

# Tecnologías de Programación



## Paradigma Orientado a Objetos

### I – Fundamentos, Características y Conceptos Básicos



# 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
- Atributos



# 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 - Estado

```
public class Persona {  
  
    // atributos  
    private String nombre;  
    private Integer documento;  
  
    // constructor  
    public Persona(String nombre, Integer documento) {  
        this.nombre = nombre;  
        this.documento = documento;  
    }  
  
    // comportamiento  
    public String mostrar() {  
        return this.nombre + " - " + this.documento;  
    }  
}
```

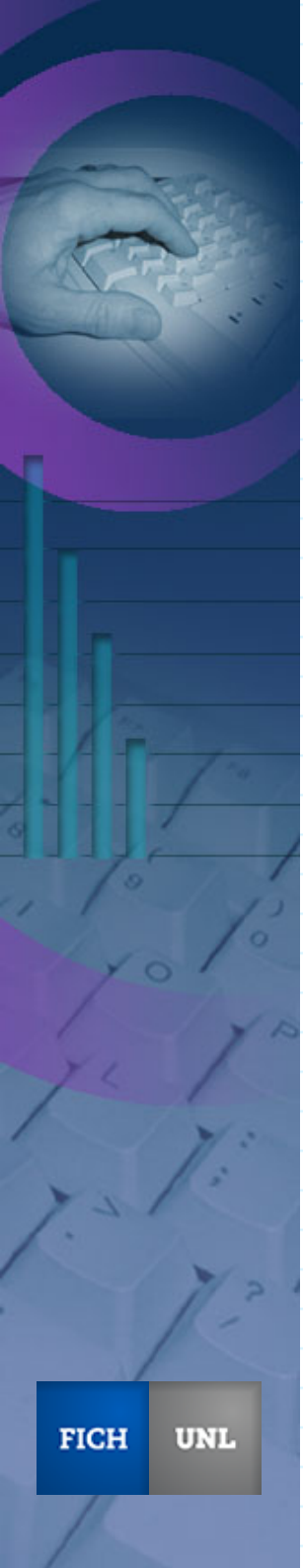




# OBJETO - Comportamiento

- 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

```
public static void main(String[] arg) {  
    Persona oPer1 = new Persona("Juan Perez", 12345678);  
    Persona oPer2 = new Persona("Juan Perez", 12345678);  
  
    if (oPer1.equals(oPer2)) {  
        System.out.println("Son iguales");  
    } else {  
        System.out.println("Son distintos");  
    }  
}
```



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

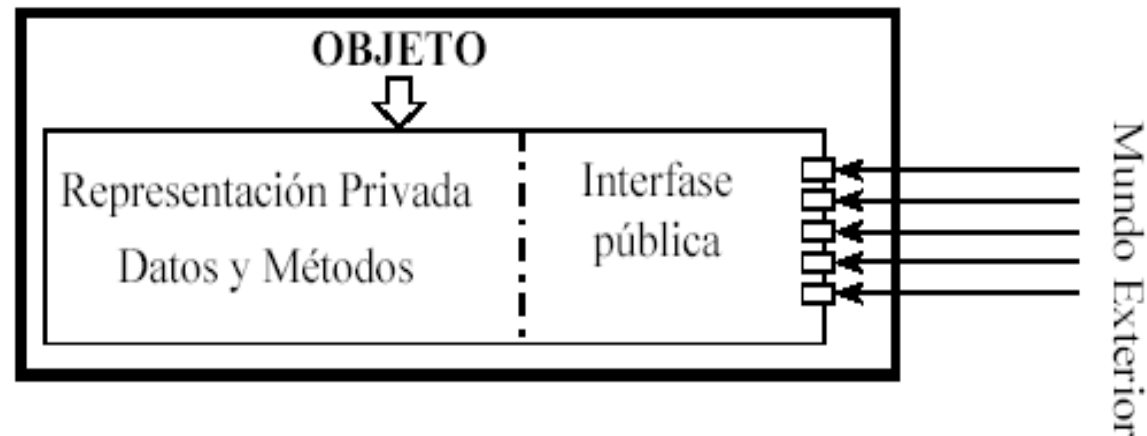


# MENSAJES

- ¿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 **interface 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)**



# Atributos

## ● De INSTANCIA / CLASE

Los atributos de una clase son definidos según esta sintaxis:

`[modifVisibilidad] [modifAtributo] tipo nombreVariable = [valorInicial];`

**modifVisibilidad** indica desde que parte del código se puede acceder a la variable:

- **public:** indica que es un atributo accesible a través de una instancia del objeto.
- **private:** indica que a través de una instancia no es accesible el atributo. Al heredar el atributo se convierte en inaccesible.
- **protected:** indica que a través de una instancia no es accesible el atributo. Al heredar si se puede usar desde la clase derivada.
- **Sin especificar:** indica visibilidad de paquete, se puede acceder a través de una instancia, pero sólo desde clases que se encuentren en el mismo paquete.





# Atributos II

**modifAtributos** son características específicas del atributo, son:

- **static:** El atributo pertenece a la clase, no a los objetos creados a partir de ella. Atributo de Clase.
- **final:** El atributo es una constante, en ese caso debe de tener valor inicial obligatoriamente. No admitirá cambios después de su declaración y asignación de valor. Determina que un atributo no puede ser sobrescrito o redefinido. O sea: no funcionará como una variable “tradicional”, sino como una constante. Por convenio en java las constantes se escriben en mayúsculas.
- **Sin especificar:** Es un Atributo de Instancia y se puede modificar su valor.



# Características

- Abstracción
- Encapsulamiento
- Modularidad



# 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.



# 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.