

Programación Orientada a Objetos

Conceptos Básicos antes de empezar

Conceptos Básicos



1. Programación Orientada a Objetos
2. Principios de la POO
3. Conceptos básicos
4. Clases e instancias
5. Atributos y métodos
6. Accessors y Mutators (Getters y Setters)

Programación Orientada a Objetos



- Se plantea que cada **programa es una simulación de un mundo real o virtual**
- **El mundo está creado por objetos.**
- **Un programa Orientado a Objetos**, debe poder simular efectivamente eventos de un mundo, **debe saber trabajar con objetos y sus relaciones.**
- El enfoque de la programación Orientada a Objetos (OO), requiere un análisis algo diferente del problema a solucionar. **Es necesario pensar en objetos** y no tanto en instrucciones, funciones y flujos de código.
- En esta metodología, es clave conocer algunos de los conceptos que la abarcan: **Objetos, clases, mensajes, métodos, atributos.**

Programación Orientada a Objetos

- **Antecesor: Programación Estructurada**

Utiliza el **concepto de función** como principal recurso. Descompone el problema en sucesivas llamadas a funciones, y éstas **describen qué debe hacer un programa en términos de acciones**. Ejemplos: Pedir un dato, calcular monto, ordenar BD, etc.

- **Programación orientada a objetos (POO)**

Otra forma de pensar los problemas. **Se basa en la descomposición en objetos**, y el enfoque no es tanto en qué se debe hacer, sino **cuál es el escenario real** del mismo, y así intentar crear ese escenario en nuestro programa.

- **La programación orientada a objetos define como el conjunto de objetos y mensajes intercambian entre sí para dar solución a un problema.**

- La diferencia con la **programación estructurada** es que ésta se define como una secuencia de instrucciones que definen **cómo debe ser resuelto un problema**.

Principios de la POO



- Existen cuatro principios fundamentales que definen a la Programación Orientada a Objetos (POO):
 1. **Abstracción,**
 2. **Encapsulamiento,**
 3. **Herencia y**
 4. **Polimorfismo.**

Principios de la POO



1. Abstracción:

- Es el principio por el cual se definen datos en función de las interfaces y la funcionalidad, y no en función de su implementación.
- Se conoce el qué, pero no es necesario que conozcan el cómo.
- Esto permite reducir la complejidad de un desarrollo.

Principios de la POO



2. Encapsulamiento:

- Muy asociado al anterior.
- Es la habilidad de ocultar el interior de un objeto al mundo exterior, y permitir sólo el acceso por medio de métodos llamados **accessors** y **mutators**.
- Permite modificar el contenido de un módulo sin afectar a quién lo esté llamando.

Principios de la POO



3. Herencia:

- Es el principio que permite clasificar y distribuir el conocimiento de nuestro sistema en forma jerárquica, permitiendo reutilizar código y no generar redundancias.
- La mayor ventaja es no sólo la del reuso, sino la facilidad para extender la funcionalidad de un componente.

Principios de la POO



4. Polimorfismo:

- Significa un nombre / múltiples formas.
- Es el principio por el cual, una misma acción (con un determinado nombre) puede tener múltiples implementaciones y funcionalidades, dependiendo del objeto en el que se aplique.

Objetos

.....

- Son la **representación de un concepto**.
- Contienen toda la información necesaria para abstraer un concepto.
- Nos referimos a cualquier sustantivo que pueda ser necesario considerar en un programa. Desde objetos bien definidos como personas, animales u objetos inanimados, hasta aspectos más subjetivos como características (color, altura, etc.), eventos (conexión, interrupción, etc.), estados (reposo, movimiento, etc.)
- Prácticamente TODO puede ser considerado un objeto que puede interactuar con otros.

Objetos

- Están siempre caracterizado por **atributos** que describen su apariencia, y por **métodos** que describen su comportamiento.
 - Ejemplos:
 - 1) El objeto Lápiz puede interactuar con el objeto Figura para poder trazar la figura.
 - 2) El objeto Cliente puede interactuar con el objeto ATM para operar con sus cuentas. A su vez el ATM puede interactuar con el Banco para obtener datos de las cuentas y permitir operar al cliente con sus cuentas.
- Lápiz** **Figura**
- Cliente** **ATM** **Banco**

Atributos

.....

- Los atributos describen la apariencia de los Objetos.
- Los objetos tienen una apariencia que los distingue y los identifica de otros objetos del mismo tipo.
- Son las **CUALIDADES** de los objetos.

Ejemplos de atributos:

- 1) un lápiz puede ser negro mientras que otro puede ser rojo
- 2) otro puede medir 20 cms, mientras que otro 10.

**Las cualidades “COLOR” y “TAMAÑO”
son claramente atributos objeto “LÁPIZ”.**

Métodos

.....

- Los métodos representan acciones.
- Son el **COMPORTAMIENTO** que describe las responsabilidades y capacidades que poseen los objetos.
(A diferencia de los atributos que representan CUALIDADES)

Ejemplos de métodos:

- 1) un lápiz puede trazar una línea,
- 2) puede escribir una letra,
- 3) puede interactuar con un sacapuntas,
- 4) borrar un trazo (si tuviera goma al final del lápiz), etc.

Los comportamientos “TRAZAR”, “ESCRIBIR”, “SACAR PUNTA”, “BORRAR” son claramente métodos del objeto “LÁPIZ”.

Clases

.....

- En grandes programas existen muchos objetos, y se hace necesario poder clasificarlos de alguna forma.
- **Las CLASES permiten clasificarlos en grupos**
- Cada grupo cuenta con características similares y se los denomina CLASES.

Ejemplos de CLASES:

- 1) Útiles escolares podría agrupar a lápiz, sacapuntas, goma.
- 2) Cartuchera también podría agrupar a lápiz, sacapuntas, goma.

Los agrupamientos como “UTILES ESCOLARES”, “CARTUCHERA” son claramente clases del objeto “LÁPIZ”.

Mensajes

- Son las herramientas que existen para permitir interactuar entre objetos.
- Todo objeto (incluso nosotros como programadores), envía un mensaje a otro objeto para que realice “algo” que necesitamos.
- Este mensaje **podrá contener PARÁMETROS** como si se tratase de una función.
- **Un mensaje será el llamado a uno de los métodos del objeto receptor, pasándole algún tipo de parámetro.**

Mensajes

.....

Ejemplos de MENSAJES:

Suponiendo que existen **los objetos: auto y persona**.

Secuencia de mensajes entre ellos para llegar hasta un destino:

unaPersona  **unAuto** abrir puerta (Antes el auto estaba cerrado, ahora tiene una puerta abierta)

unaPersona  **unAuto** ingresar (Antes el auto estaba vacío, ahora tiene una persona adentro)

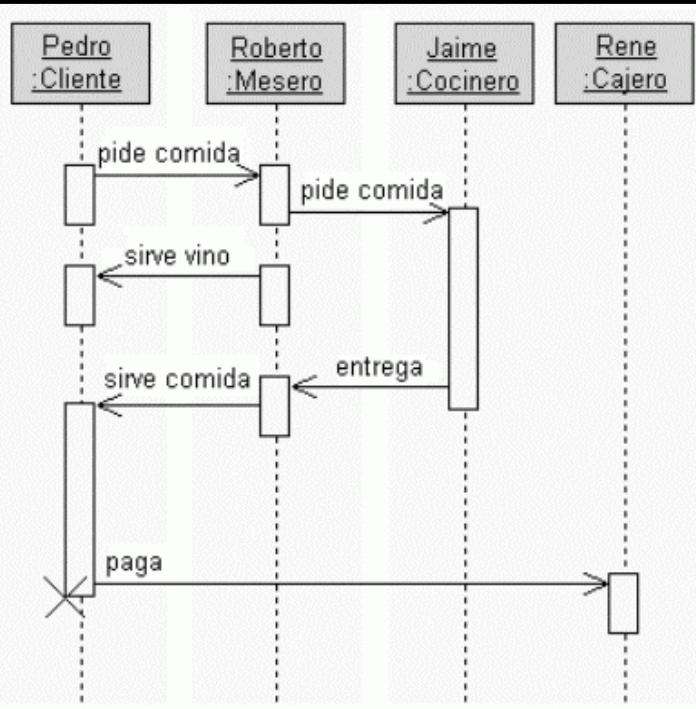
unaPersona  **unAuto** cerrar puerta (Antes el auto tenía una puerta abierta, ahora está cerrado)

unaPersona  **unAuto** ir a (destino) (Antes el auto estaba apagado, ahora está encendido)

unaPersona  **unAuto** arrancar (Antes el auto estaba detenido, ahora está viajando al destino)

unAuto  **unaPersona** llegar a (destino) (Antes la persona no había llegado al destino, ahora llegó al destino)

Diagrama de Secuencia



1. **Pedro** es un objeto tipo **Cliente**, que envía el mensaje “pedir comida” al objeto **Roberto**, que es de tipo **Mesero**. Mientras tanto **Pedro** queda esperando.
 2. **Roberto**, tras recibir el mensaje de **Pedro**, envía un mensaje “pedir comida” al objeto **Jaime**, que es de tipo **Cocinero**.
 3. **Roberto** termina su proceso (y podría continuar atendiendo a otras personas).
 4. **Jaime**, tras recibir el mensaje de **Roberto**, comienza su proceso para cocinar el plato solicitado.
 5. Mientras tanto, **Roberto** puede enviar un mensaje “servir vino” al cliente **Pedro**, para ayudar a esperar su pedido.
 6. Cuando **Jaime** finaliza la cocción, envía un mensaje “entregar comida” al objeto **Roberto**, que es de tipo **Mesero**.
 7. **Roberto**, tras recibir el mensaje de **Jaime**, envía un mensaje “servir comida” al objeto **Pedro**.
- Pedro** entonces inicia su proceso de comer, hasta que desea terminar. Cuando eso pasa, envía un mensaje “pagar” al objeto **René**, que es de tipo **Cajero**, y finaliza todo el proceso completo.

Notar que no existe mensaje entre los objetos **Cocinero** y **Cliente**.

Clases e Instancias

¿Cuál es la diferencia entre objeto y clase?

- La CLASE es una estructura que define cómo deben ser los objetos, qué atributos y métodos debe tener.
Representa una abstracción de datos ya que no rebela detalles de implementación.
- El OBJETO es una “instancia” de una clase, es decir, es una clase instanciada con valores particulares que lo distinguen de otras instancias de esa misma clase.

Clases e Instancias

Ejemplo del Restaurant

- **clase Cliente** es una estructura que define los atributos: nombre, DNI, cant_dinero, etc. y define los métodos: comer, beber, etc. La clase
- **clase Mesero** en cambio tendría atributos: nombre, antigüedad, horario_laboral, etc. y métodos: registrar pedido, recibir plato preparado, recibir vino, etc.

Observar que EN UNA CLASE NO HAY VALORES CONCRETOS para sus atributos. Ahí entra en juego la “INSTANCIA”. UNA INSTANCIA del OBJETO es definir una “fotografía” de la clase con valores concretos.

- **objeto Cliente** podría tener nombre=Pedro, dni:28756132, cant_dinero: \$380
- **otro Cliente** podría tener nombre=María, dni:29786135, cant_dinero:\$150.30.
- **objeto Mesero** podría tener nombre=Roberto, antigüedad:12 años, horario:11a17
- **y otro mesero** podría ser nombre:Inés, antigüedad:3 años, horario: 20a24hs.

Clases e Instancias

.....

Para empezar a trabajar con programación orientada a objetos, lo primero que es necesario saber es cómo crear una clase.

- **La sintaxis es similar a la de una función, pero sin parámetros:**

```
class nombreClase {  
    Código de la clase  
}
```

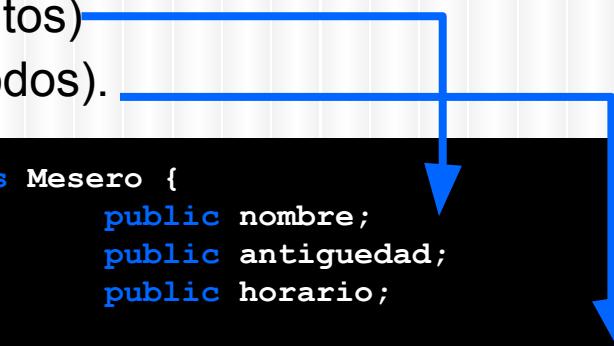
Atributos y Métodos

- Dentro del código de la clase deben ir dos secciones:

- La sección de las variables (atributos)
- La sección de las funciones (métodos).

```
class Cliente {  
    public nombre;  
    public dni;  
    public cant_dinero;  
  
    public function comer(){}
    public function beber(){}
}
```

```
class Mesero {  
    public nombre;  
    public antiguedad;  
    public horario;  
  
    public function registrarPedido(){}
    public function recibirPlato(){}
    public function recibirVino(){}
}
```



Atributos y Métodos

.....

- Una vez creada la clase, ya es posible crear instancias de ella con la siguiente sintaxis:

```
objeto = new nombreClase();
```

- Ejemplos:

Ejemplos:

```
unCliente = new Cliente();  
  
mozo = new Mesero();
```

Atributos y Métodos

- ATRIBUTOS se definen con VARIABLES y los MÉTODOS con FUNCIONES.
- Pueden ser **privados, públicos, estáticos o protegidos**.
- Un **atributo o variable pública** significa que puede ser accedida desde dentro y fuera de la clase.
- Un **atributo o variable privada** significa que puede ser accedida únicamente desde dentro de la misma clase.

Ejemplo:

```
class Cliente {  
    private nombre;  
    private dni;  
    private cant_dinero;  
    private function comer() {}  
    private function beber() {}  
    public  function abandonar() {}  
}
```

Este código indica que sólo el mismo cliente conoce su nombre, dni y cantidad de dinero que lleva encima. También sólo él puede iniciar el proceso de comer y beber. Pero la acción de abandonar el restaurante, puede ser ejecutada por otro objeto (derecho de admisión)

Atributos y Métodos

.....

Para invocar a atributos y métodos, se usa la siguiente sintaxis:

Ejemplos:

```
objeto.atributo;  
objeto.metodo();
```

```
unCliente = new Cliente();  
unCliente.nombre = ToString("Pedro");  
echo unCliente.dni;  
unCliente.comer("Sandwich");
```

Este código considera que todos los atributos y métodos son públicos.

La sentencia “new” crea una instancia. El operador “.” permite acceder a los atributos y métodos del objeto.

Atributos y Métodos

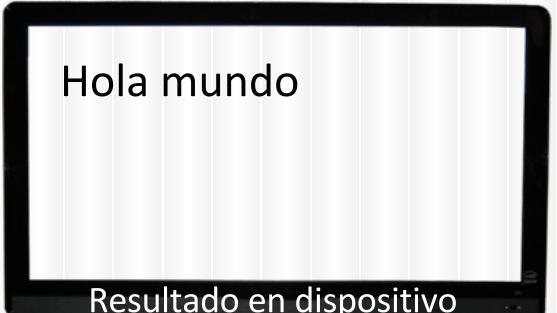
Como los métodos son funciones, éstos pueden recibir parámetros.

Como los atributos son variables, éstos pueden inicializarse con un valor default.

Si no se especifica el alcance (private o public) se considera public por default.

Ejemplos:

```
class Prueba {  
    function mostrar((String) texto){  
        echo texto;  
    }  
  
    objeto = new Prueba();  
    objeto.mostrar("Hola mundo");
```



Hola mundo

Resultado en dispositivo

Accessors(Get) y Mutators (Set)

- Son dos tipos de métodos que permiten exteriorizar y modificar datos privados de un objeto.
 - Un **accessor** accede al objeto para **consultar** el valor de un atributo.
 - Un **mutator** accede al objeto para **modificar** el valor de un atributo.

Permiten implementar el principio de **encapsulamiento**.

- Algunas bibliografías llaman a los accessors “**getters**” y a los mutators “**setters**”. Porque los primeros “obtienen” valores y los segundos “sestean” valores.

Ejercicios

.....

Para el siguiente problema, identificar posibles objetos y mensajes:

- 1) Para los siguientes casos, identificar posibles atributos y métodos:
 - Clase Reloj: Debe simular el comportamiento de un cronómetro digital.
 - Clase Vehículo: Debe simular el funcionamiento básico de un auto.
 - Clase Teléfono
 - Clase Televisor

Crear las clases y métodos

Crear dos instancias de cada clase creada.

Ejercicios

.....

- 2) Para el siguiente problema, identificar objetos y mensajes, subrayando lo importante:

Definición del problema: Desarrollo de un sistema sencillo de procesamiento de texto

Descripción de la solución: El sistema de procesamiento de texto permite a los usuarios crear documentos. Los documentos creados se pueden archivar en un directorio. Los usuarios pueden imprimir o mostrar sus documentos. Se pueden modificar los documentos. También se pueden borrar del directorio.

- Crear las clases y métodos
- Crear dos instancias de cada clase creada.

Ejercicios

.....

- 3) Para el siguiente problema, identificar posibles objetos y mensajes:

Se pide modelar un SISTEMA DE CONSULTA BANCARIA.

Para ello se plantean los siguientes requerimientos:

1. Los clientes deberán interactuar con sus cuentas por medio de un ATM.
2. Las cuentas son almacenadas en una BD dentro del banco.
3. Los clientes deben validarse con su tarjeta y su PIN para poder interactuar con el ATM.
4. La única actividad que permite el ATM es la consulta de saldos en sus cuentas.

Efectuar un análisis más detallado para descartar datos poco relevantes