



Metodologías de Programación I

Introducción a Objetos



Objetos

Clases

Métodos

UML

Qué es un Programa Orientado a Objetos?



Un conjunto de objetos que colaboran
enviándose mensajes

Lo que implica la definición anterior...

- Sólo hay objetos
- Lo único que pueden hacer es enviar y recibir mensajes

Si querés que algo se haga...

Necesitás un objeto que lo haga, y...

Otro objeto que le envíe un mensaje....

Objeto (def de Wikipedia)

Los objetos son entidades que combinan *estado*, *comportamiento* e identidad:

- El *estado* está compuesto de datos, será uno o varios atributos a los que se habrán asignado unos valores concretos (datos).
- El *comportamiento* está definido por los procedimientos o métodos con que puede operar dicho objeto, es decir, qué operaciones se pueden realizar con él.
- La *identidad* es una propiedad de un objeto que lo diferencia del resto

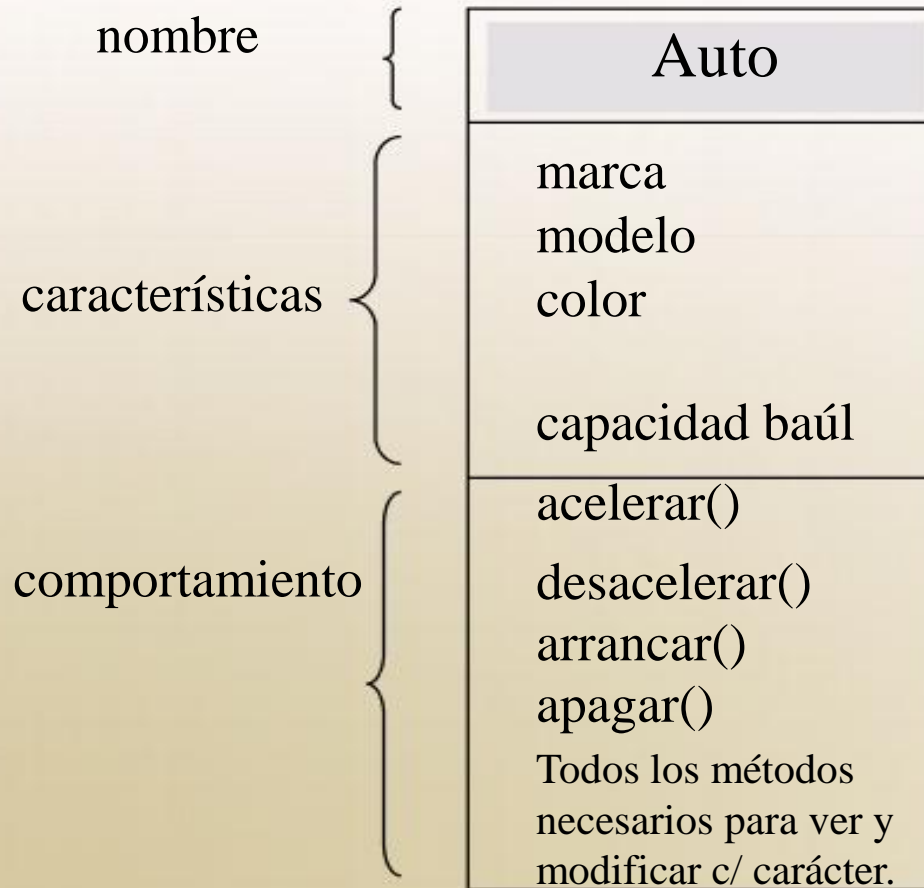
Objeto

Encapsula ***funcionalidad*** e ***información***:

- Retiene cierta ***información***.
- Sabe como realizar ciertas ***operaciones***

A diferencia del diseño estructurado aquí las “***operaciones***” y la “***información***” están juntas y sólo se puede acceder a esa información a través de esas operaciones (**encapsulamiento**).

En nuestro ejemplo




Comportamiento

- El comportamiento indica qué sabe hacer el objeto, es decir sus responsabilidades.
- Se especifica a través del conjunto de mensajes que puede recibir el objeto.

Implementación

- La implementación indica **cómo** hace el objeto para responder a sus mensajes.
- Es especificado mediante:
 - Un conjunto de colaboradores.
 - Un conjunto de métodos.
- Es privado del objeto. Ningún otro objeto debe acceder

Método

- 
- Lo único que puede hacer un objeto cuando recibe un mensaje es enviar mensajes (colaborar) a otros objetos (colaboradores).
 - Entonces, un método es simplemente
 - El conjunto de “colaboraciones” que lleva a cabo un objeto para responder un mensaje.

Método

- Al recibir un mensaje, un objeto lleva a cabo la operación mediante la ejecución de un método.
- El método es el algoritmo particular con el que el objeto realiza dicha operación.

*Un método está asociado a un mensaje.
Generalmente con el mismo nombre.*

Todo es público?

- La representación interna de un objeto es su lado “privado”, sólo tenemos acceso a aquellas partes de su estado que el objeto revela mediante su interfase pública

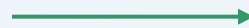


Todo es público?



Clase Auto;

Variables



Son privadas no accesibles desde afuera.

Métodos




Son privados o públicos.

End;

Estado interno de un objeto

- La información almacenada dentro de un objeto conforman su **estado interno**.
- El estado interno de un objeto puede ser cambiado sólo a través de las operaciones provistas por el objeto para dicho fin.

Encapsulamiento

- 
- *El encapsulamiento es el proceso de agrupar dentro de un objeto “**colaboradores**” y “**comportamiento**”*
 - *Es una de las principales claves para conseguir software confiable.*
 - *El encapsulamiento permite que los cambios hechos en los programas sean fiables con el menor esfuerzo*

*Una de las premisas de la programación orientada a objetos es tratar de **no** violar el encapsulamiento.*


Ocultamiento de la Información

- El estado interno conforma el lado **privado** de un objeto.
- El lado privado maneja la información que es necesaria para su funcionamiento interno, pero innecesaria para los demás objetos
- En él se especifica:
 - **Cómo** lleva a cabo los requerimientos que le hacen otros objetos
 - **Cómo** representa la información que mantiene
- La manera en que el objeto lleva a cabo estos requerimientos o trata su información “no es asunto” de los demás objetos.
- De esta manera los objetos pueden cambiar el modo de realizar ciertas operaciones o representar cierta información sin afectar el resto del sistema.

Estructuración de responsabilidades

- Un objeto sabe cumplir sólo **su** rol y el de ningún otro objeto dentro del sistema.
- No existe información fuera de objetos.
- No existen operaciones fuera de objetos.
- El diseño orientado a objetos, estructura responsabilidades mediante las preguntas: ¿qué puede hacer este objeto? y ¿qué conoce este objeto?

Clases.

- 
- Los objetos dentro de un sistema no son completamente distintos uno del otro.
 - Distintos objetos pueden comportarse de una manera muy similar.

*Se dice que los objetos que comparten el mismo comportamiento pertenecen a la misma **clase**.*

Es la declaración o abstracción de un objeto cuando se programa según el paradigma de orientación a objetos.

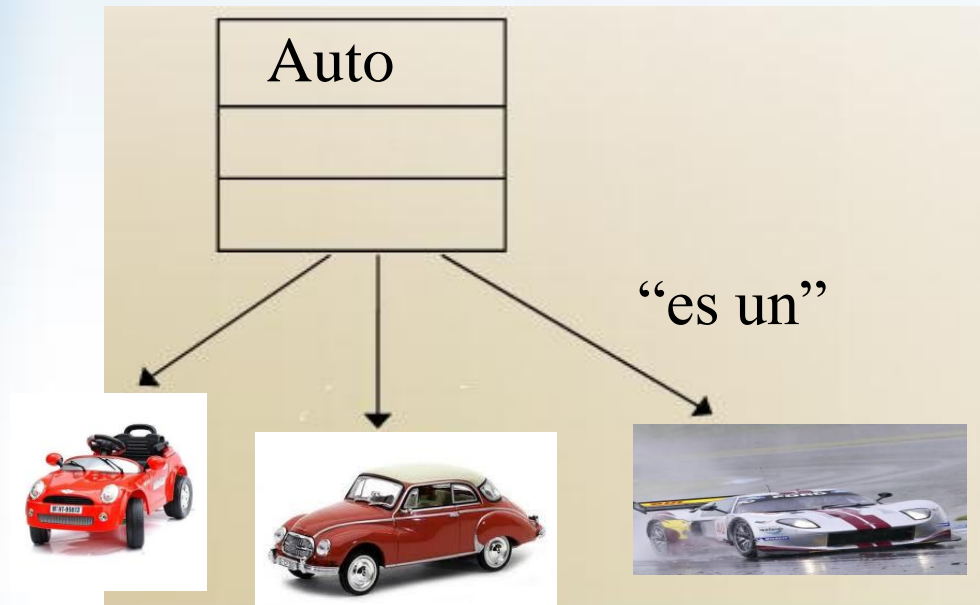
Instancias

- Los objetos que se comportan de la manera descrita en una clase son llamados **instancias** de esa clase.
- Todo objeto es instancia de alguna clase.
- Una instancia de una clase se comporta de la misma manera que las demás instancias de esa clase.
- Almacena su información en variables de instancia.

Programación Orientada a Objetos

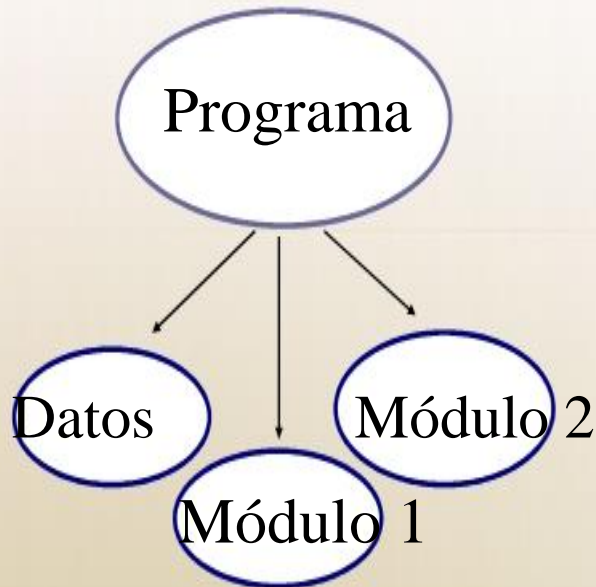
Cada uno de los diferentes autos vistos anteriormente tienen características comunes pero con valores diferentes. Es decir los tres autos tienen color pero cada uno un color diferente.

Instancia de una clase = OBJETO

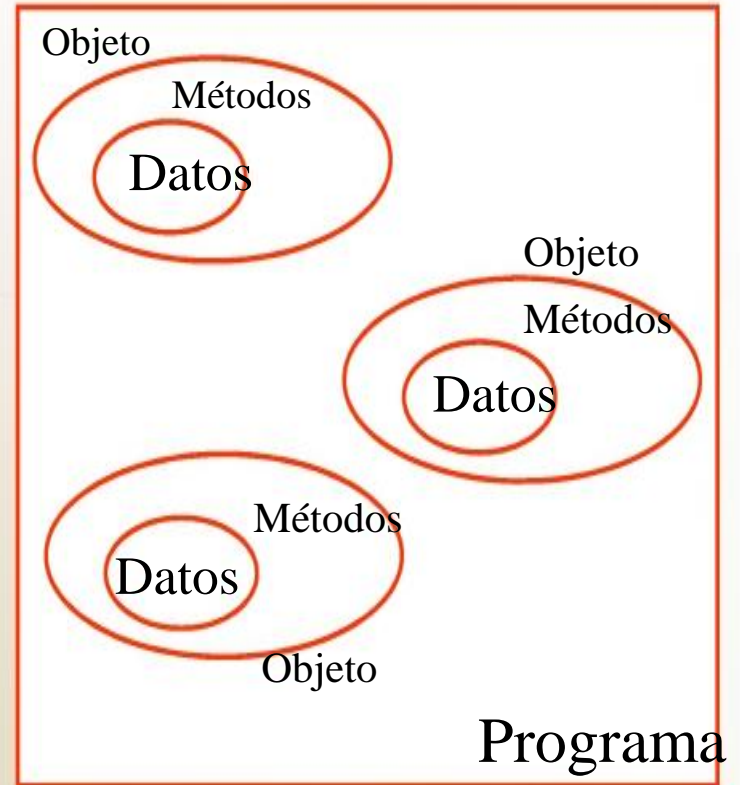


Programación Orientada a Objetos


Programación Estructurada



Programación Orientada a Objetos




Programación Orientada a Objetos



Grady Booch resume la diferencia de la siguiente forma:

“Lea las especificaciones del sistema que desea construir. Subraye los verbos si persigue un código procedimental, o los **sustantivos** si su objetivo es un programa orientado a objetos”.

Programación Orientada a Objetos




Todo lo que vemos a nuestro alrededor puede ser considerado un objeto (una computadora, un teléfono celular, un árbol, un automóvil, etc). Ejemplo: una computadora está compuesta por varios componentes (tarjeta madre, chip, disco duro y otros), el trabajo en conjunto de todos ellos hace operar a una computadora. El usuario no necesita saber como trabajan internamente cada uno de estos componentes, sino como es la interacción con ellos. Es decir, **cuando se conoce como interaccionan los componentes entre sí**, el usuario podrá armar la computadora.



UML

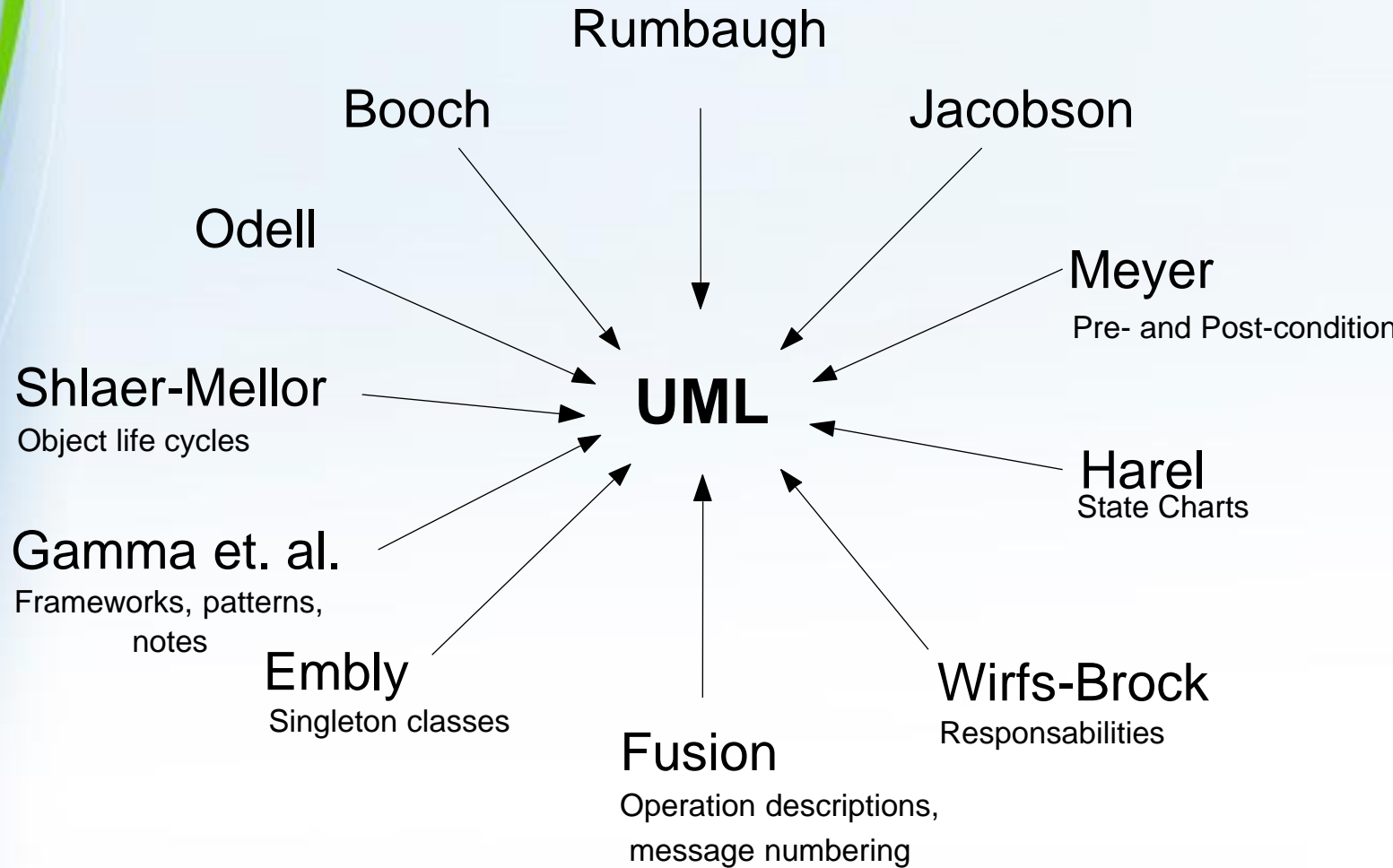
Lenguaje Unificado de Modelado

UML - Lenguaje Unificado de Modelado




EL UML es la creación de G. Booch, J. Rumbaugh e I. Jacobson. En la década de los '80 cada uno diseñó su propia metodología para el análisis y diseño orientado a objetos. Sus metodologías predominaron sobre las de sus competidores. A mediados de los '90 intercambian sus ideas y deciden trabajar en conjunto

UML aglutina otros enfoques



Ventajas de la unificación

- 
- Reunir los puntos fuertes de cada método
 - Idear nuevas mejoras
 - Proporcionar estabilidad al mercado
 - Eliminar confusión en los usuarios

Modelos

- Esencial para la comunicación entre miembros de un equipo
- Ayudan a la comprensión de sistemas complejos
- Indican QUE hará el sistema pero NO como lo hará
- Ayuda a la corrección de errores
- Ayuda a la evolución y reuso

UML y el modelado

Lenguaje gráfico para

- visualizar
- especificar
- documentar

cada una de las partes que comprende el desarrollo de software

UML y OO




- Sistemas mas complejos
- Necesidad de mayor nivel de abstracción

Utilidad de UML

- Permite especificar todas las decisiones de análisis, diseño e implementación, construyéndose modelos precisos, no ambiguos y completos.
- Permite documentar todos los pasos de un desarrollo (requisitos, arquitectura, pruebas, versiones,..)

Características de UML ...

- 
- UML es independiente del lenguaje de implementacion.
 - UML esta pensado para poder ser implementado en varios lenguajes
 - Provee una base formal para el entender el lenguaje de modelado

UML - Lenguaje Unificado de Modelado

Compuesto por elementos gráficos, que se combinan para conformar diagramas. Debido a que UML es un lenguaje, cuenta con reglas para combinar dichos elementos.

Diagramas:

- De clases
- De objetos
- De casos de uso
- De estados
- De secuencias
- De colaboraciones
- De componentes
- De distribución

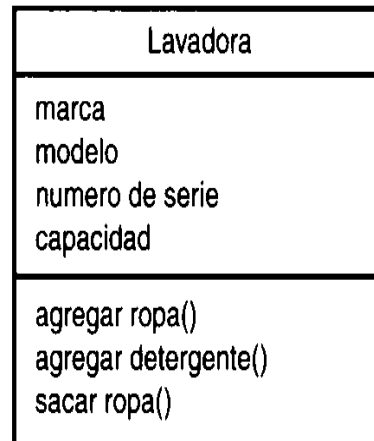
Un modelo UML indica qué hará un sistema, y no cómo se implementará.

UML –Diagrama de clases


La figura 1.1 le muestra un ejemplo de la notación del UML que captura los atributos y acciones de una lavadora. Un rectángulo es el símbolo que representa a la clase, y se divide en tres áreas. El área superior contiene el nombre, el área central contiene los atributos, y el área inferior las acciones. Un diagrama de clases está formado por varios rectángulos de este tipo conectados por líneas que muestran la manera en que las clases se relacionan entre sí.

FIGURA 1.1

*El símbolo UML
de una clase.*



CLASE: estructura



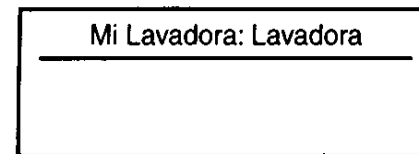
nombreClase
atributos
métodos
Responsabilidades Y restricciones

UML –Diagrama de objetos

La figura 1.2 le muestra la forma en que el UML representa a un objeto. Vea que el símbolo es un rectángulo, como en una clase, pero el nombre está subrayado. El nombre de la instancia específica se encuentra a la izquierda de los dos puntos (:), y el nombre de la clase a la derecha.

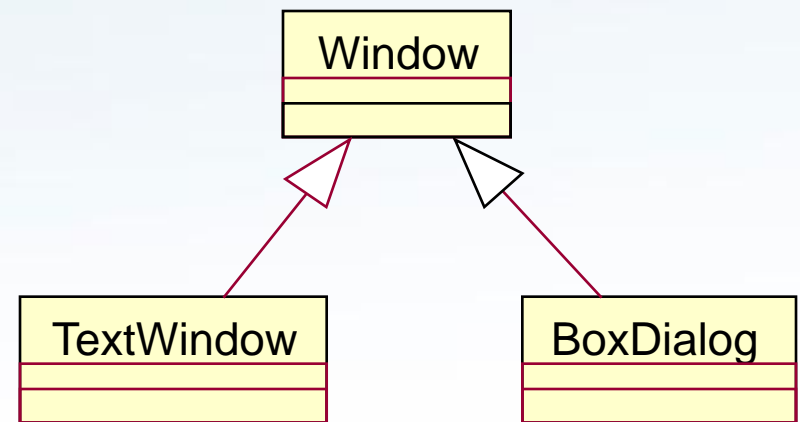
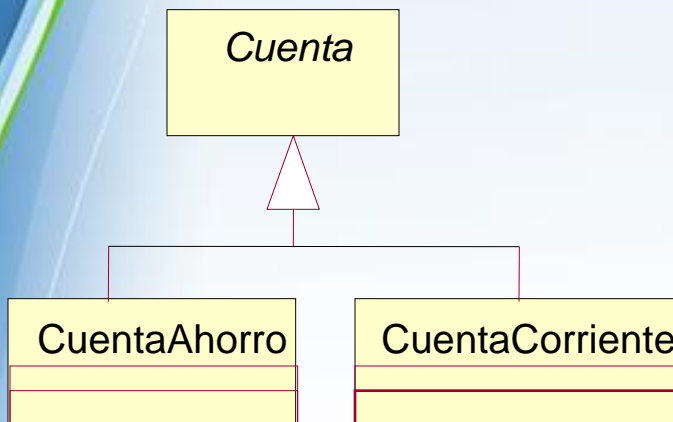
FIGURA 1.2

El símbolo UML del objeto.



Diagramas de Clases: Relaciones

- **Generalización**
– “*Es-un*”



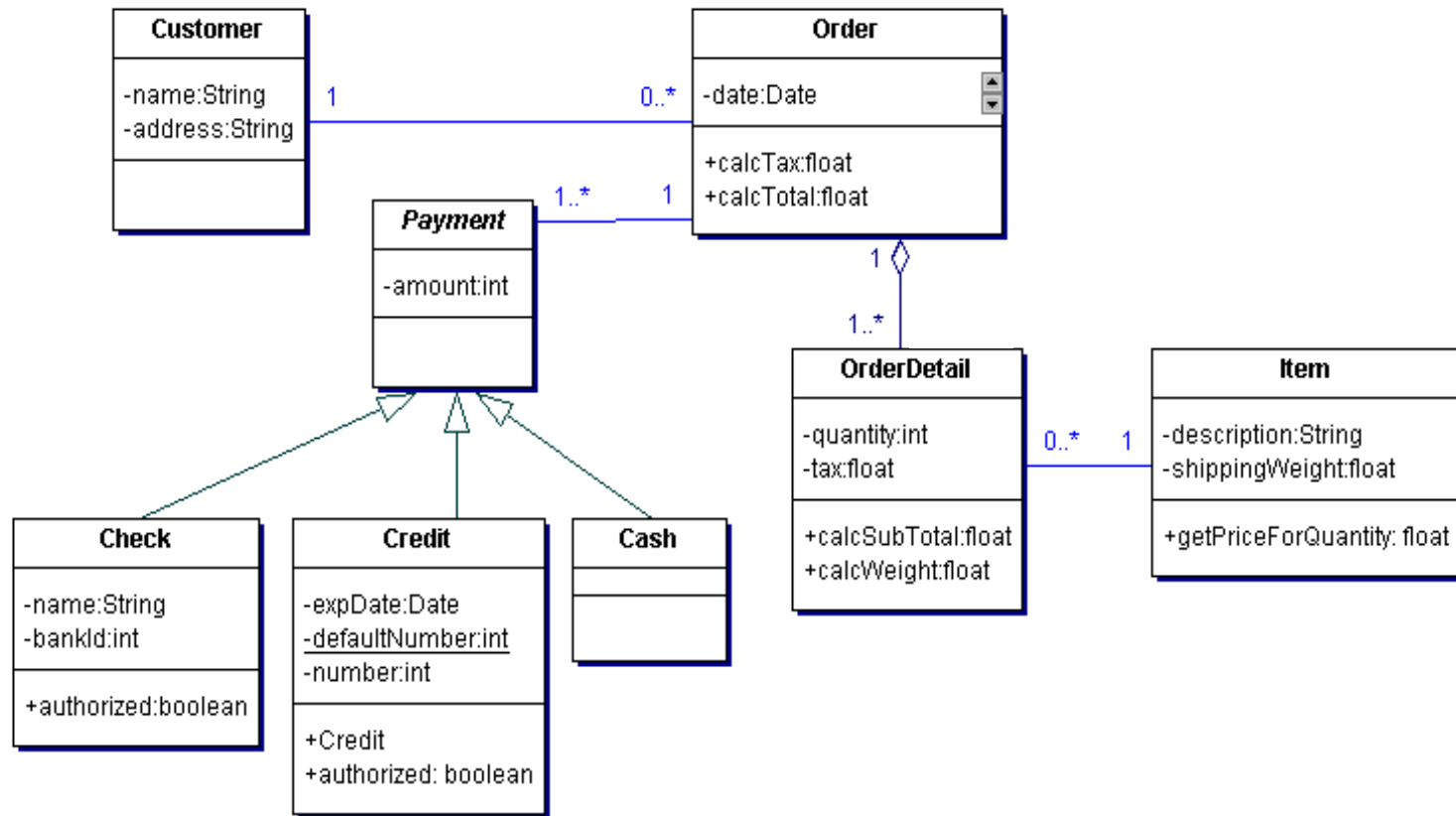
Información adicional en los compartimientos

- Variables y métodos
 - - private
 - # protected
 - + public

Clases - Ejemplo

Window {autor= Matias, versión =1.1}
+ size : 10 - visibility: Boolean = true + default-size: int = 10 # maximum-size: int = 100
+ hide() + create() <i>+display()</i>
exceptions arrayException

Diagrama de clases - Ejemplo

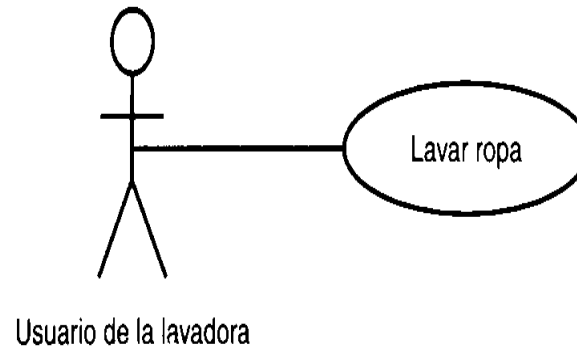


UML –Diagrama de casos de usos

Un caso de uso es una descripción de las acciones de un sistema desde el punto de vista del usuario.

FIGURA 1.3

Diagrama de casos de uso UML.

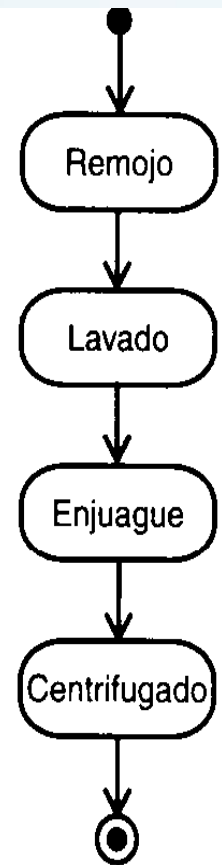


UML – Diagrama de estados

Muestra las transiciones
de un estado a otro.
Punto inicial y final.

FIGURA 1.4

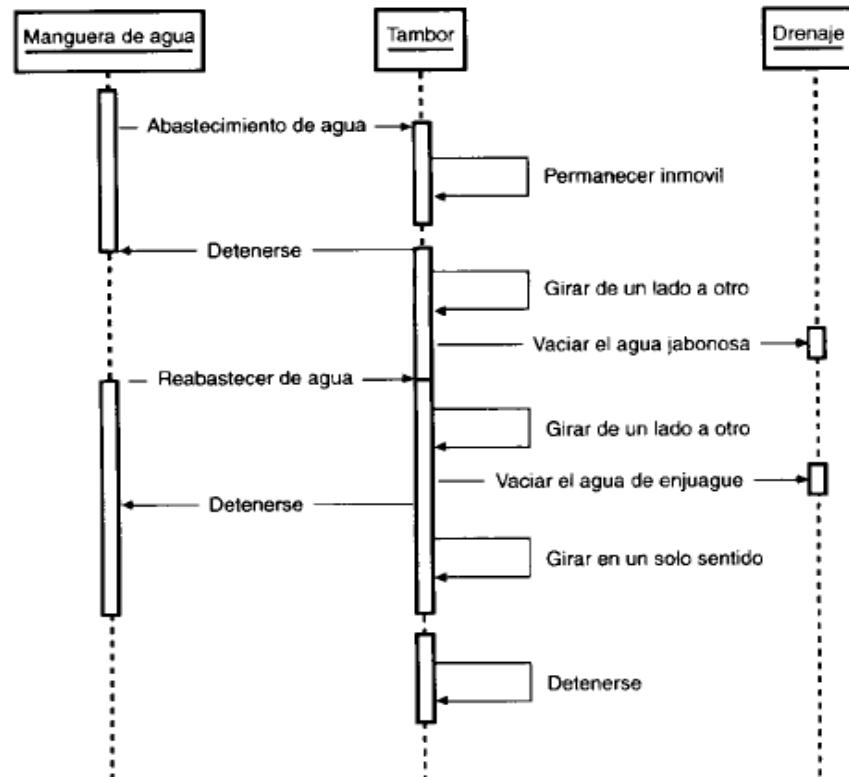
*Diagrama de estados
UML.*



UML –Diagrama de secuencia

Los diagramas de clases y objetos representan información estática. Sin embargo en los sistemas funcionales los objetos interactúan entre sí y dichas interacciones suceden en el tiempo. Los diagramas de secuencias muestran dicha interacción

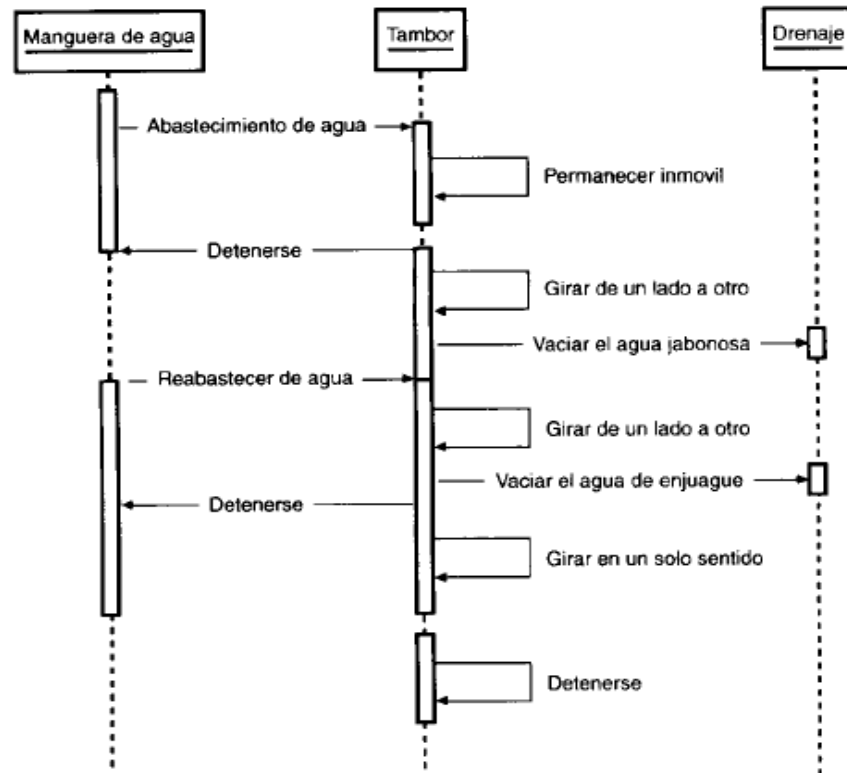
FIGURA 1.5
Diagrama de secuencias UML.



UML –Diagrama de secuencia

Los diagramas de clases y objetos representan información estática. Sin embargo en los sistemas funcionales los objetos interactúan entre sí y dichas interacciones suceden en el tiempo. Los diagramas de secuencias muestran dicha interacción

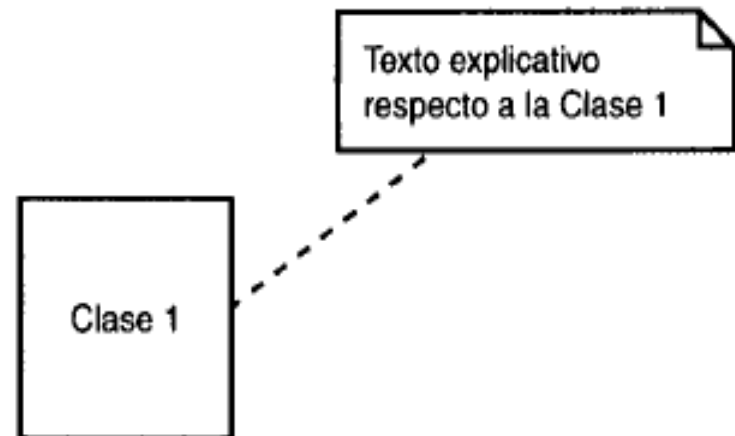
FIGURA 1.5
Diagrama de secuencias UML.



UML – Notas o Aclaraciones

FIGURA 1.11

En cualquier diagrama, podrá agregar comentarios aclaratorios mediante una nota.



UML – Mas ejemplos de diagramas

FIGURA 2.2

La adición de atributos y acciones al modelo lo acerca a la realidad.

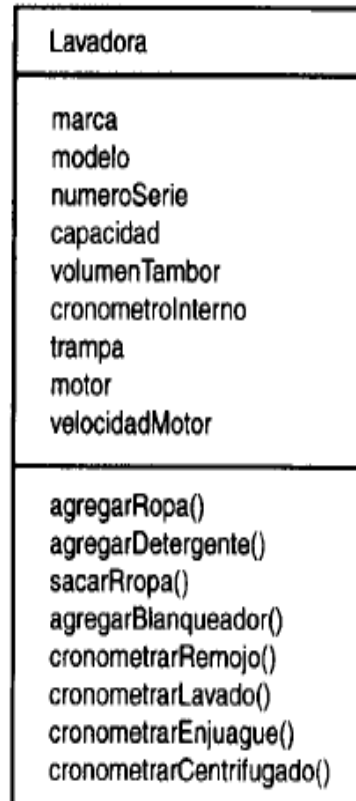
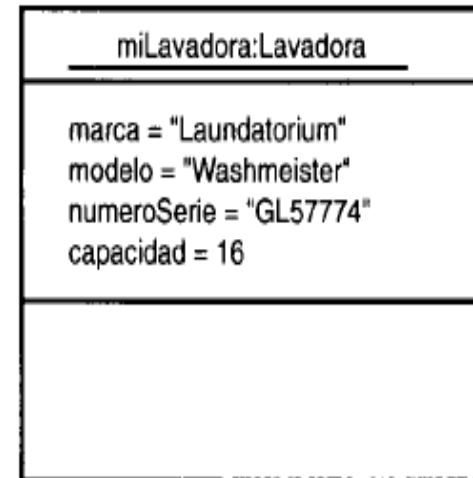


FIGURA 3.5

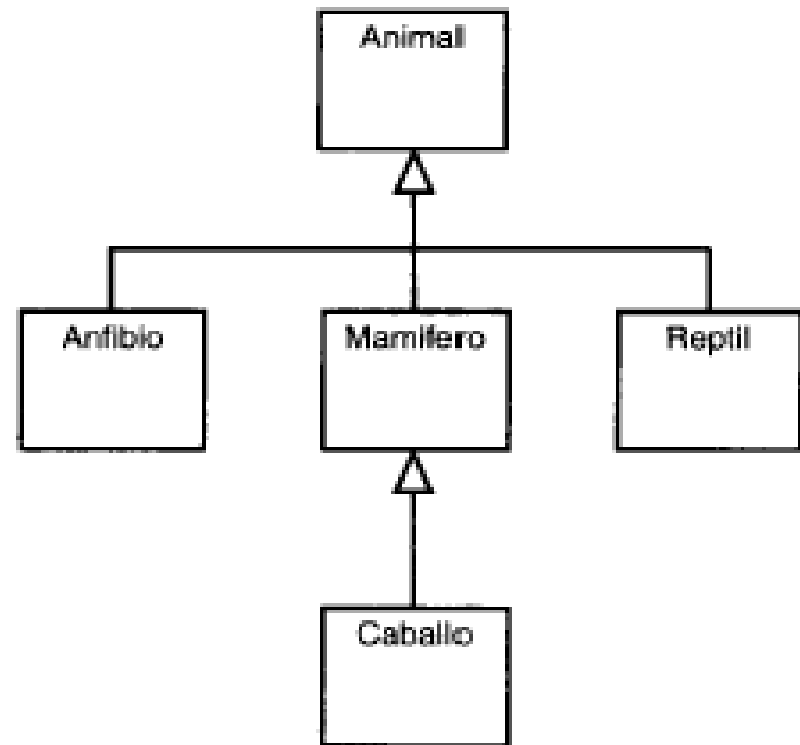
Un objeto cuenta con un valor específico en cada uno de los atributos que lo componen.



UML – Herencia

FIGURA 4.13

Una jerarquía de herencia en el reino animal.




UML

- Definición de los elementos a Utilizar

Tipos de Elementos	Grafico
Package	
Class	
Stereotipos	
Herencia	
Implementacion	
Uses - Instanciacion	
Interfaces	
Agregaciones	

UML – continuacion



Retomaremos durante el curso para profundizar los siguientes temas:

- Dependencia, composición
- Herencia
- Asociaciones y multiplicidades
- Clases Abstractas
- Diagramas de Interacción