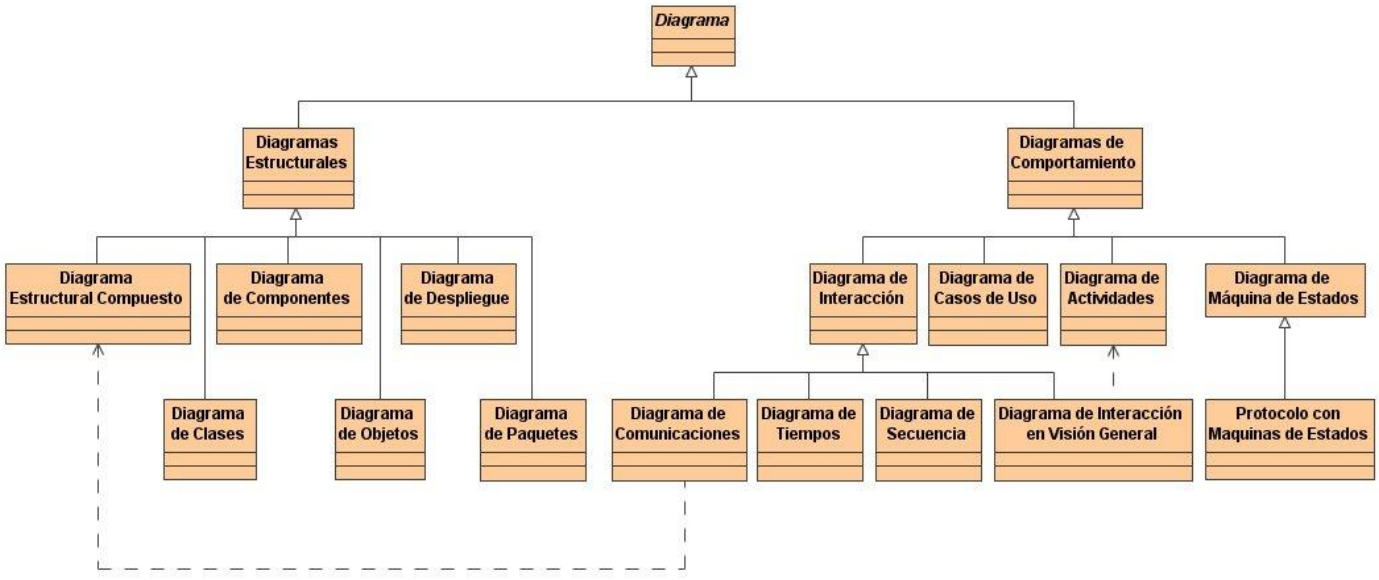
- La técnica de hacer un modelo de tus ideas del mundo es el uso de la **abstracción**
    - Por ejemplo, un mapa es un modelo del mundo, no el mundo en miniatura
- En los diagramas UML se muestra una abstracción del sistema, no todo el sistema, con el objetivo de que sea fácil de entender

Puntos de vista

- UML permite crear diagramas que reflejan diferentes **puntos de vista** del mismo sistema.
    - Por ejemplo, hay mapas físicos, mapas políticos, mapas históricos … todos sobre el mismo mundo
- Esto permite mostrar ciertos aspectos y ocultar otros para que sean más fáciles de comprender

Tipos de diagramas

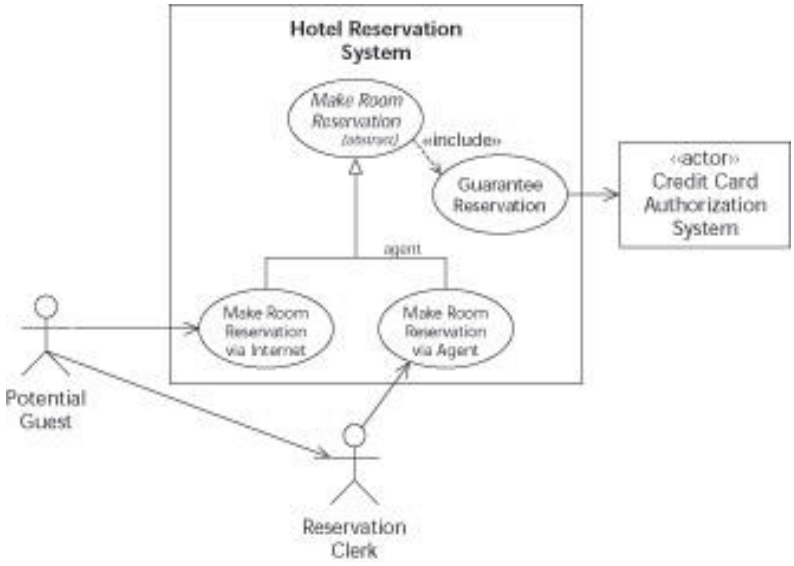
- **Diagramas Estructurales:** Muestran los elementos de construcción del sistema. Características que no cambian con el tiempo
- **Diagramas de Comportamiento:** Muestra como el sistema responde a las peticiones o evoluciona con el tiempo.
- **Diagramas de Interacción:** Engloba a ciertos diagramas de comportamiento que muestran el intercambio de mensajes dentro de un grupo de objetos que cooperan (colaboración) para obtener un objetivo



Casos de uso

Diagrama de Casos de Uso

- Muestra los servicios que los actores (usuarios y otros sistemas) pueden pedir al sistema
- No se muestran en este módulo por estar más relacionados con el Proceso Unificado de Desarrollo



¿Qué son los casos de uso?

Es una técnica para especificar el comportamiento de un sistema

"Un caso de uso es una secuencia de interacciones entre un sistema y alguien o algo que usa alguno de sus servicios."

Ejemplo: Un sistema de ventas debe ofrecer servicio para ingresar nuevo pedido de un cliente. Cuando este "cliente" accede al servicio estamos diciendo que está *ejecutando* el caso de uso Ingresando Pedido

Casos de uso y UML

A pesar de que UML (Lenguaje Unificado de Modelado) es considerada una técnica de Análisis Orientado a Objetos hay que destacar:

- No se tiene que entender a un sistema como un conjunto de objetos que interactúan (premisa básica del análisis orientado a objetos "*clásico*")
- El éxito de los casos de uso propone que la mejor forma de empezar a entender un sistema es a partir de los servicios o funciones que ofrece a su entorno, independientemente de los objetos que interactúan dentro del sistema para proveerlos.
- Los casos de uso son independientes del método de diseño que se utilice, y por lo tanto del método de programación.
- Luego de documentar los requerimientos de un sistema con casos de uso:
    - Se puede diseñar un sistema "estructurado" (manteniendo una separación entre datos y funciones)
    - O un sistema Orientado a Objetos
    - Esto da más flexibilidad al método, y probablemente contribuya a su éxito.

Importancia de los Casos de uso

Como ya se ha mencionado, los diagramas de casos de uso se utilizan para reunir los requisitos de uso de un sistema. Dependiendo de sus necesidades, puede utilizar esos datos de diferentes maneras:

- Identificarlas funciones y la forma en que los roles interactúan con ellas
- Para una visión de alto nivel del sistema.
- Los casos de uso son independientes del método de diseño que se utilice, y por lo tanto del método de programación.
- Identificar los factores internos y externos

Características de los Casos de uso

Los casos de uso tienen las siguientes características:

1) Están expresados desde el punto de vista del actor. Que tiene que hacer, no el cómo

2) Se documentan con texto informal.

3) Describen tanto lo que hace el actor como lo que hace el sistema cuando interactúa con él, aunque el énfasis está puesto en la interacción.

4) Son iniciados por un único actor.

5) Están acotados al uso de una determinada funcionalidad –claramente diferenciada– del sistema.

Componentes del Diagrama de Caso de Uso

- Actor
    - Son cualquier entidad que desempeñe un papel.
    - Se representan con dibujos simplificados de personas, llamados en inglés “stick man” (hombres de palo)
- Caso de uso (funcionalidades)
    - Es una función o una acción dentro del sistema.
    - Se representan con un óvalo y nombrado con la función
- Relaciones
    - Permiten unir dos casos de usos, cuyos eventos están unidos
    - Se representan con flechas con línea punteada
- Sistema
    - Define el alcance del caso de uso
    - Se representa con un rectángulo

Relaciones de diagramas de Caso de Uso

Hay cinco tipos de relaciones en un diagrama de casos de uso. Son:

- Asociación entre un actor y un caso de uso
- Generalización de un actor
- Ampliar la relación entre dos casos de uso
- Incluir la relación entre dos casos de uso
- Generalización de un caso de uso

Cómo crear un diagrama de Caso de Uso

- Identificar de los actores
- Identificar de los casos de uso
- Identificar los límites (alcances)

Ejemplo de un diagrama de caso de uso de un cajero automático

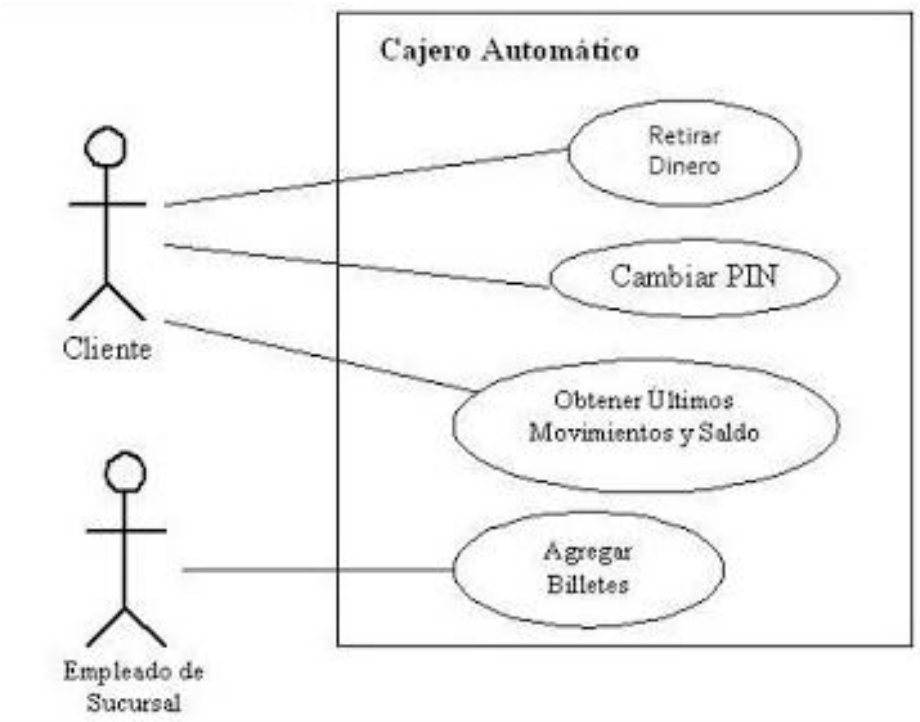
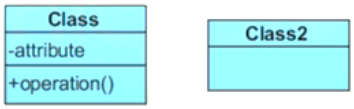


Diagrama de clases

- Entendiendo el diagrama de clases
- Clasificador
- Objeto
- Atributos
- Operaciones
- Visibilidad
- Estereotipos
- Asociación
- Multiplicidad
- Asociación reflexiva
- Clase asociada
- Agregación
- Composición
- Generalización
- Realización
- Dependencia
- Notas
- Restricciones
- Clase abstracta
- Interfaz proveedor
- Interfaz requerida
- Template

Diagrama de clases

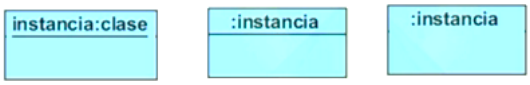
- Uno de los diagramas más comunes
- Es un diagrama estático
- Usamos un clasificador, que es una figura rectangular
- En el clasificador para la clase colocamos el nombre, los atributos y las operaciones



Como se relacionan los componentes pero no “que y como” lo hacen

Objeto

- Es posible describir un objeto, aunque no es frecuente
- Se pueden colocar objetos anónimos



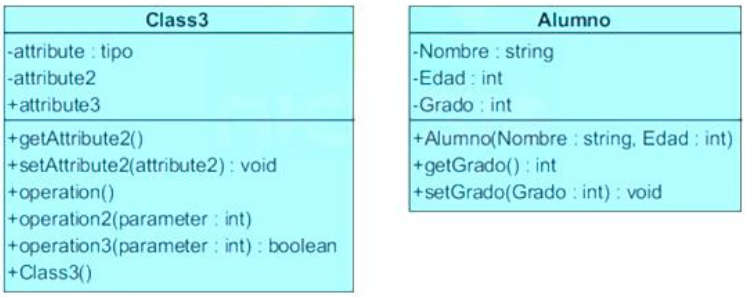
El segundo y el tercero son objetos anónimos, no son útiles desde el punto de vista de la programación, pero son válidos para UML

Atributos

- Los atributos son propiedades, datos que guarda la clase para su funcionamiento
- Visibilidad nombre : tipo

Operaciones

- Las funciones que el objeto va a hacer
- Métodos
- Visibilidad nombre(parámetros): tipo de retorno



- Se puede escribir en forma larga o completa “con parámetros”, forma corta “sin parámetros”.
- Class3() es un constructor que no recibe ni devuelve.
- Get y setter funciones de interface

Visibilidad

- + público
- - privado
- # protegido
- ~ paquete

- #protegido, los objetos que no formen parte de la cadena de herencia son privados, en cambio los que si formen parte de la cadena de herencia son públicos
- ~paquete, solo las clases definidas dentro del mismo paquete pueden ver atributos y métodos

Estereotipos

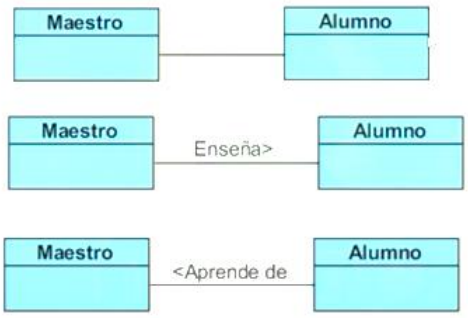
- Nos permiten extender UML
- Lo usamos en elementos que no forman parte de UML, pero que son similares a alguno en UML
- Nos permiten definir nuestros propios elementos
- También ayudan a definir roles
- Otra función es la de especializar elementos de UML
- Las dependencias también se puede estereotipar
- <<nombre>>
- El nombre indica como se esta especializando

<<NOMBRE>> es una dependencia estereotipada

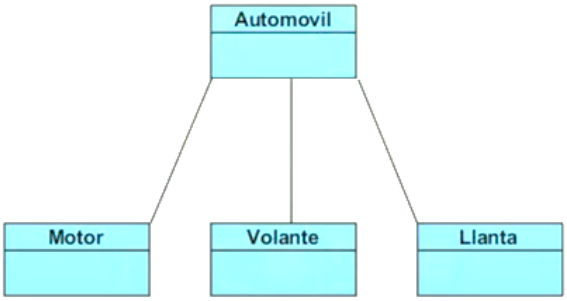
Interfaces y enumeraciones no son parte de UML

Asociación

- Muestra una relación entre clases
- Se lee generalmente como: tiene un
- Se muestra como una linea y a veces como una flecha abierta
- Se puede indicar por medio de un verbo el tipo de asociación
- < o > indican la dirección al ser colocados junto al verbo



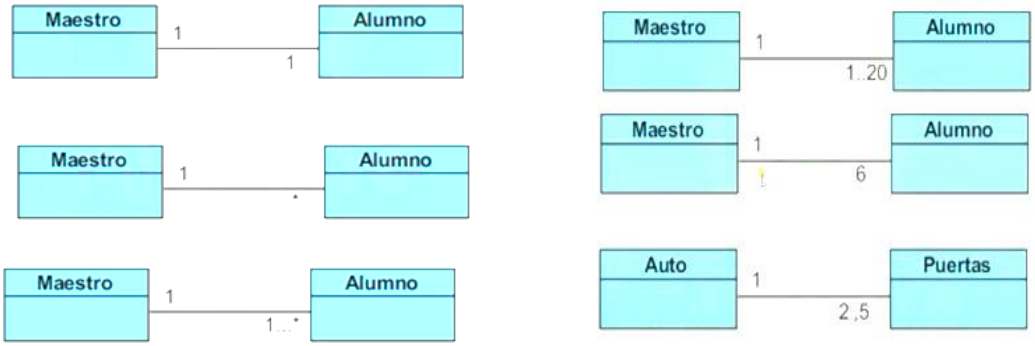
Cuando no tiene sentido se lee como MAESTRO TIENE UN ALUMNO o viceversa



Una clase puede tener varias clases asociadas

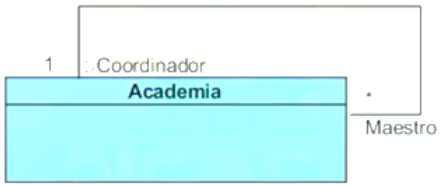
Multiplicidad

- También se le llega a llamar cardinalidad
- Indica el número de objetos en una asociación



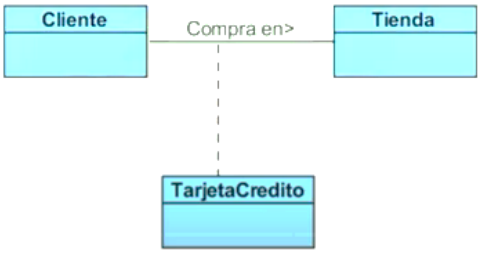
Asociación reflexiva

- El objeto de una clase puede actuar en más de un rol



Clase asociada

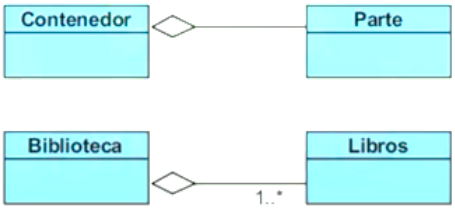
- Cuando la asociación en si se lleva a cabo por medio de una clase que tiene sus propios atributos y operaciones
- El cliente compra en la tienda por medio de la tarjeta de crédito



Clase asociada tarjeta de crédito. Se lee como Cliente compra en tienda por Tarjeta de Crédito (línea punteada)

Agregación

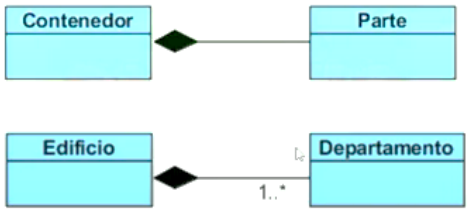
- Es parte de
- Esta hecho de



- Diamante en blanco.
- Tienen existencia por separado (si uno deja de existir, el otro puede existir)
- Libros es parte de Biblioteca o Biblioteca está hecho de Libros

Composición

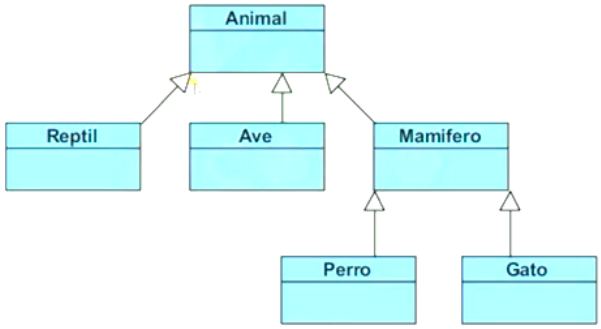
- Si el contenedor se destruye, se destruyen las partes
- Regla de no compartir →En la composición la parte puede pertenecer a un solo contenedor



- Rombo lleno
- Regla de no CONVERTIR
- Parte puede pertenecer a un solo contenedor. Ej.: Departamento solo puede pertenecer a un Edificio

Generalización

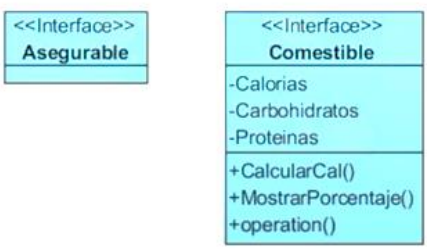
- Describe la herencia
- Es un



Reptil es un Animal

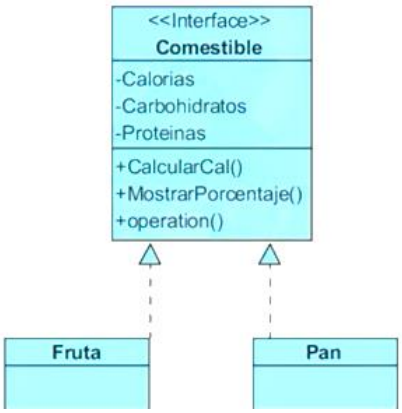
Realización

- Muestra una implementación
- Muestra la relación entre una interfaz y una clase o componente



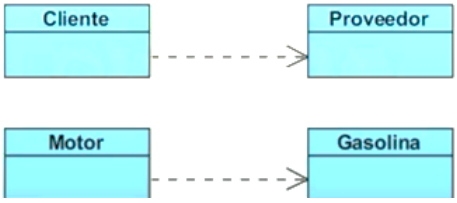
* Algunos lenguajes no permitan interfaces que tengan atributos

- Implementa a



Dependencia

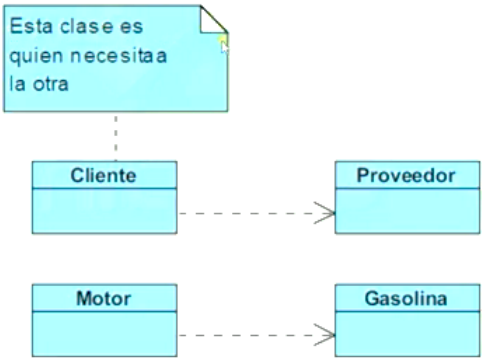
- Muestra una relación en la que una clase usa a otra clase de alguna manera
- Usa a
- Cambios en el proveedor afectan al cliente, cuidado con eso



CUIDADO: un cambio en el proveedor afecta al cliente a nivel sistema.

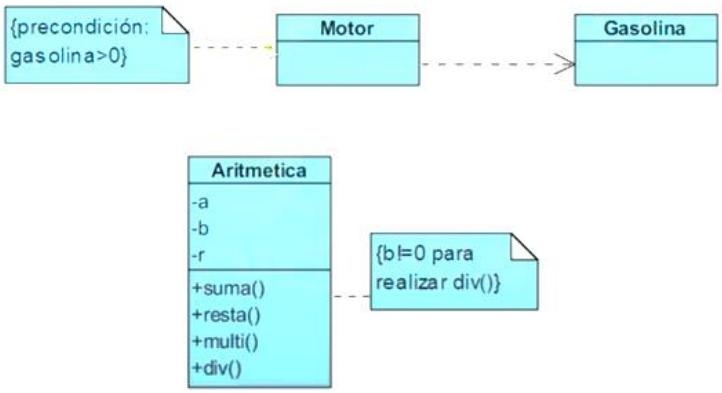
Notas

- Nos sirven para dar indicaciones o aclaraciones sobre algo
- Usamos una ancla para unirla al elemento



Restricciones

- Muestra limitaciones o condiciones de un objeto en un diagrama
- Podemos indicar precondiciones o postcondiciones
- Pueden colocarse en notas, o en algunos casos en el mismo elemento
- La multiplicidad es un tipo de restricción
- Se usan { } para indicar la restricción

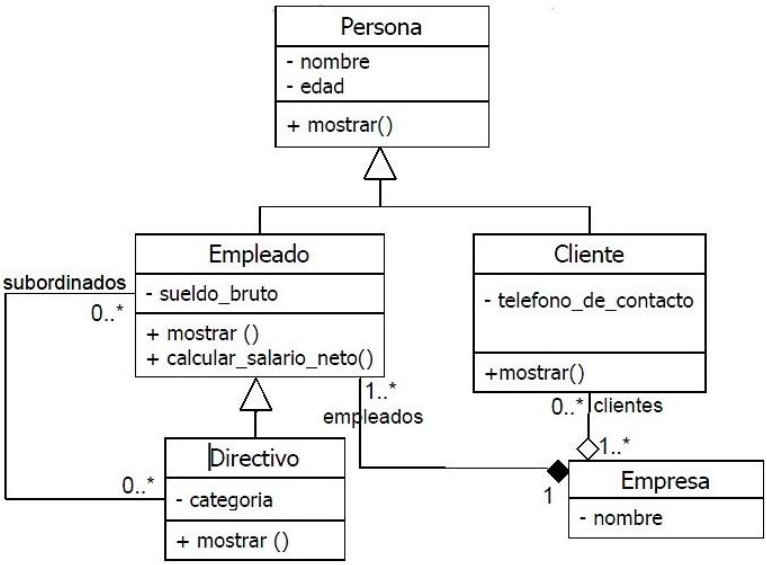


Ejercicio

Representa mediante un diagrama de clases la siguiente especificación:

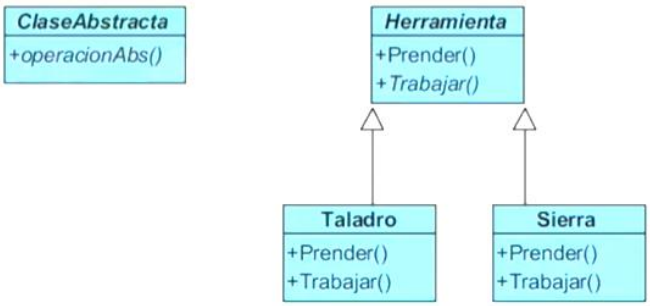
- Una aplicación necesita almacenar información sobre empresas, sus empleados y sus clientes. Ambos se caracterizan por su nombre y edad.
- Los empleados tienen un sueldo bruto, los empleados que son directivos tienen una categoría, así como un conjunto de empleados subordinados.
- De los clientes además se necesita conocer su teléfono de contacto.
- La aplicación necesita mostrar los datos de empleados y clientes

Resolución



Clase abstracta

- No puede ser instanciada
- Otras clases heredan de ella
- Se denota igual que las otras clases pero con el nombre en italica
- Puede tener operaciones abstractas



Tanto el nombre de las clases como los métodos abstractos se escriben en Itálica o Cursiva

Interfaz proveedor

- La clase provee una una implementación a la interfaz
- Lollipop

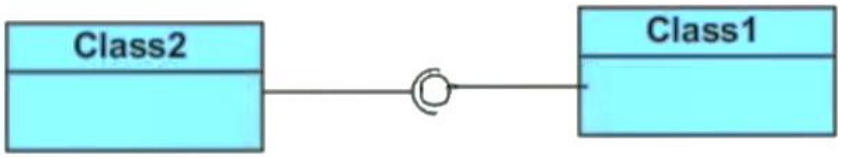
La clase provee una implementación a la interfaz proveída (círculo completo)

Interfaz requerida

- La clase requiere de la interfaz para poder hacer alguna funcionalidad
- Requiere de otra clase que la implemente

La implementación de la interfaz requerida (medio círculo) la realiza otra clase

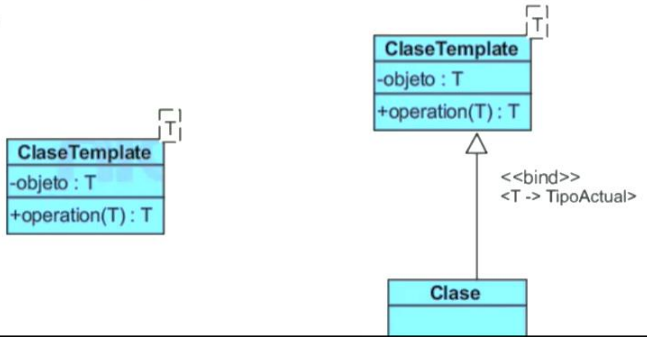
Dentro del diagrama de clases



- En el primer ejemplo vemos que Class2 tiene una interface requerida y Class1 tiene una interface proveedor.
- Class1 implementa la interface. Class2 necesita de una instancia de Class1
- Si no hay demasiado espacio en el diagrama, podemos utilizar la figura 2.

Template

- Elemento parametrizado que puede ser usado para generar otros elementos del modelo
- Funciona similar a los tipos genéricos en C# y otros lenguajes <T>
- <T -> TipoActual>



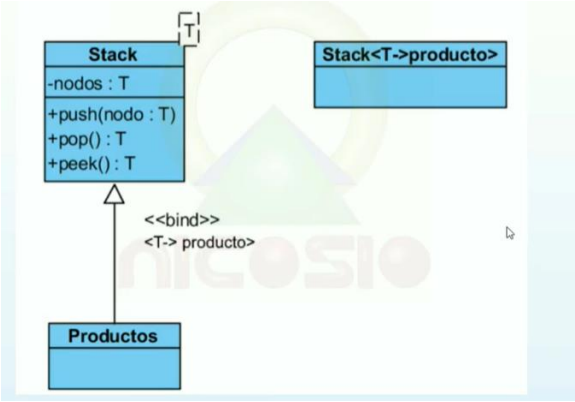
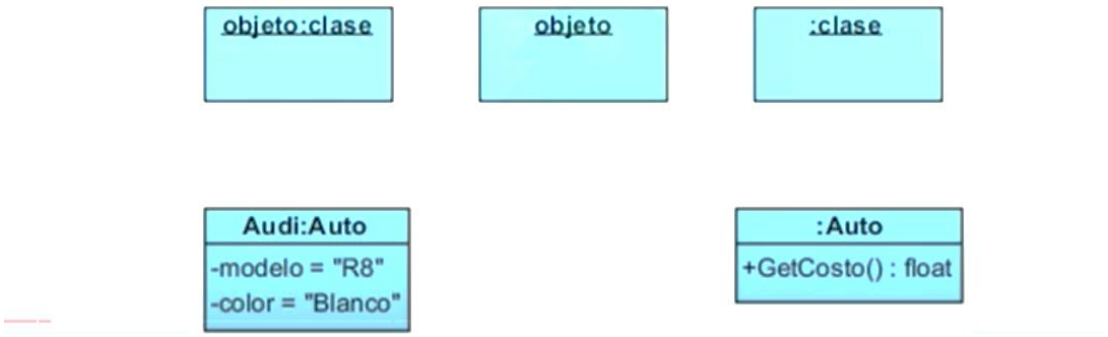


Diagrama de objetos

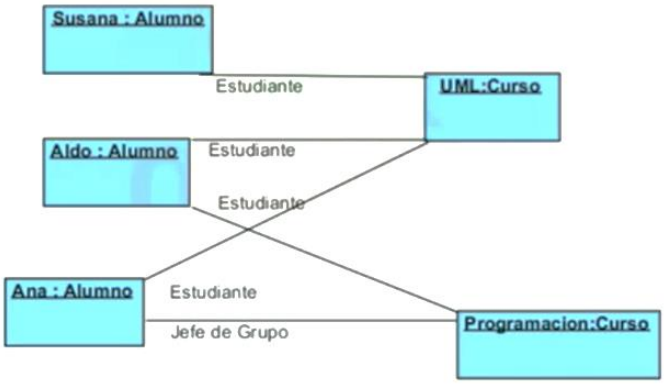
- Representan la lógica del negocio
- Muestran la forma en la que los objetos interactúan entre ellos
- Se les conoce también como diagramas de instancia



- Se usan para mostrar el estado del sistema en un momento concreto
- El diagrama se enfoca en la parte lógica y no tanto a lo estructural.
- El nombre del objeto y la clase van subrayados

Conectando objetos

- Los objetos se conectan por medio de una linea que es nombrada
- Cuando hay una relación entre objetos o clases, está debe de mostrarse en ambos diagramas



- Si en el diagrama de clases aparece una asociación, en el diagrama de objetos puede aparecer un **enlace**
- Se puede poner el nombre de la asociación (subrayado)
- Se puede poner el nombre del rol cerca del objeto (no subrayado)

Ejemplo

Diagrama de clases

Nos da información acerca de la estructura

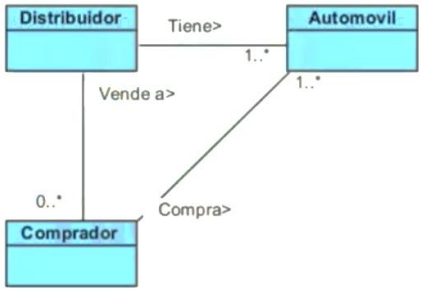


Diagrama de Objetos

Nos da más información acerca de la lógica como tal

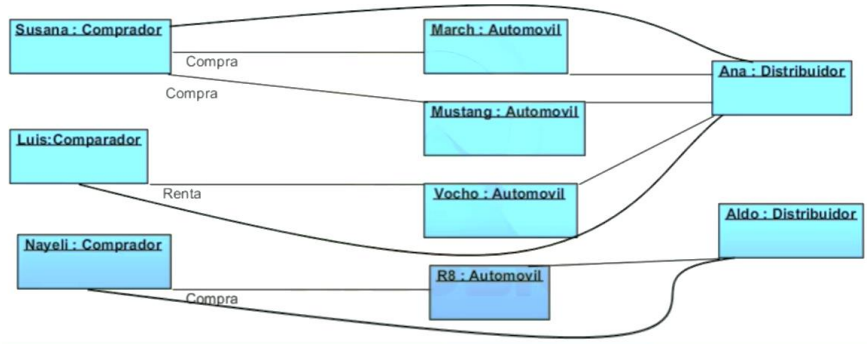


Diagrama de estados

- Entendiendo el concepto
- Estado
- Transición
- Actividades

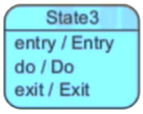
Entendiendo el concepto, estado y transición

La máquina de estados tiene un número finito de estados, es decir, se conocen de antemano. Nos vamos a concentrar en el de comportamiento.

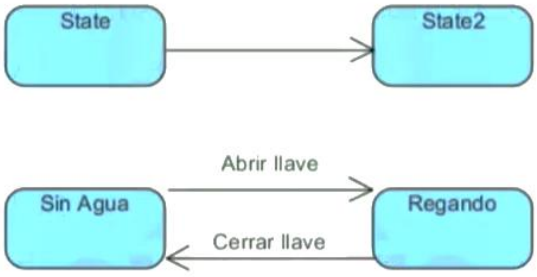
- Nos sirve para mostrar el comportamiento del sistema, ya sea:
  - Caso de uso
  - Clase
  - Sistema completo
- El estado lleva en su interior actividades
- La transición lleva una etiqueta que se conoce como firma
  - <gatillo>[<guarda>]/<efecto>
    - Gatillo es el evento
    - Guarda es la condición
    - Efecto es la actividad que sucede (no después) durante la transición

Actividades

- El estado puede estar activo o inactivo
- Tenemos tres posibles actividades
  - Entry
  - Do
  - Exit



Ejemplos



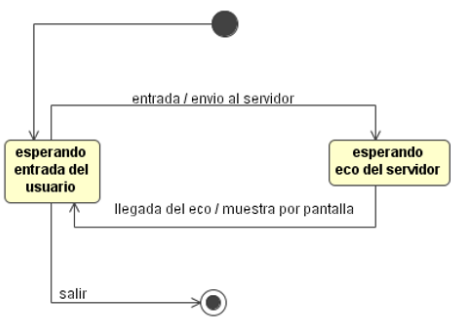


Diagrama de secuencia

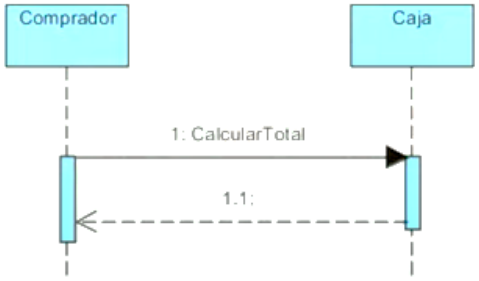
- Los diagramas de secuencia muestran como los objetos se intercambian mensajes a lo largo del tiempo
- Indica la secuencia de interacción, el orden en que suceden
- Se leen de arriba hacia abajo, tomando el tiempo en el eje vertical
- Muestra la secuencia de acciones que pasan en el sistema
- Los objetos se muestran como rectángulos y algunas veces también se los llama participantes
- De ellos sale una línea de vida, donde ocurrirán los envíos o recepciones de mensajes
- Los objetos no necesariamente son instancias de clases, pueden representar también una parte del sistema
- Los mensajes pueden estar escritos en lenguaje natural o con una sintaxis más precisa y cercana a los lenguajes de programación dependiendo del nivel de abstracción

Temario

- Mensaje de llamada
- Mensaje asincrónico
- Auto mensaje
- Mensaje recursivo
- Mensaje encontrado
- Mensaje perdido
- Mensajes para crear y destruir objetos
- Límite
- Control
- Actor y Entidad
- Marcos de interacción
    - Opcional
    - Alternativo
    - Ciclos
    - Paralelas
    - Otros marcos

Mensaje de llamada

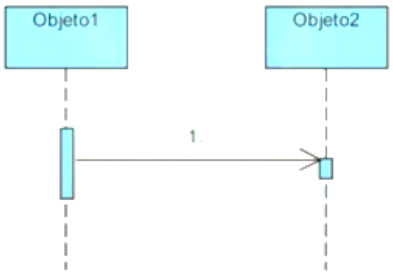
- También conocido como mensaje síncrono
- Representa la invocación a un método y espera a recibir una respuesta



- Las líneas punteadas verticales se las conoce como Línea de Vida
- Comprador invoca un método de Caja, se representa con flecha y cabeza llena. Y el mensaje es 1: CalcularTotal, donde 1 representa el número de mensaje, es decir es el primero.
- 1.1: Es el mensaje de retorno y el 1: es el mensaje asociado. Se representa con línea punteada y flecha abierta

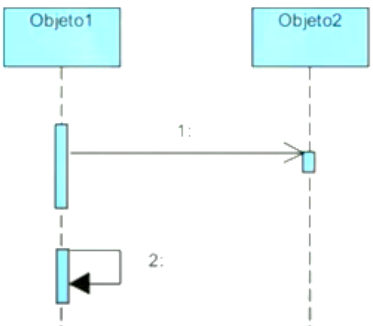
Mensaje asincróno

- No esperamos por la respuesta



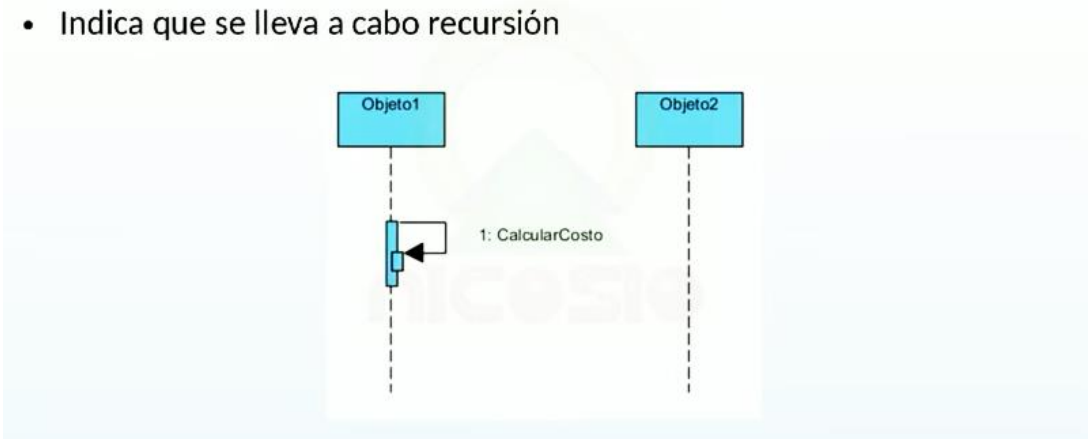
Automensaje

- El objeto se mande un mensaje a si mismo



Mensaje recursivo

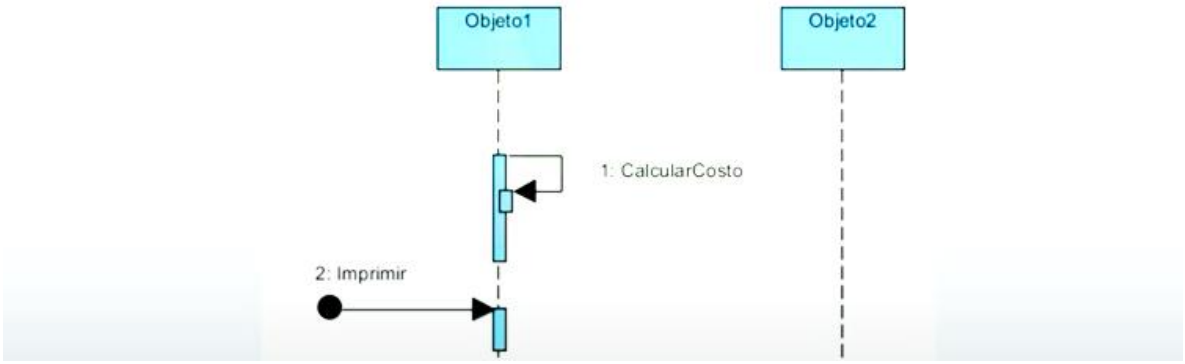
- Indica que se lleva a cabo recursión



- La recursión lleva el cuadrado pequeño a diferencia del automensaje

Mensaje encontrado

- No conocemos a quién envía el mensaje, pues no forma parte del diagrama



- Lleva una flecha de mensaje encontrado hacia el objeto y se representa con un círculo lleno

Mensaje perdido

- Quién lo recibe no forma parte del diagrama

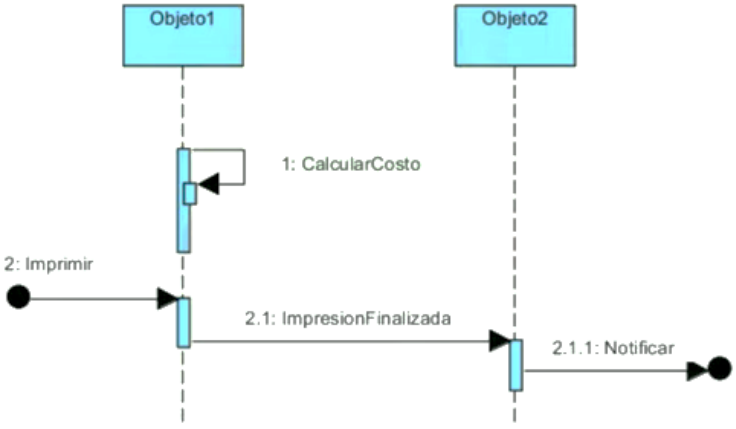


Diagrama de comunicación

- Partes
  - Objetos
  - Liga
  - Mensaje
- Mensajes anidados
- Auto mensaje
- Mensajes condicionales
- Ciclos para mensajes
- Mensajes paralelos

Los diagramas de comunicación, también conocidos como diagramas de colaboración:

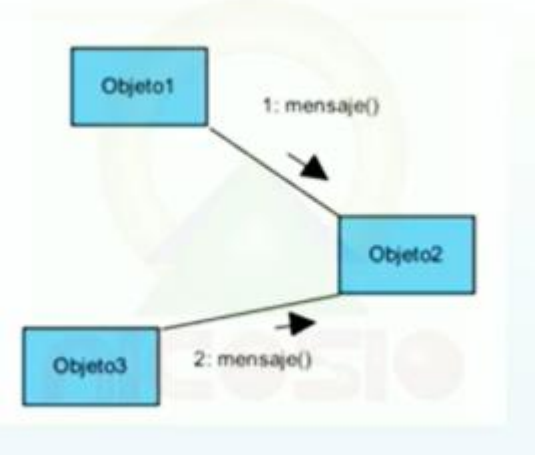
- Muestra el orden en el que suceden los mensajes
- Hay énfasis en las ligas entre los objetos
- Consta de tres elementos
  - Objetos
  - Ligas
  - Mensajes

A diferencia del diagrama de secuencia que hace enfoque en el tiempo, el diagrama de comunicación pone más énfasis en las ligas de los objetos y en el orden que suceden, pero no sabemos en qué momento o tiempo va a suceder.

Partes

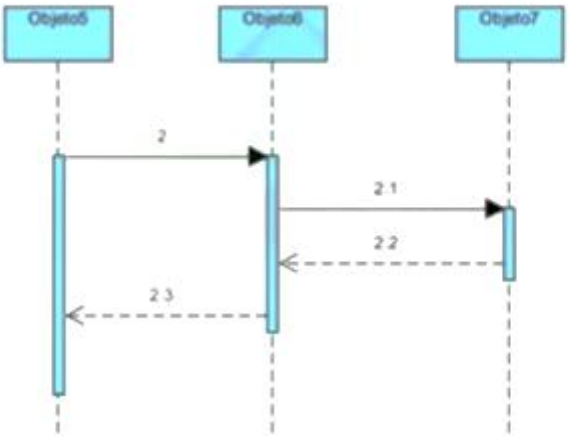
- Objeto
  - Interactúan entre ellos
  - Se muestra de la forma usual por medio de un rectángulo
- Liga
  - También se conocen como ligas de comunicación
  - Se muestran por medio de una línea
  - Si la línea conecta dos objetos eso significa que los objetos pueden comunicarse/interactuar entre ellos.
- Mensajes
  - Se colocan a lo largo de las ligas
  - Muestra la comunicación entre los objetos
  - Se usa una flecha
  - Los mensajes se nombran
  - Tienen un número que nos indica el orden de los mensajes

Ejemplo

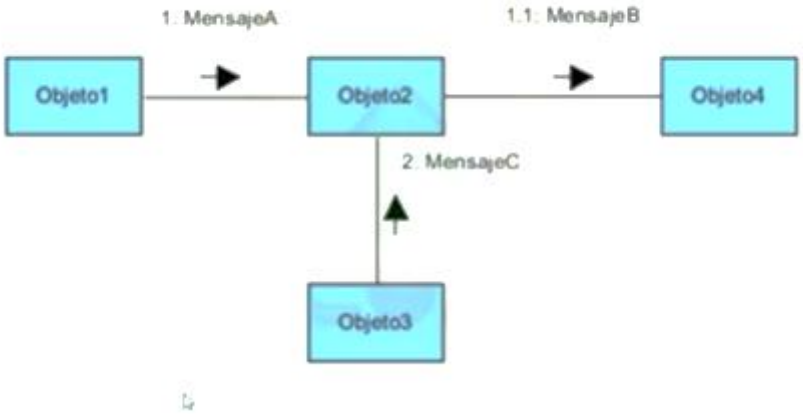


Mensajes anidados

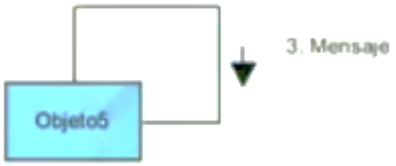
- Hay que tener cuidado con la numeración de los mensajes para mostrar el orden correctamente
- n.m



Ejemplo



Auto mensaje

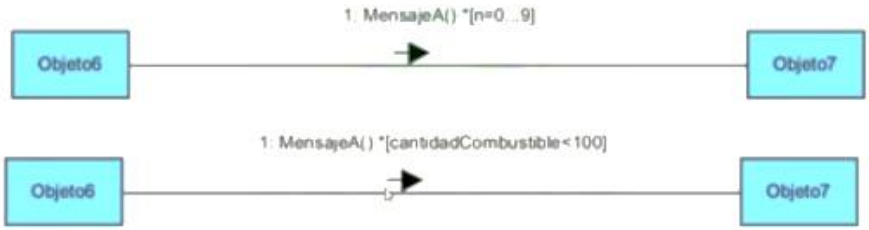


Mensajes condicionales

- El mensaje se lleva a cabo solo si la condición se evalúa como true

Ciclos para mensajes

- Los usamos cuando vamos a invocar un mensaje múltiples veces
- Se usa * para indicar ciclo y luego la condición del ciclo



Mensajes paralelos

- Los objetos envían múltiples mensajes al mismo tiempo hacía el mismo objeto