UNIVERSIDAD POLITÉCNICA DE MADRID
ESCUELA UNIVERSITARIA DE INFORMÁTICA
LENGUAJES, PROYECTOS Y SISTEMAS INFORMÁTICOS
Luis Fernández Muñoz
setillo@eui.upm.es

Programación Orientada a Objetos en Java

Tema 1. INTRODUCCIÓN

SINTESIS DE LA PROGRAMACIÓN ORIENTADA A OBJETOS:

nuevo equilibrio en condiciones de igualdad de importancia entre procesos y datos con un enfoque más antropomórfico

ATENCIÓN:

"X es bueno; Orientado a Objetos es bueno; Ergo, X es Orientado a Objetos". [Stroupstrup, 88]

"Mi conjetura es que la orientación a objetos será en los 80 lo que la programación estructurada en los 70. Todo el mundo estará a favor suyo. Cada productor prometerá que sus productos lo soportan. Cada director pagará con la boca pequeña el servirlo. Cada programador lo practicará. Y nadie sabrá exactamente lo que es". [Rentsch, 82]

ÍNDICE

1. Bases de la Programación (I)

Abstracción: proceso mental de extracción de las características esenciales de algo, ignorando los detalles superfluos

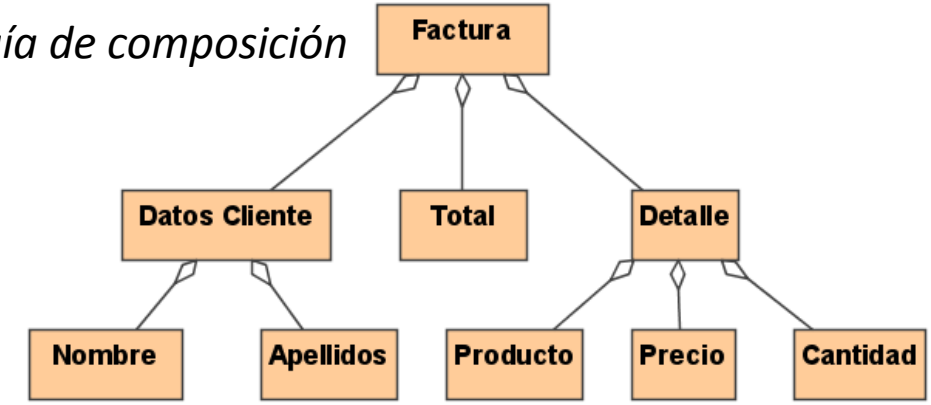
- **Encapsulación**: proceso por el que se ocultan los detalles del soporte de las características de una abstracción
- **Modularización**: proceso de descomposición de un sistema en un conjunto de módulos ('piezas') poco acoplados (independientes) y cohesivos (con significado propio)

> El acoplamiento "[...] es la medida de fuerza de la asociación establecida por una conexión ente un módulo -elemento- y otro. El acoplamiento fuerte complica un sistema porque los módulos son más difíciles de comprender, cambiar o corregir por sí mismos si están muy interrelacionados con otros módulos. […] La cohesión mide el grado de conectividad entre los elementos de un solo módulo." [Booch, 96]

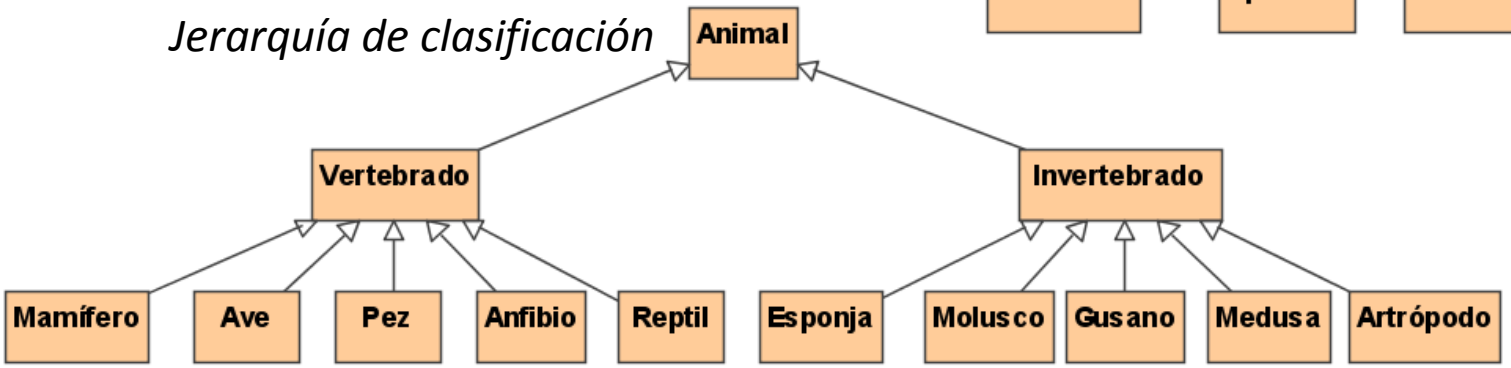
1. Bases de la Programación (II)

Jerarquización: proceso de estructuración por el que se produce una organización (jerarquía) de un conjunto de elementos en grados o niveles de responsabilidad, de incumbencia o de composición, entre otros.

Jerarquía de composición



Jerarquía de clasificación



2. Evolución de los Lenguajes de Programación (I)

BASES	CÓDIGO MÁQUINA	ENSAMBLADOR	PROGRAMACIÓN DE ALTO NIVEL
Abstracción	{0,1}	Identificadores	Subprogramas
Encapsulación	Nula	Nula	Reglas de Ámbito
Modularización	Nula	Macros	Subprogramas
Jerarquización	Nula	Nula	Expresiones, Registros y Subprogramas

2. Evolución de los Lenguajes de Programación (II)

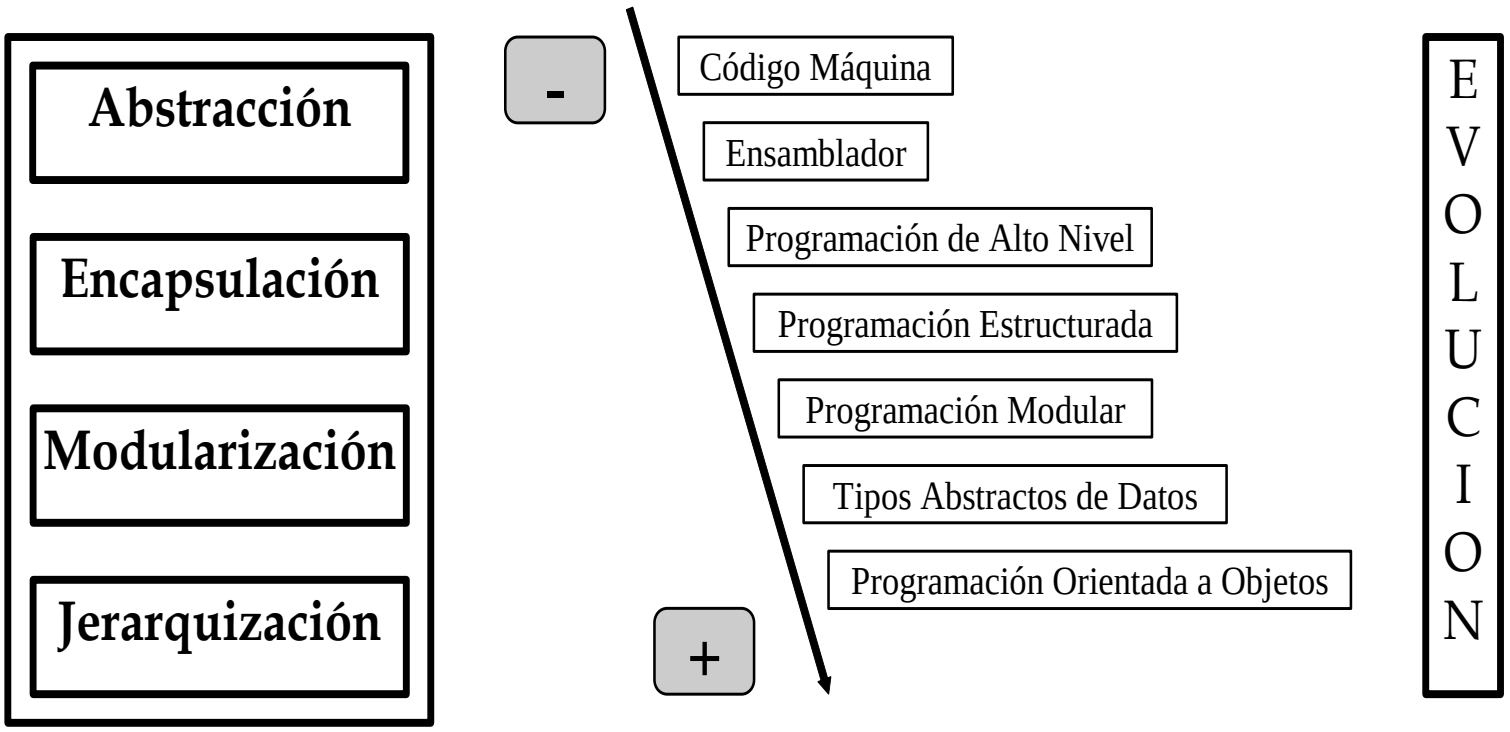
BASES	PROGRAMACIÓN ESTRUCTURADA	PROGRAMACIÓN MODULAR
Abstracción	Estructuras de Control de Flujo de Ejecución	Espacios de nombres
Encapsulación		Privacidad en Módulos
Modularización		Módulos
Jerarquización		Jerarquías de dependencia cliente/servidor entre los módulos

2. Evolución de los Lenguajes de Programación (III)

BASES	TIPOS ABSTRACTOS DE DATOS	PROGRAMACIÓN ORIENTADA A OBJETOS
Abstracción	Vista Pública	Herencia y Polimorfismo
Encapsulación	Vista Privada	=
Modularización	TAD's	Clases
Jerarquización	Jerarquías de composición y dependencia	Jerarquías de clasificación

OO = TAD's + Herencia

2. Evolución de los Lenguajes de Programación (IV)



2. Evolución de los Lenguajes de Programación (V)

LEYES DE LHEMAN Y BELADY:

***Ley del Cambio Continuo**: "Un programa que se usa en un ámbito del mundo real, necesariamente debe cambiar o convertirse cada vez en menos útil".*

***Ley de la Complejidad Creciente**: "Debido a que los programas cambian por evolución, su estructura se convierte en más compleja a menos que se hagan esfuerzos activos para evitar este fenómeno".*

2. Evolución de los Lenguajes de Programación (VI)

OBJETIVO DE LA POO (II):

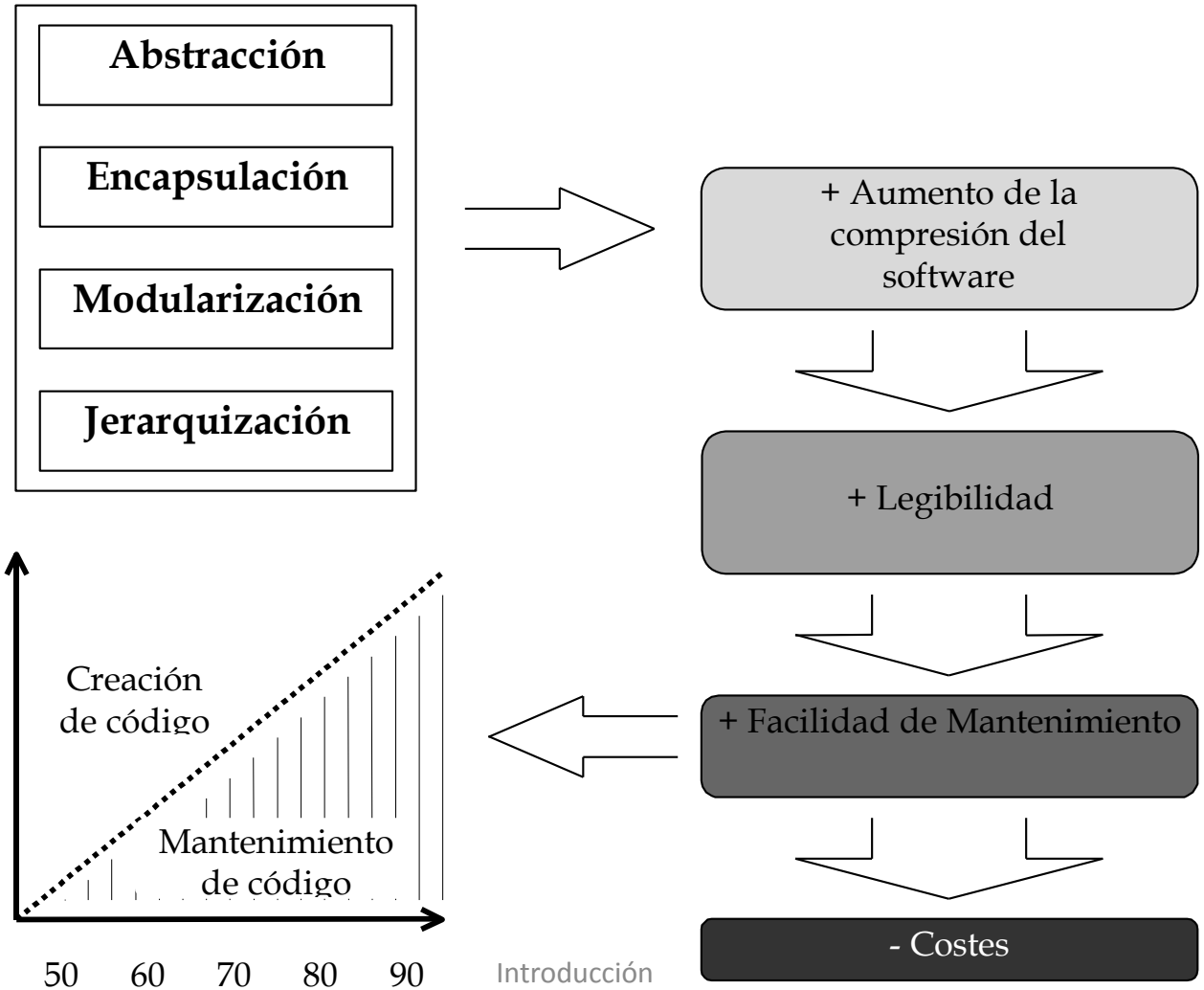
Implicaciones:

• El incremento de abstracción, encapsulación, modularidad y jerarquización aumentan la compresión, la escalabilidad y la flexibilidad del software

• El incremento de la comprensión , la escalabilidad y la flexibilidad del software reduce los costes del mantenimiento del software (correctivo, adaptativo y perfectivo)

• La reducción de los costes del mantenimiento del software reduce drásticamente los costes del desarrollo del software

2. Evolución de los Lenguajes de Programación (VII)



3. Elementos de la Programación Orientada a Objetos (I)

Clase: descripción de los datos y de las operaciones que describen el comportamiento de un cierto conjunto de elementos homogéneos.

Ej. Clase Intervalo

- *datos: extremos inferior y superior;*
- *operaciones: intersección, longitud, desplazar, …*

Objeto: ejemplar concreto (*instancia*) de una clase, que responde al comportamiento definido por las operaciones de la clase a la que pertenece, adecuándose al estado de sus datos particulares.

Ej. Objetos de la clase Intervalo :

- *constantes: (8,10), (-100,100),…*
- *variables: misHorasDeTrabajo, miPresiónArterial, …*

3. Elementos de la Programación Orientada a Objetos (II)

Mensaje: invocación de una operación sobre un objeto. Un objeto es el agente activo que lanza el mensaje y otro objeto es el agente pasivo que recibe el mensaje. El objeto receptor del mensaje debe contemplar dicha operación entre las definidas en su clase.

Ej. Mensajes a objetos de la Clase Intervalo

- *(8,10).longitud = 2; (-100,100).desplazar(3) => (-97,103),…*
- *misHorasDeTrabajo.interseccion(tusHorasDeTrabajo), …*

Método: definición de una operación de una clase.

Ej. Métodos de la Clase Intervalo

- *longitud: extremo superior menos extremo inferior;*
- *desplazar: acumular cantidad a ambos extremos;*

3. Elementos de la Programación Orientada a Objetos (III)

Atributo: cada uno de los datos de una clase, y por tanto, presente en todos los objetos de esa clase.

Ej. Atributos de la clase Intervalo

- *extremos inferior y superior*

Estado: conjunto de los valores de los atributos que tiene un objeto, por pertenecer a una clase, en un instante dado.

Ej. Estados de objetos de la clase Intervalo

- *8 en el extremo inferior y 14 en el extremo superior de presiónArterial;*
- *9 en el extremo inferior y 18 en el extremo superior de misHorasDeTrabajo;*

3. Elementos de la Programación Orientada a Objetos (IV)

RELACIÓN DE LOS ELEMENTOS DE LA POO:

*Una **clase** es la definición de los **atributos** y **métodos** que describen el comportamiento de un cierto conjunto de **objetos** homogéneos.*

*Un **objeto** es un ejemplar concreto de una **clase** que responde a los **mensajes** correspondientes a los **métodos** de ésta, adecuándose al **estado** de sus **atributos.***

3. Elementos de la Programación Orientada a Objetos (V)

Las clases asumen el principio de encapsulación: cuando se describe una clase, se debe describir tanto su vista pública o interfaz como su vista privada o implantación.

*La **vista pública o interfaz** describe qué operaciones responden los objetos de esta clase, o sea, su comportamiento.*

*La **vista privada o implantación** describe las estructuras de datos de la clase y cómo manipulan las operaciones los datos. De esta forma, se conjugan la abstracción inherente en la clase con la encapsulación de sus datos y de su forma de operar*

3. Elementos de la Programación Orientada a Objetos (VI)

Las clases que conjugan de forma equilibrada atributos (datos) y métodos (operaciones) son el único bloque de construcción de programas: módulos.

Esta modularidad exige que:

- *los datos y operaciones de una clase son elementos estrechamente relacionados favoreciendo que "una clase se entiende por sí misma" –cohesión-.*
- *las clases se relacionan -acoplamiento- al colaborar en jerarquías de composición, clasificación, … para constituir los programas*

3. Elementos de la Programación Orientada a Objetos (VII)

Herencia: transmisión de atributos y métodos de una clase a otra clase.

Ej. A partir de la Clase Intervalo

- *IntervaloCerradoCerrado [x,x], IntervaloAbiertoCerrado (x,x],…*

Polimorfismo: enlace dinámico de expresiones a clases y/o de mensajes a métodos.

Ej. Objetos intercambiables de las clases Intervalo, IntervaloCerradoCerrado, …

3. Elementos de la Programación Orientada a Objetos (VIII)

RELACIÓN DE LOS ELEMENTOS DE LA POO:

*La **clase** hija **hereda** los **atributos** y **métodos** de la **clase** padre y se especializa añadiendo y/o redefiniendo atributos y métodos.*

*En el **polimorfismo**, el **objeto** activo sólo necesita conocer qué **mensajes** puede aceptar el **objeto** pasivo, no qué **clase** de **objeto** cree que es y, por tanto, qué **método** ejecuta en cada instante.*

3. Elementos de la Programación Orientada a Objetos (IX)

COMPARATIVA DE LA POO vs PROGRAMACIÓN ESTRUCTURADA:

PROGRAMACIÓN ESTRUCTURADA	BASES DE LA PROGRAMACIÓN	POO
Registro + Funciones	Abstracción, Encapsulación y Modularidad	Clase
Variable de Tipo Registro	Abstracción, Encapsulación	Objeto
Función	Modularidad	Método
Llamada a Función	Abstracción	Mensaje
Campo de un Registro	Encapsulación y Modularidad	Atributo
Estado de una Variable Registro	Encapsulación	Estado
Registros de Campos Variables	Abstracción y Modularidad	Herencia
Punteros a Funciones	Abstracción y Encapsulación	Polimorfismo