Tema 1

Introducción a la Orientación a Objetos

Iconos


Contenidos


Objetivos / competencias


Bibliografía


Ejercicios para hacer en casa


Ejercicios para hacer en clase


Ejercicios en ordenador


Juego / competición

Objetivos generales

- Entender que el **paradigma de Programación Orientada a Objetos (POO)** supone un cambio de filosofía radical respecto a la programación estructurada.
- Adquirir una idea aproximada del significado de los principales **conceptos de orientación a objetos**, eliminando posibles creencias erróneas.
- Convencerse de la importancia de realizar un buen **diseño orientado a objetos**.
- Advertir la **diversidad** de herramientas **de diseño y lenguajes de POO**.

Contenidos y bibliografía

Lección	Título	Nº sesiones
1.1	Conceptos básicos de OO	1'5
1.2	Lenguajes y técnicas de diseño OO	

http://groups.diigo.com/group/pdoo_ugr

Etiquetas: conceptosOO, libros

Lección 1.1

Conceptos básicos de Orientación a Objetos

Objetivos de aprendizaje

- Conocer los fundamentos del **paradigma de orientación a objetos**.
- Identificar las principales diferencias del paradigma orientado a objetos respecto a otros paradigmas.
- Adquirir una idea aproximada del significado de los principales **conceptos de orientación a objetos**, eliminando posibles creencias erróneas.

Contenidos

1. El paradigma de la Orientación a Objetos
2. Conceptos básicos de Programación Orientada a Objetos
   - objeto o instancia
   - estado
   - clase
   - método
   - mensaje
   - herencia
   - polimorfismo
3. Conceptos básicos de Diseño de Software (DS) adaptados a Orientación a Objetos (OO)
   - abstracción
   - modelo
   - modularidad
   - encapsulamiento
   - reutilización

1. El paradigma de la OO

- Un **paradigma de programación** es una propuesta tecnológica que es adoptada por una comunidad de programadores, cuyo núcleo central es incuestionable.
- El paradigma de programación más empleado en la actualidad es el de **orientación a objetos**. Su núcleo central es la unión de datos y procesamiento en una entidad llamada **"objeto"** y está basado en varias técnicas, incluyendo **herencia, polimorfismo, encapsulamiento y abstracción.**
- La orientación a objetos está ligada en sus orígenes con lenguajes como Lisp y Simula, aunque el primero que acuñó el término de programación orientada a objetos fue el lenguaje **Smalltalk**.

2. Conceptos básicos de POO: **Objeto/instancia**

- Representación lógica de un elemento real o imaginario con **estado**, **responsabilidades** e **identidad** propia.

2. Conceptos básicos de POO: **Clase**

- Categoría a la que pertenece un objeto y que determina su estado y responsabilidades.

2. Conceptos básicos de POO: **Estado**

- Configuración de valores o propiedades de un objeto

¿Cuál es su **estado** como **objeto Reloj**?

Nombre: Rekih de oue

Color Fondo: Blanco

Tipo: Reloj de manecillas

Pilas: LR6 AA 1,5V

Hora: 15:19h

¿Cuál es su **estado** como **objeto Estantería**?

Nombre: Rekih de oue

Color: Negro

Dimensiones: 56 x 17 x 198 cm

Numero estantes: 5

2. Conceptos básicos de POO: **Método**

- **Acción** (responsabilidad o funcionamiento) que pueden llevar a cabo todos los objetos de una clase

¿Qué responsabilidades (métodos) podrían tener los objetos de la clase Reloj?

- Mover manecillas para dar la hora.
- Señalar las horas en punto.
- Indicar pilas gastadas.

¿Qué responsabilidades (métodos) podrían tener los objetos de la clase Estantería?

- Soportar objetos.
- Decorar una habitación.

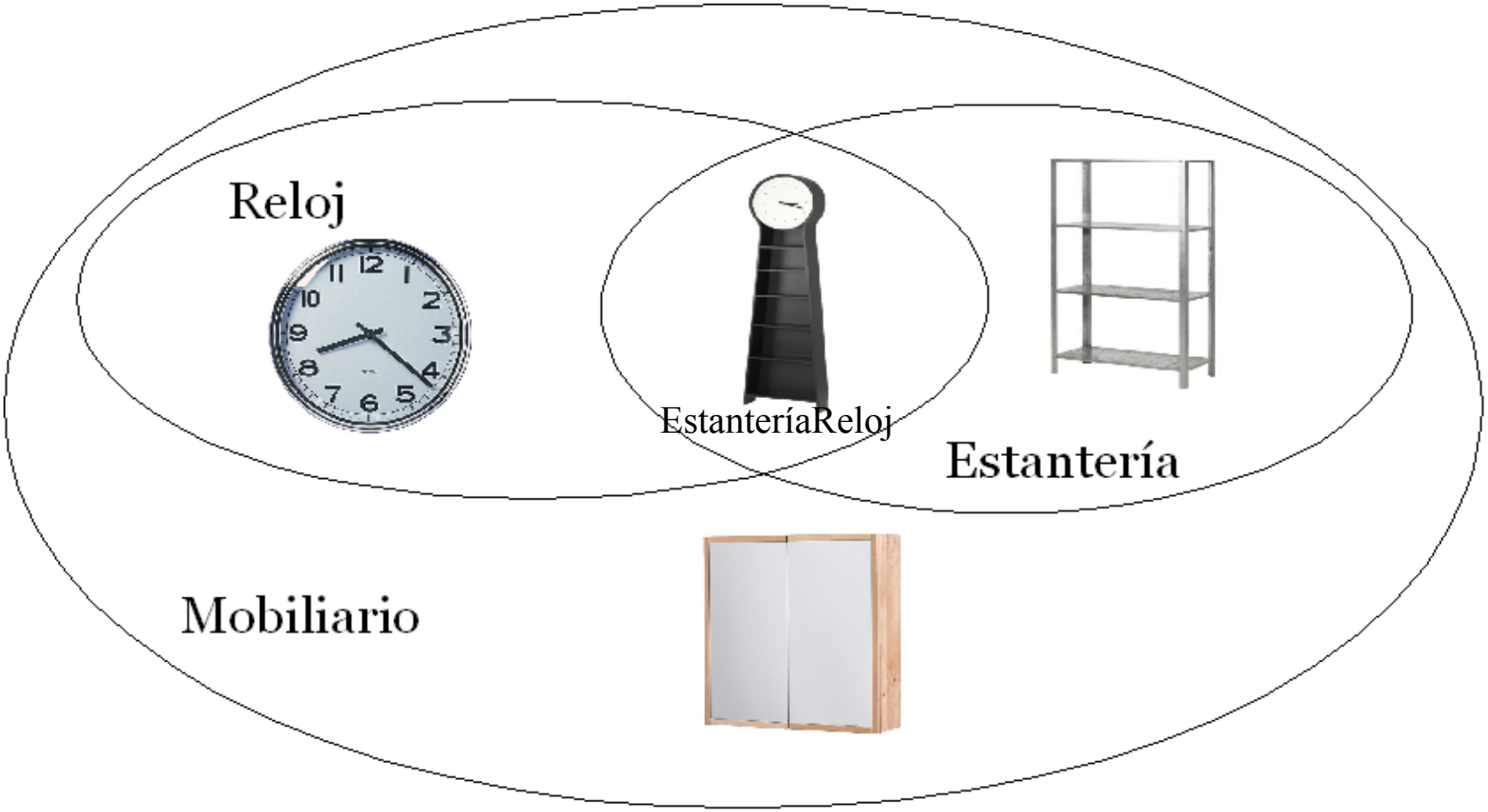
2. Conceptos básicos de POO: **Mensaje**

- **Petición** enviada a un objeto para que realice un método.

Máxima de POO: Los objetos colaboran entre sí enviándose mensajes para pedirse ayuda unos a otros.

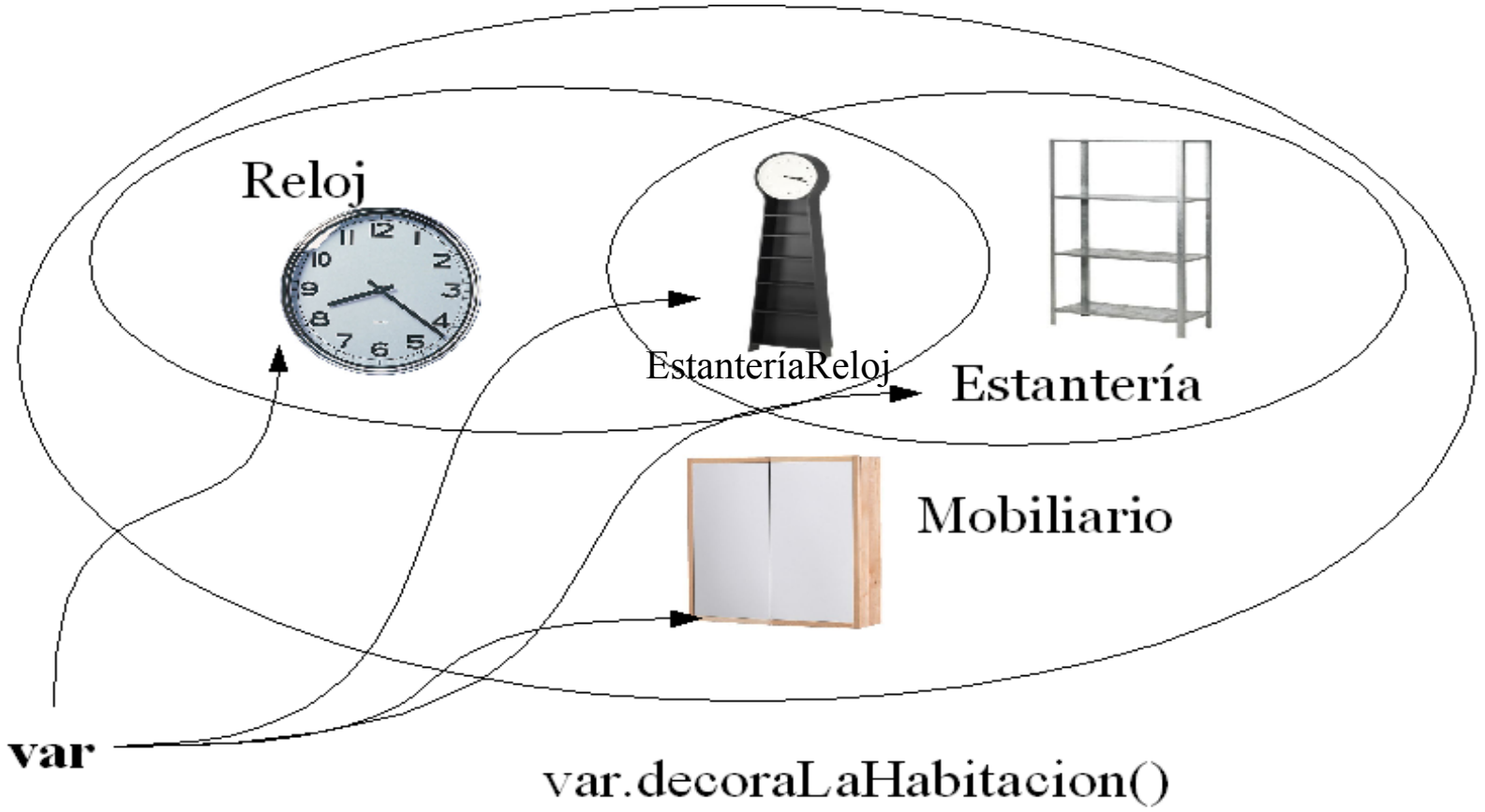
2. Conceptos básicos de POO: **Herencia**

- Relación jerárquica entre dos clases por la que una clase (subclase) hereda los métodos y propiedades de otra/s clase/s (superclase/s).



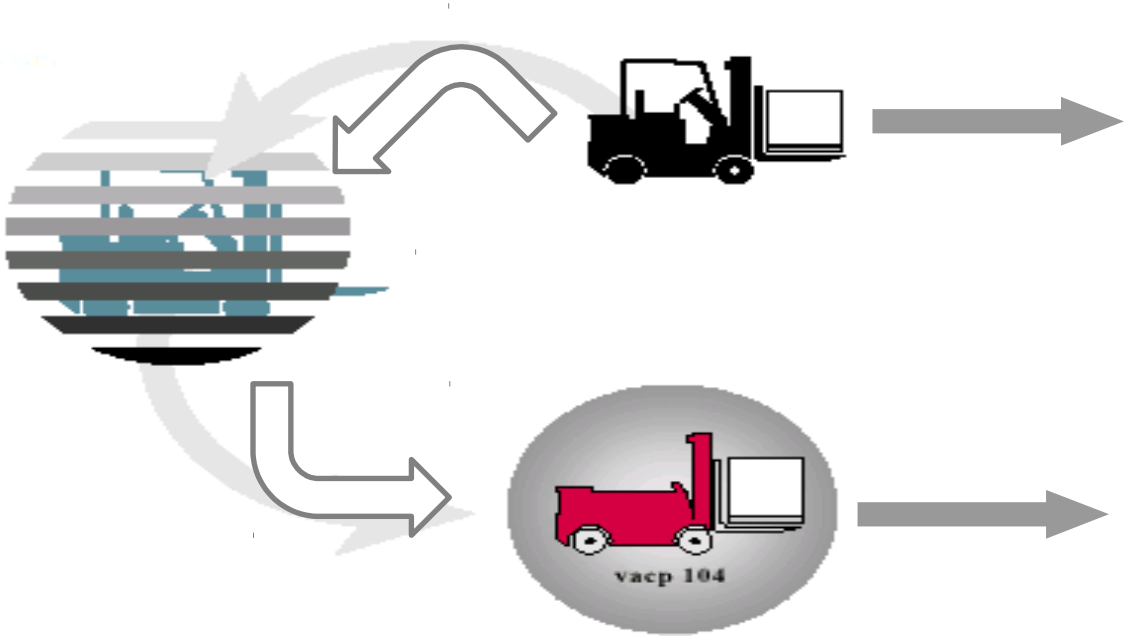
2. Conceptos básicos de POO: **Polimorfismo**

- Capacidad de una variable (variable polimorfica) para referenciar a objetos de distinta naturaleza de forma que el objeto real no se conoce hasta tiempo de ejecución (ligadura dinámica).



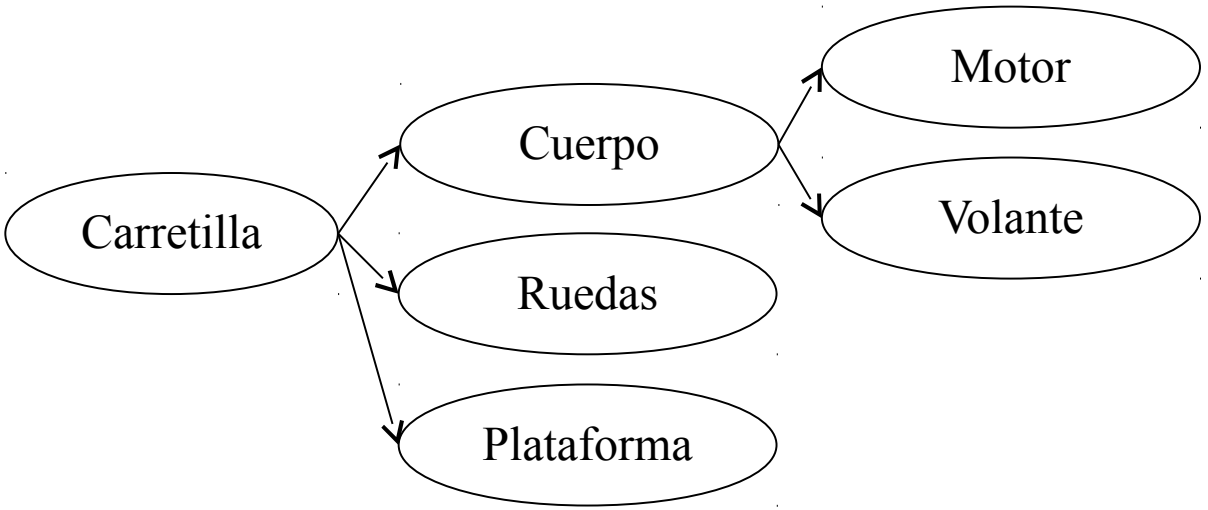
3. Conceptos básicos de DS adaptados a OO: **Abstracción**

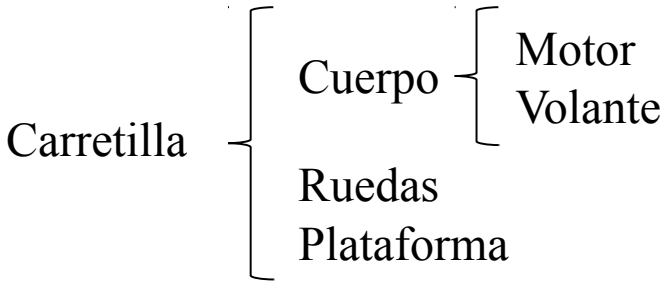
- Extraer las **propiedades relevantes** del dominio de un problema ignorando los detalles concretos.
- Los **criterios de abstracción** en DOO se obtienen observando características que presentan los objetos en su estado y comportamiento.



3. Conceptos básicos de DS adaptados a OO: **Modelo**

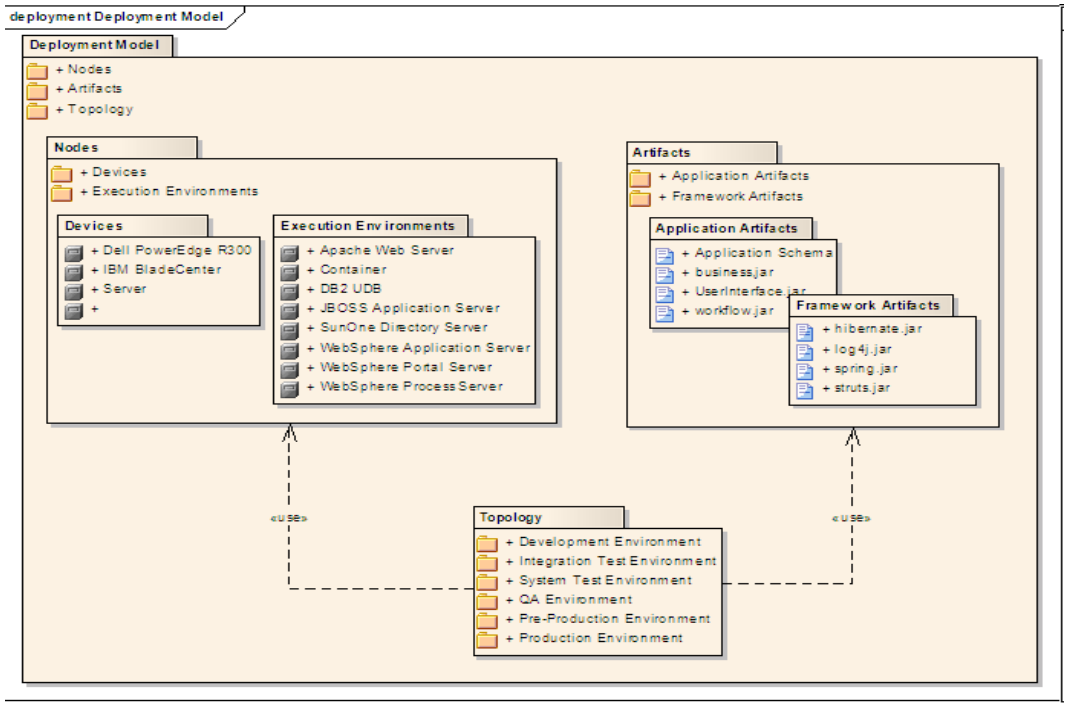
- Representación de la realidad observada mediante un determinado lenguaje, ya sea textual o gráfico.





3. Conceptos básicos de DS adaptados a OO: **Modularidad**

- Un sistema software está formado por piezas (denominadas módulos) que deben encajar perfectamente.
- En el diseño orientado a objetos, la pieza más pequeña es la **clase**. A su vez, las clases se agrupan en **paquetes,** y así sucesivamente hasta llegar al sistema completo.



Propiedad de la pieza: máxima cohesión y mínimo acoplamiento

3. Conceptos básicos de DS adaptados a OO: **Encapsulamiento**

- Capacidad de **agrupar** diversos elementos en uno solo cuyos límites están bien definidos.
- El elemento básico de encapsulación en la OO es la **clase**, que encapsula datos y operaciones.
- El encapsulamiento permite ocultar todo lo que se desee que no sea visible desde el exterior (**ocultamiento de información**).

3. Conceptos básicos de DS adaptados a OO: **Reutilización**

Uso de métodos, técnicas y heurísticas de diseño para:

- Facilitar el uso de una pieza software en futuras aplicaciones.
- Incorporar software ya desarrollado en la aplicación en curso.

Formas de conseguir piezas con alto grado de reutilización:

- Uso de la herencia.
- Mantener la independencia entre los distintos módulos del sistema (bajo acoplamiento, alta cohesión).
- Especificar de forma clara y completa la interfaz de cada módulo.
- Hacer una búsqueda de módulos ya existentes que puedan ser incorporados, antes de proceder a implementarlos desde cero.
- Usar patrones (guías) de diseño OO (así se reutiliza la solución dada a un determinado problema recurrente).

Lección 1.2

Lenguajes y técnicas de diseño orientado a objetos

Objetivos de aprendizaje

- Identificar los distintos tipos de lenguajes de programación orientados a objetos disponibles actualmente.
- Comprender la diferencia entre un lenguaje de programación orientado a objetos puro y uno mixto.
- Conocer los principales lenguajes de programación orientados a objetos.
- Identificar UML como un lenguaje de modelado que puede ser utilizado en distintos métodos de diseño.
- Conocer los principales diagramas de diseño disponibles en UML.

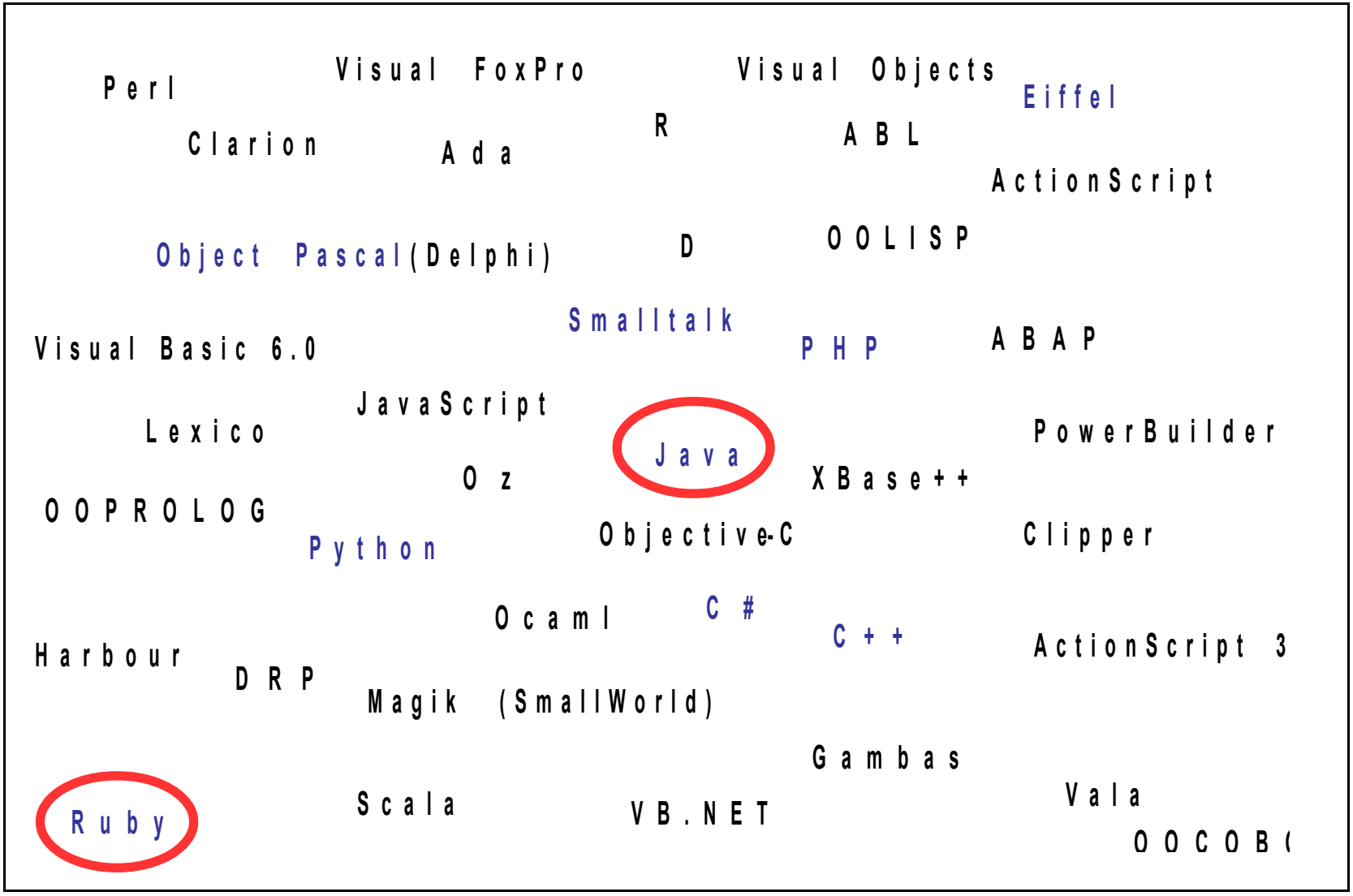
Contenidos

1. Clasificación de los lenguajes OO.

2. Introducción al Lenguaje Unificado de Modelado.

1. Clasificación de los lenguajes OO

Existen muchos lenguajes de programación OO:



1. Clasificación de los lenguajes OO

TIOBE Programming Community Index (popularidad)

Aug 2016	Aug 2015	Change	Programming Language	Ratings	Change
1	1		Java	19.010%	-0.26%
2	2		C	11.303%	-3.43%
3	3		C++	5.800%	-1.94%
4	4		C#	4.907%	+0.07%
5	5		Python	4.404%	+0.34%
6	7	▲	PHP	3.173%	+0.44%
7	9	▲	JavaScript	2.705%	+0.54%
8	8		Visual Basic .NET	2.518%	-0.19%
9	10	▲	Perl	2.511%	+0.39%
10	12	▲	Assembly language	2.364%	+0.60%
11	14	▲	Delphi/Object Pascal	2.278%	+0.87%
12	13	▲	Ruby	2.278%	+0.86%
13	11	▼	Visual Basic	2.046%	+0.26%
14	17	▲	Swift	1.983%	+0.80%

1. Clasificación de los lenguajes OO

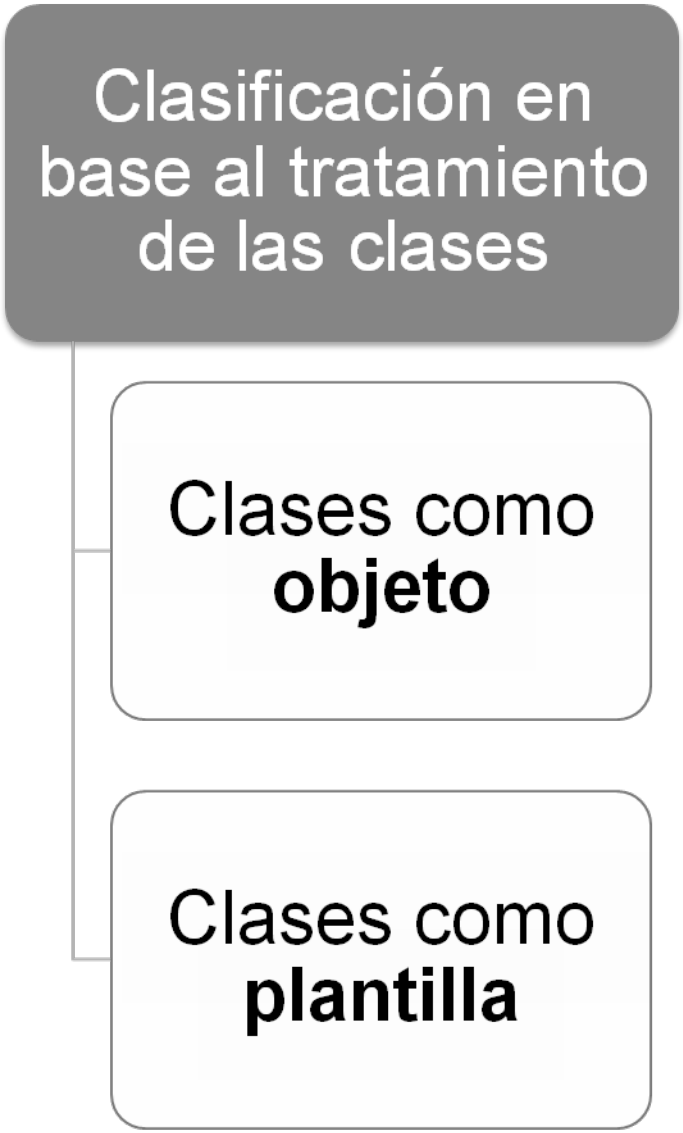
Criterios de comparación en base a conceptos de OO:

- Especificidad en cuanto a OO (OO, multiparadigma)
- Implementación de las clases (objetos, plantilla)
- Tipo de herencia (simple, múltiple)
- Tipo de ligadura (estática, dinámica, mixta)
- Rigurosidad en ocultamiento de la información (muy riguroso, poco, medio)

Otros criterios:

- Plataforma (linux, windows, iOs, ..., multiplataforma)
- Compilación o interpretación
- Gestión de liberación de memoria (automática, manual)
- Eficiencia
- Facilidad de uso
- Manejo de excepciones
- Reflexión

1. Clasificación de los lenguajes OO



- Todas las clases son objetos (metaclase).
- Pueden recibir y enviar mensajes.

→ Smalltalk, Python (> v2.0, clases de nuevo estilo) y Ruby

- La clase es un definición textual estática (nombre, declaración de las variables e instancia, declaración del protocolo, implementación de los métodos).

→ Java, C++, Eiffel, Python (clases de estilo clásico)

1. Clasificación de los lenguajes OO

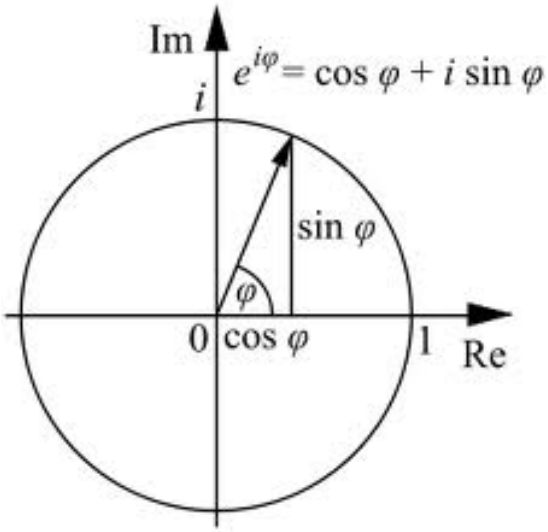
Comparativa de los dos lenguajes más usados en la asignatura

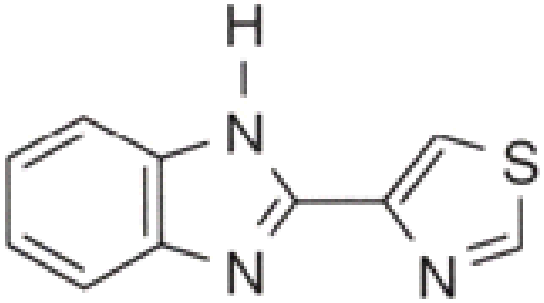
- Especificidad: OO
- Implementación de las clases: plantilla
- Herencia simple
- Ligadura dinámica
- Rigurosidad en ocultamiento de la información: medio (4 niveles de protección)
- Multiplataforma
- Compilado e interpretado
- Gestión de liberación de memoria: automática
- Eficiente y fácil de usar
- Manejo de excepciones y Reflexión (introspección)

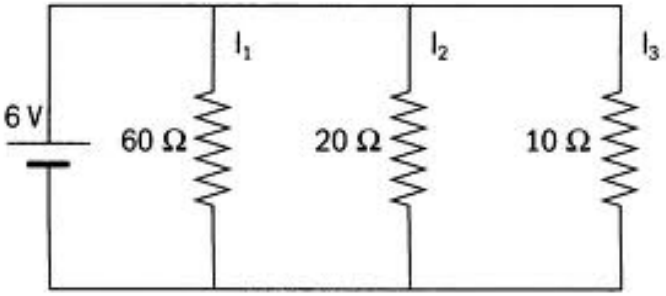
- Especificidad: multiparadigma (procedural, OO, funcional)
- Implementación de las clases: objeto
- Herencia simple
- Ligadura dinámica
- Rigurosidad en ocultamiento de la información: medio (3 niveles de protección)
- Multiplataforma
- Interpretado
- Gestión de liberación de memoria: automática
- Eficiente y fácil de usar
- Manejo de excepciones y Reflexión (modificación)

2. Introducción al lenguaje unificado de modelado: **UML**

Muchas disciplinas científicas manejan lenguajes que permiten expresarse y entenderse eliminando en parte la ambigüedad del lenguaje natural…







2. Introducción al lenguaje unificado de modelado: **UML**

UML es un lenguaje de diseño que permite:

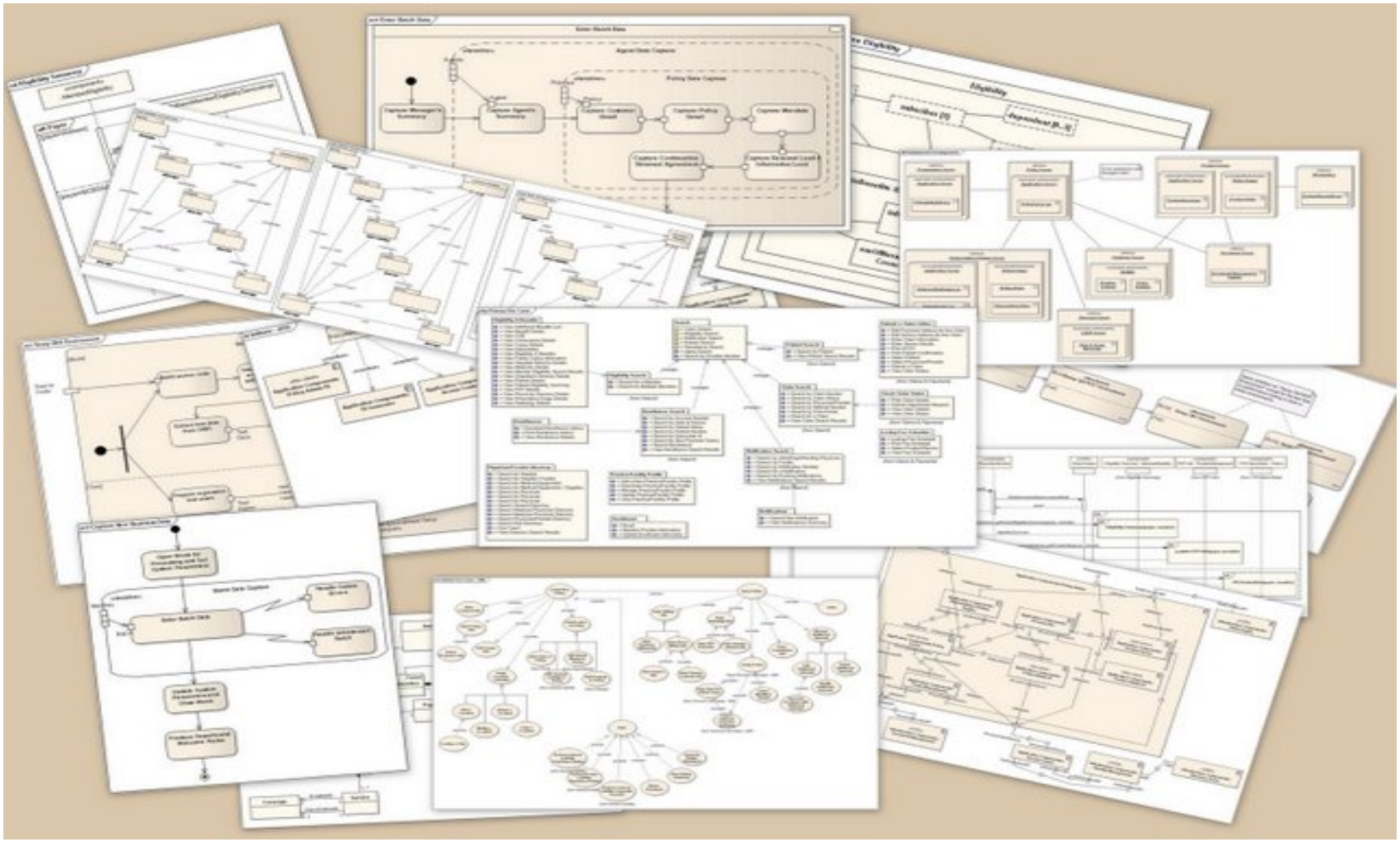
- **Visualizar** gráficamente un sistema software de manera que diversos desarrolladores lo puedan entender.
- **Especificar** mediante modelos cuáles son las características de un sistema antes de su **construcción** a partir de esos modelos.
- **Documentar** el sistema desarrollado, ya que los modelos ayudarán en su futura revisión.

Al tratarse de un lenguaje y no de una metodología, sus diagramas se pueden emplear con cualquier metodología.

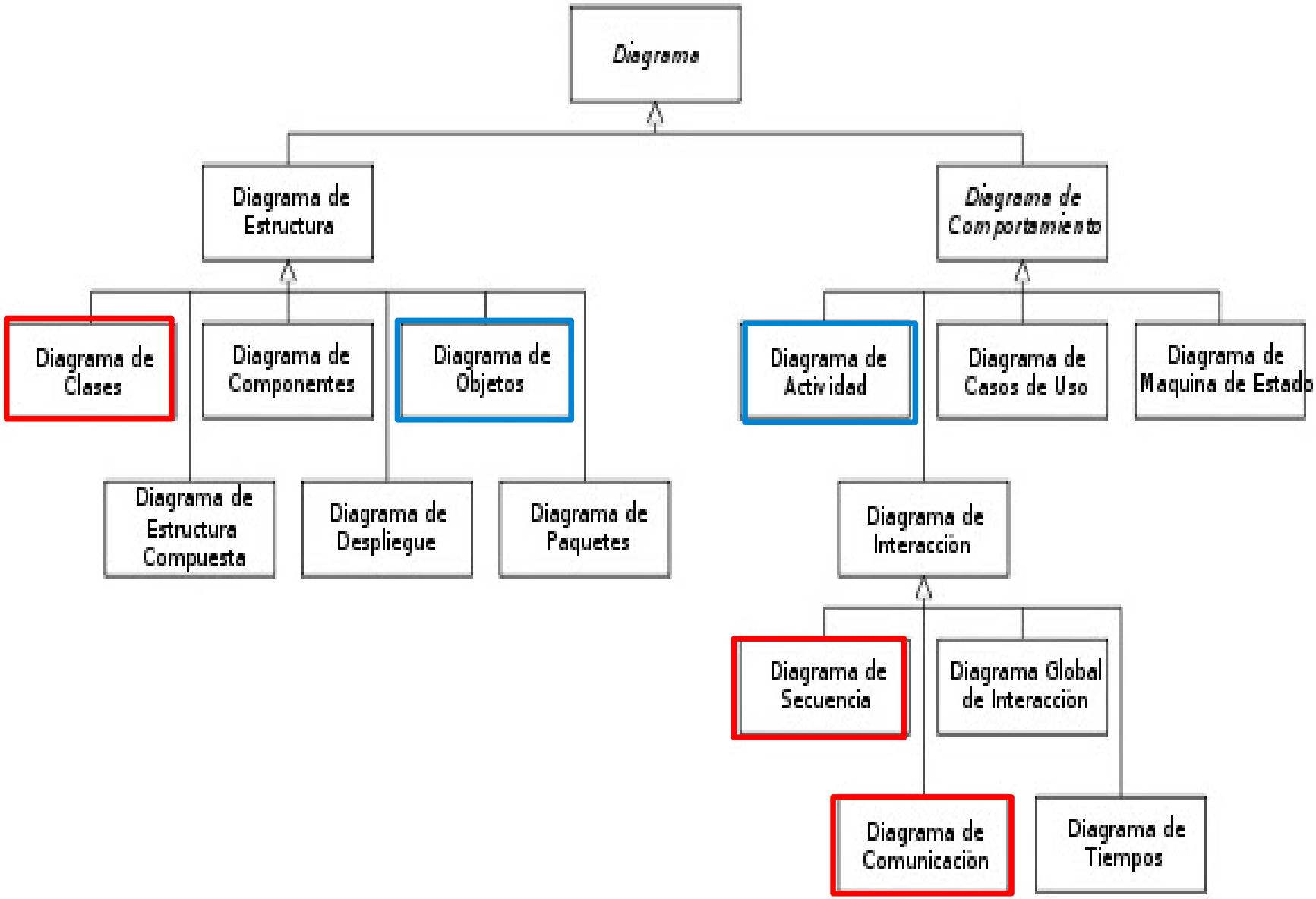
Es de diseño, y por tanto independiente del lenguaje de programación.

2. Introducción al lenguaje unificado de modelado: **UML**

- UML tiene muchos más diagramas…

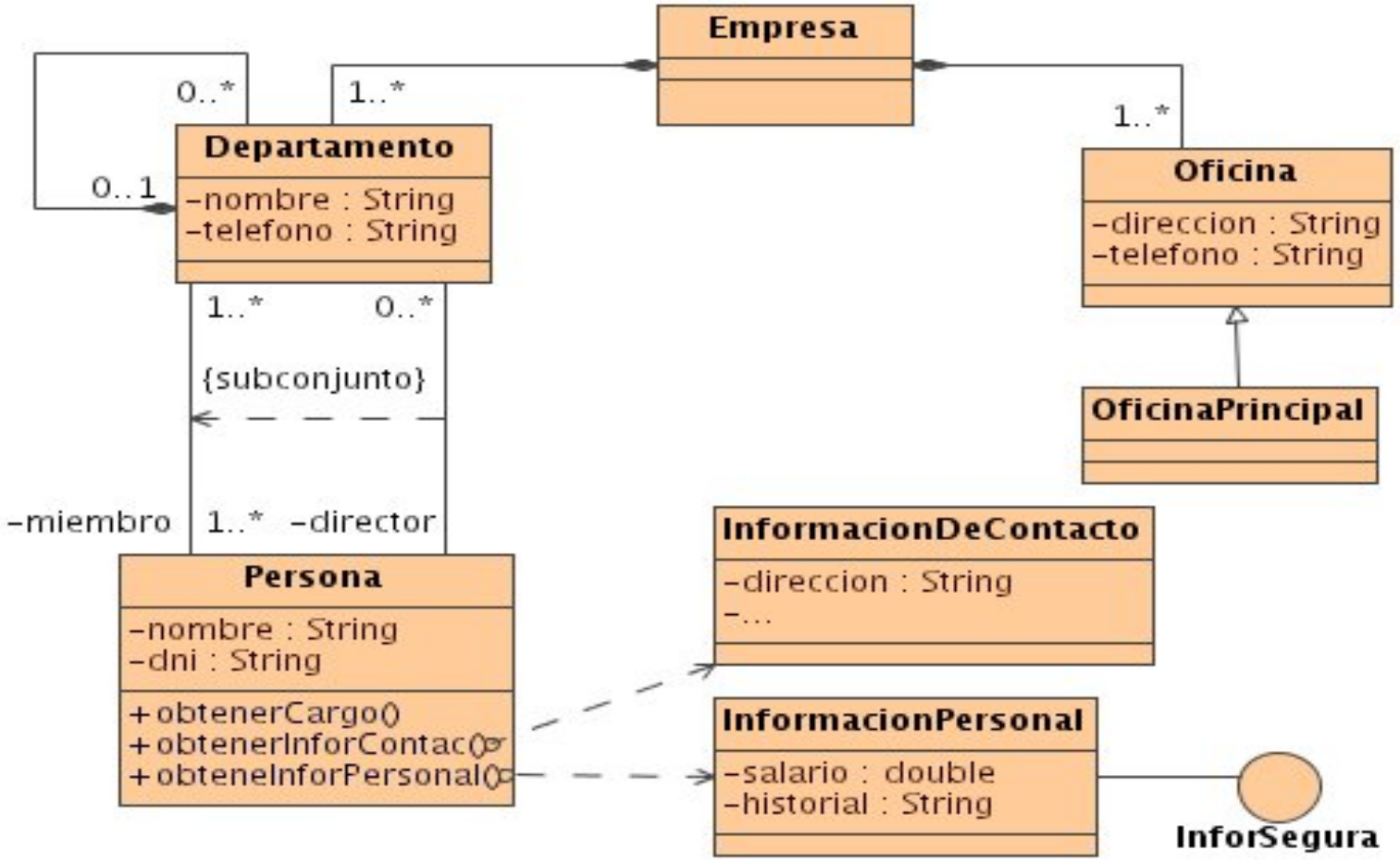


2. Introducción al lenguaje unificado de modelado: **UML**



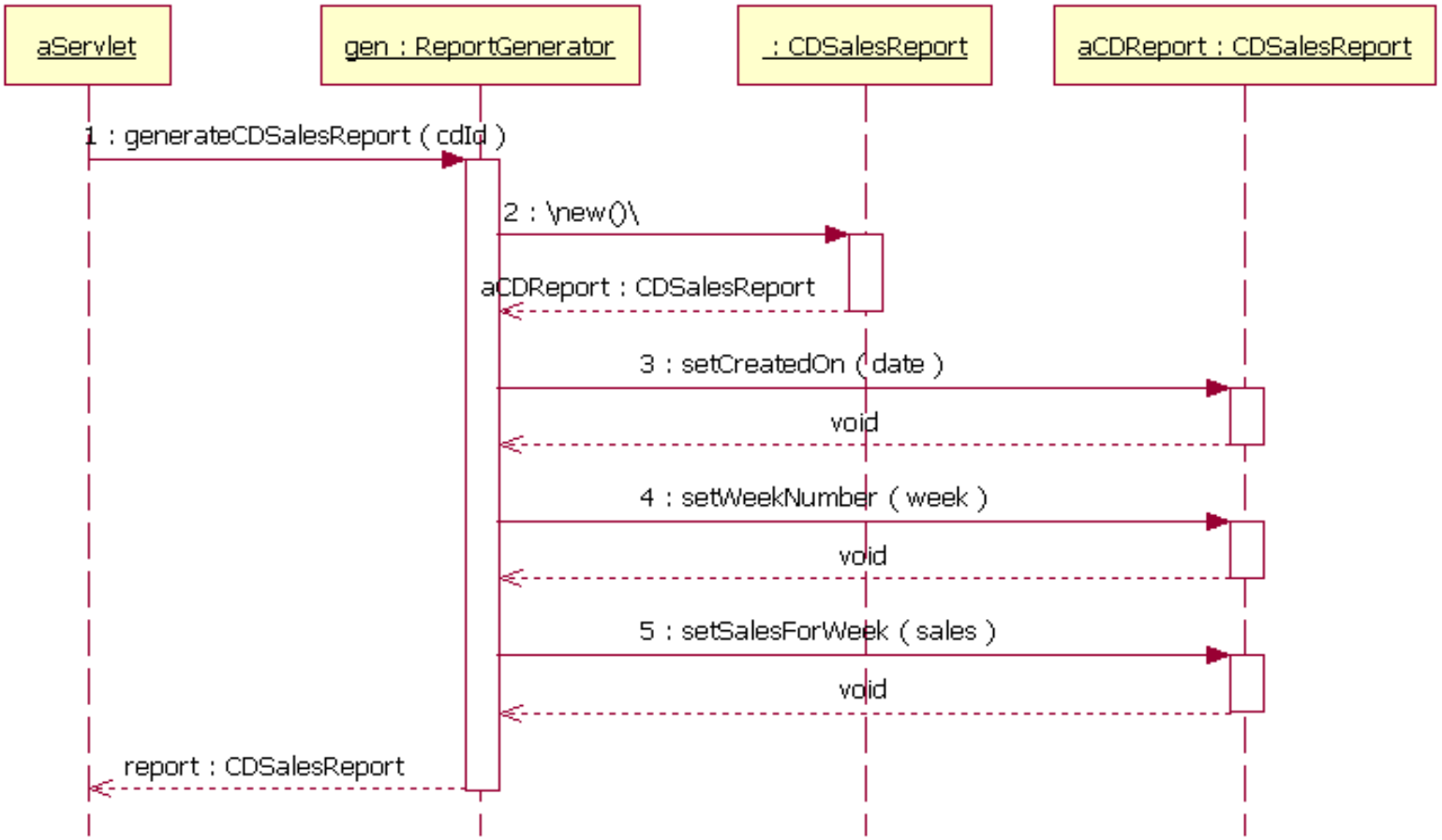
2. Introducción al lenguaje unificado de modelado: **UML**

- **Diagrama de clases**: muestra las clases (nodos) y sus relaciones (arcos).



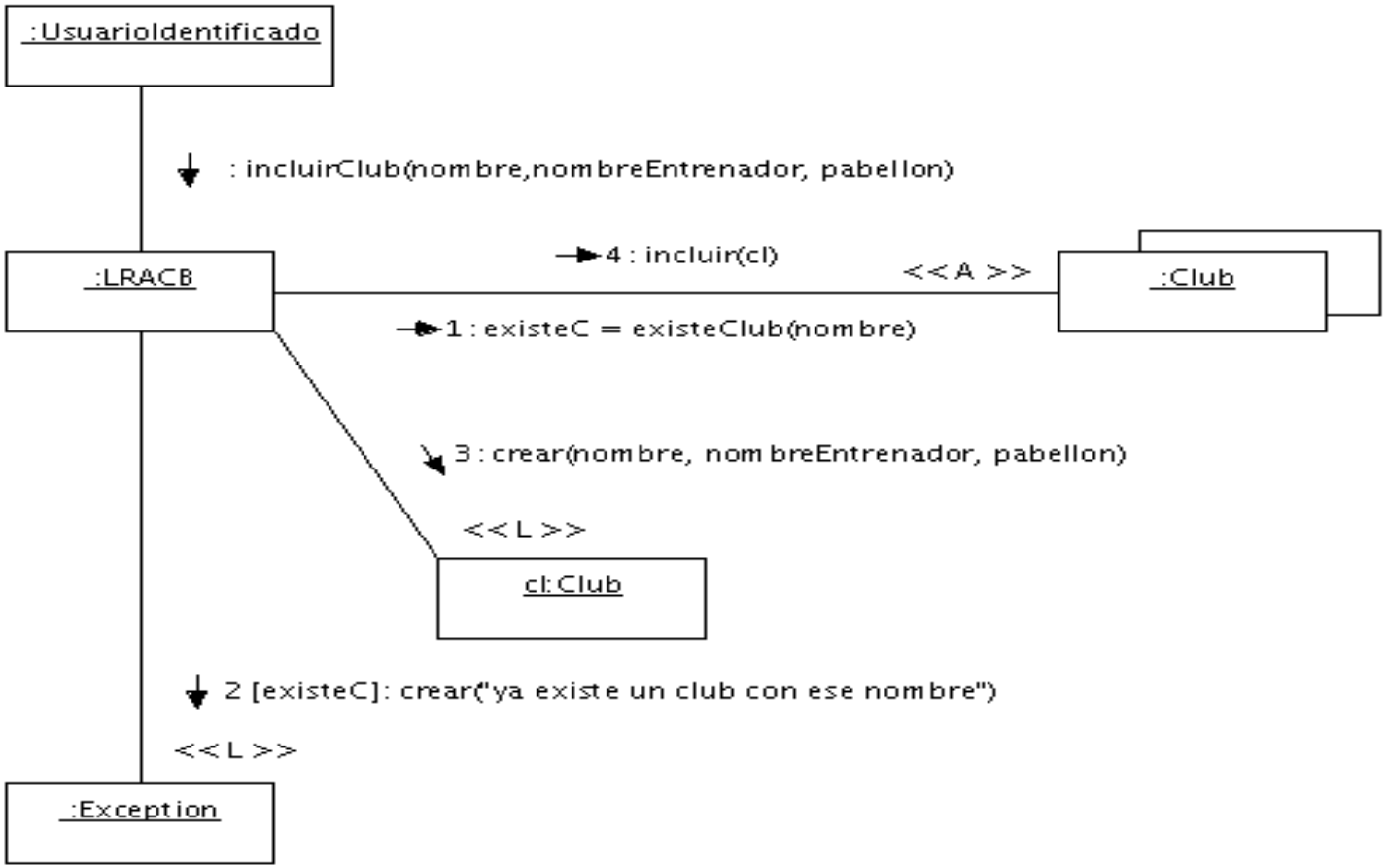
2. Introducción al lenguaje unificado de modelado: **UML**

- **Diagrama de secuencia:** muestra la interacción de un conjunto de objetos para llevar a cabo una operación (método).



2. Introducción al lenguaje unificado de modelado: **UML**

- **Diagrama de comunicación**: semánticamente equivalente a los diagramas de secuencia, pero con una notación distinta.



Tema 1

Introducción a la Orientación a Objetos