



Tecnologías de Programación

Paradigma Orientado a Objetos

I – Fundamentos, Características y Conceptos Básicos



¿ Por que la evolución de los distintos paradigmas ?



Los distintos paradigmas y avance en el tiempo



Avance de la informática (hardware – software) problemas a los que se aplica la informática.



¿Un paradigma es mejor que otro?



Definición

La programación orientada a objetos es una metodología que descansa en el concepto de **objeto** para imponer la estructura modular de los programas.

Permite comprender el dominio del problema a resolver, al intentar construir un modelo del mundo real que envuelve nuestro sistema. Es decir, la representación de este mundo mediante la identificación de los objetos que constituyen el vocabulario del dominio del problema, su organización y la representación de sus responsabilidades.



Conceptos Básicos

- Objetos
- Clases
- Envío de mensajes



OBJETO

- Un objeto es una abstracción conceptual del mundo real que se puede traducir a un lenguaje computacional o de programación OO.
- También tiene un **estado**, que permite informar lo que éste representa y su comportamiento, es decir lo que él sabe hacer. Hablando en términos computacionales, la **identidad** del objeto se puede interpretar como la referencia. El estado del objeto es una lista de variables conocidas como sus **atributos**, cuyos valores representan el estado que caracteriza al objeto.



OBJETO II

- Los objetos también se conocen como **instancias**.

Los objetos encapsulan **datos** y **operaciones** (o métodos).

DATOS (Estado)

METODOS (Comportamiento)

OBJETO



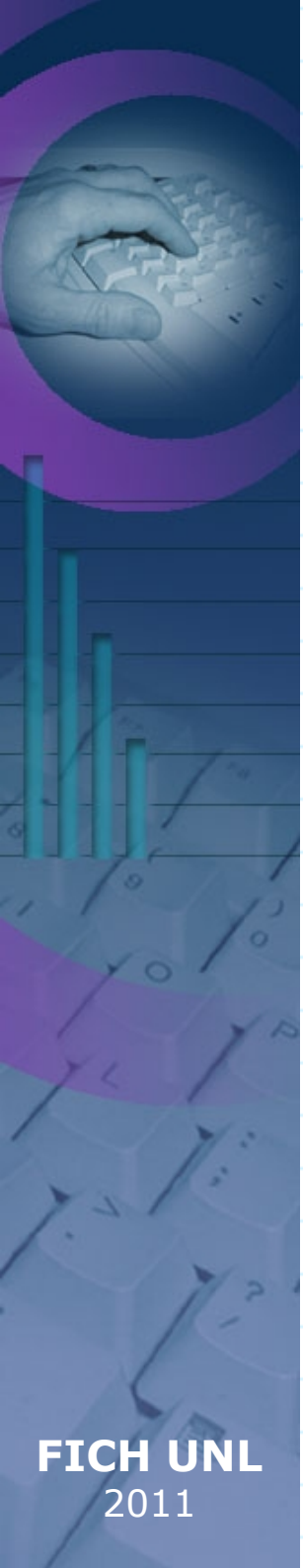
OBJETO - Estado

- Es una buena práctica de ingeniería encapsular el estado de un Objeto, ocultar la representación del estado del objeto a sus clientes externos.
- Ningún objeto puede modificar el estado de otro objeto.
- El estado interno de un objeto sólo puede modificarse mediante el envío de algún mensaje válido.



OBJETO - Comportamiento

- El comportamiento expresa de que forma el objeto actúa y reacciona en términos de su cambio de estado y solicitud de servicio.
- Representa las actividades visibles exteriormente.
- Los métodos asociados a un objeto implementan el comportamiento del objeto.



OBJETO - Identidad

- Es aquella propiedad del objeto la cual lo distingue de los otros objetos.
- Identidad <> estado

```
Persona oPersonaA = new Persona("Juan Perez");  
Persona oPersonaB = new Persona("Juan Perez");
```

```
oPersonaA.equals(oPersonaB) → FALSO
```



CLASE

- Una **clase** contiene la descripción de las características comunes de todos los **objetos** que pertenecen a ella:
 - la especificación del **comportamiento**
 - la definición de la **estructura interna**
 - la implementación de los **métodos**
- También se puede ver una clase como un molde, esquema o un patrón que define la forma de sus objetos.



CLASE - INSTANCIA

- Un **objeto** es una **instancia** de una **clase**.
- Tiene un **estado**, un **comportamiento** y una **identidad**.
- Una **instancia** es un **individuo** de la **clase**

CLASE	INSTANCIA
<ul style="list-style-type: none">• Define atributos• Define Métodos• Puede generar instancias	<ul style="list-style-type: none">• Tiene valores• Ejecuta Métodos• -

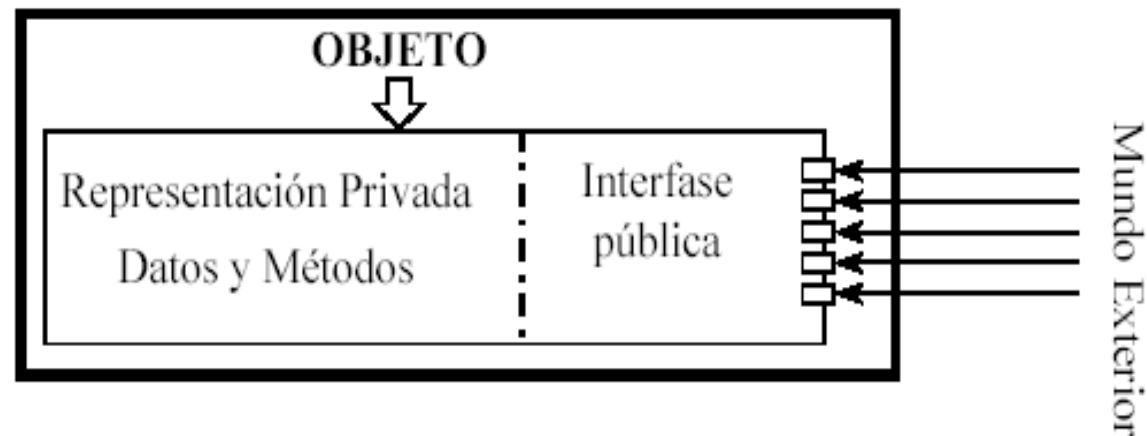


MENSAJES

- ¿Qué sucede cuando los objetos desean comunicarse entre sí?. Un objeto recibe un estímulo externo de solicitud de un servicio que se traduce en la invocación de un método de éste objeto.
- Al ejecutarse el método, el objeto puede solicitar servicios de otros objetos, enviándoles mensajes que implican a su vez la invocación de sus métodos, y dentro de estos, nuevamente invocar servicios de otros objetos y así sucesivamente. Este envío de mensajes en cascada representa el comportamiento dinámico del modelo de objetos.

MENSAJES II

- En el envío de mensajes interactúan dos objetos: el ***emisor*** del mensaje y el ***receptor***.
- La **interfase pública** es el **protocolo o conjunto de mensajes** a los que puede responder el **objeto** (Punto de Vista del Usuario)





MENSAJES III

- La **representación privada** son:
 - los datos necesarios para describir el **estado** y
 - la implementación de los métodos o procedimientos**(Punto de Vista del Implementador)**



Características

- Abstracción
- Encapsulamiento
- Modularidad
- Herencia - Jerarquía
- Polimorfismo



Abstracción

- Denota las características esenciales de un objeto las cuales lo distinguen de todos los otros tipos de objetos. Provee una definición conceptual relativa a la perspectiva del observador.
- Separaremos el comportamiento de la implementación
- Es más importante saber ***qué*** se hace en lugar de ***cómo*** se hace



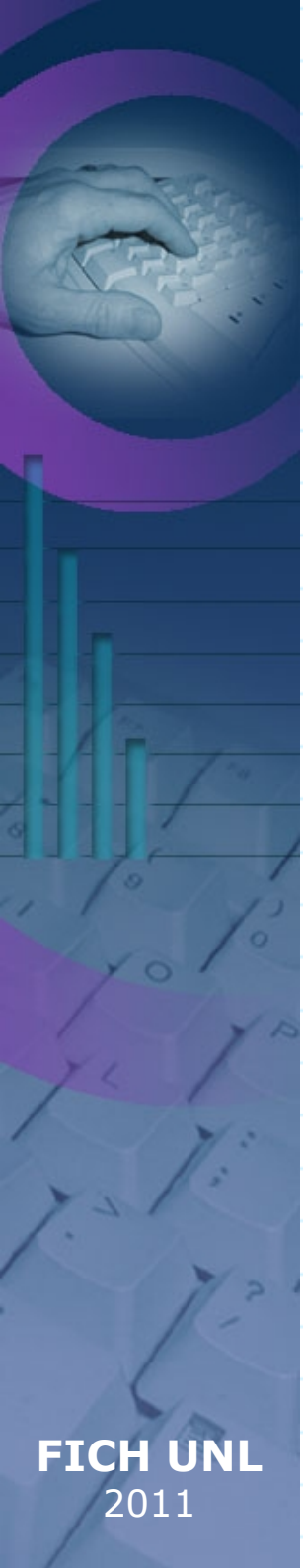
Encapsulamiento

- Es la propiedad que asegura que la información de un *módulo* esta *oculta* al mundo exterior.
- Es el proceso de *ocultar* todos los secretos de un módulo que no contribuyen a sus características esenciales.
- Es una **técnica de diseño** para descomponer sistemas en **módulos**
- Ninguna parte de un sistema complejo debe depender de los detalles internos de otra



Modularidad

- Consiste en separar el sistema en bloques poco ligados entre sí: módulos.
- Es una especie de encapsulamiento de más alto nivel. En Java (packages)
- Difícil pero muy importante en sistemas grandes. Suele aplicarse refinando el sistema en sucesivas iteraciones.



Herencia - Jerarquía

- Es una clasificación u ordenamiento de las abstracciones
- Hay dos jerarquías fundamentales:
 - Estructura de clases:
Jerarquía “*es un/a*”
Relaciones de **herencia**
 - Estructura de objetos:
Jerarquía “*parte de*”
Relaciones de **agregación**
Está implementada de manera genérica en la estructura de clases



Polimorfismo

- Permite enviar **el mismo mensaje a objetos diferentes** y que cada uno responda en la forma apropiada según el tipo de objeto que sea, ejecutando **su método**.
- El **polimorfismo** se extiende hacia abajo en la red de herencia porque las subclases heredan los protocolos, pero los métodos pueden estar especializados localmente. Usualmente requiere del empleo de clases abstractas.



Ventajas de POO

- Fomenta la reutilización y extensión del código.
- Permite crear sistemas más complejos.
- Relacionar el sistema al mundo real.
- Facilita la creación de programas visuales.
- Agiliza el desarrollo de software
- Facilita el trabajo en equipo
- Facilita el mantenimiento del software



Desventajas de POO

- Hay que ser muy cuidadosos en la creación de los objetos, ya que de ello dependerá el éxito de nuestro proyecto. Un error en estas primeras definiciones podría resultar catastrófico. Precisamente el secreto de esta técnica está en la correcta definición inicial de los objetos.