Programación Avanzada

UNIDAD 5: PROGRAMACIÓN ORIENTADA A OBJETOS

Actividad por BD

Al ingresar al siguiente link, identificarse con nombre y apellido.

https://es.educaplay.com/recursos-educativos/16912197-base_de_datos.html

Unidad 5

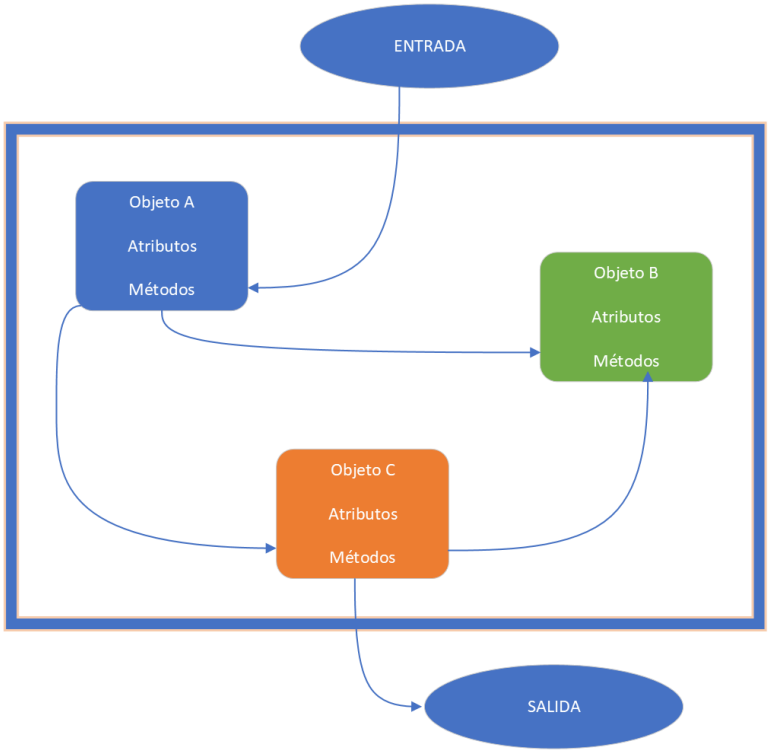
Programación Orientada a Objetos. Clases y Objetos. Instanciación de objetos. El objeto this. El constructor. Extensión y herencia. Visibilidad y encapsulación

¿QUÉ ES LA POO?

La Programación Orientada a Objetos (POO) es un **paradigma de programación**.

Se basa en el concepto de **clases** y **objetos**.

Este tipo de programación se utiliza para estructurar un programa de software en piezas simples y reutilizables de planos de código (clases) para crear instancias individuales de objetos.



Reseña Histórica de la POO

Años 60: Simula. (Noreweigan Computing System):

- Inspirado por los problemas involucrados en la simulación de sistemas de la vida real.

Años 70: Smalltalk (Xerox PARC):

- Alan Kay y su equipo de investigación desarrollan Smalltalk basándose en los conceptos que introducía Simula.

Años 80: Smalltalk 80 y C++ (Laboratorios Bells)

- Casi contemporáneo al trabajo de Kay desde los laboratorios Bell desarrollan C++ como extensión del lenguaje C.
- Comienzan proyectos similares, como Objective-C, Eiffel, Actor, entre otros.

Reseña Histórica de la POO

Años 90: Java (Sun Microsystems)	2002: C# y .NET (Microsoft)	Recientemente
• Es el resultado de la búsqueda de una plataforma independiente del hardware adoptando los conceptos de OO.	• Microsoft lanza su plataforma .NET para el desarrollo de aplicaciones multiplataforma en respuesta a la popularidad de JAVA. Junto con la plataforma aparece el lenguaje C#.	• Ruby, Scala, Phyton, etc.

Ventajas de la POO

- **Reutilización de Código:** La POO facilita la reutilización de código a través de conceptos como la herencia y la composición. Esto permite aprovechar clases existentes para crear nuevas clases, lo que ahorra tiempo y esfuerzo.
- **Organización del Código:** La POO proporciona una estructura organizada para el código al dividirlo en ***clases*** y ***objetos***. Esto mejora la modularidad y facilita la comprensión y mantenimiento del software.
- **Abstracción:** Permite la abstracción de detalles complejos, centrándose en la representación de objetos del mundo real. Esto facilita la comprensión del sistema y la resolución de problemas.
- **Mantenibilidad:** La modularidad y la estructura organizada de la POO facilitan la identificación y corrección de errores. Los cambios y actualizaciones se pueden realizar de manera más eficiente sin afectar otras partes del sistema.
- **Escalabilidad:** La POO facilita la escalabilidad del software al permitir la adición de nuevas funcionalidades mediante la creación de nuevas clases o la extensión de clases existentes.

Desventajas de la POO

- Cambio en la forma de pensar de la programación tradicional, a la orientada a objetos
- La ejecución de programas orientada a objetos es más lenta
- Complejidad para adaptarse
- Dificultad en la abstracción

POO

EN EL PARADIGMA DE OBJETOS, SÓLO HAY **OBJETOS** Y **MENSAJES** (QUE TAMBIÉN SON OBJETOS).

PROGRAMA ORIENTADO A OBJETOS

Un programa en POO es un conjunto de objetos que colaboran enviándose mensajes

CONCEPTOS GENERALES

PROGRAMACIÓN ORIENTADA A OBJETOS

Clases y Objetos

Definición: "En POO, una clase es un plano para crear objetos, mientras que un objeto es una instancia específica de una clase. Las clases definen atributos y métodos comunes para los objetos."

Ejemplos:

Una clase 'Vehículo' puede tener

- atributos como 'color' y 'modelo', y
- métodos como 'arrancar' y 'detener'.”

Una clase ‘Persona’ puede tener

- Atributos como ‘nombre’, ‘apellido´, ‘edad’, ‘genero’
- Métodos como ‘esMayorEdad’

Clase

Una clase es una plantilla mediante la cual se crean los diferentes objetos requeridos para la solución del problema. Los objetos son instancias de las clases.

Una clase se compone de:

- **Información:** campos (atributos, propiedades)
- **Comportamiento:** métodos (operaciones, funciones)

Ejemplo:

- Clase VEHICULO
- Clase PERSONA

ATRIBUTOS

- propiedad1
- propiedad2
- propiedad3

METODOS

- metodo1()
- metodo2()
- metodo3()

Objeto

Un objeto es una instancia de una clase. Por lo tanto, los objetos hacen uso de los ***Atributos*** (variables) y ***Métodos*** (Funciones y Procedimientos) de su correspondiente clase.

Es una variable de tipo clase.

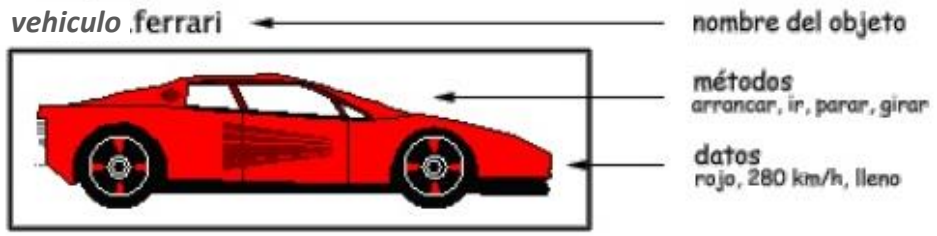
- Por ejemplo
  - el objeto **ferrari** es un objeto de tipo Clase: **vehículo**.
  - El objeto **Juan** es un objeto de la clase **persona**

Un objeto permite modelar entidades del mundo real

Clase y Objeto: Ejemplo

Clase:
VEHICULO

- **Objeto**: *Ferrari*



Objetos: Identificación

La primera tarea a la que se enfrenta un programador en POO es la identificación de los objetos inmersos en el problema a solucionar.

Los objetos generalmente se ubican en las siguientes categorías:

- Cosas tangibles: ***árbol, auto, etc***
- Cosas intangibles: ***emociones, marca***
- Organizaciones o entidades: ***universidad, empresa de transporte***
- Roles: ***alumno, profesor, etc***

Objetos y Atributos

Sea la clase PERSONA y el objeto JUAN

- El nombre de Juan es Juan Manuel
- El apellido de Juan es Rodriguez
- La edad de Juan es 27
- El género de Juan es Masculino

«Atributos, determinan el estado interno de un objeto»

Objetos y Comportamientos

Sea la clase PERSONA y el objeto JUAN

- Juan es mayor de edad
- Juan habla
- Juan compra tickets
- Juan viaja en avión
- Juan entra a un hotel

«Comportamiento, determina el protocolo del objeto»

Características de la POO

- ABSTRACCIÓN
- ENCAPSULAMIENTO
- MENSAJES
- POLIMORFISMO
- HERENCIA

Abstracción

Es una de las principales características a tener en cuenta ya que permite vislumbrar los diferentes agentes u objetos implicados en un problema.

Captar los atributos y métodos que conforman cada objeto y la relación que existen entre ellos.

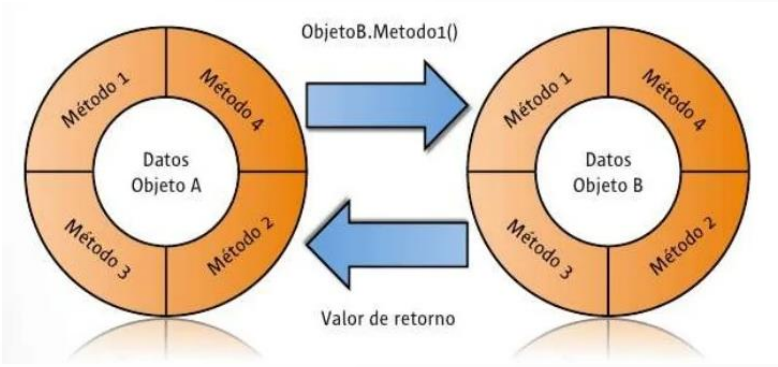
Resolver el problema en subproblemas donde cada objeto se haga cargo de cada subproblema.

La comunicación entre objetos genera la solución general a todo el problema (Divide y vencerás)

Encapsulamiento

Permite la ocultación de la información es decir permite asegurar que el contenido de un objeto se pueda ocultar del mundo exterior dejándose ver lo que cada objeto necesite hacer público.

Ejemplo: Una persona desea llevar su auto descompuesto para que sea arreglado por un mecánico

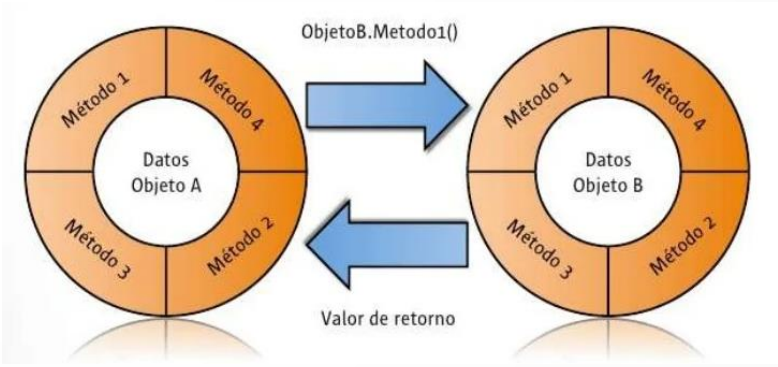


Mensajes

Un objeto sin comunicación con el mundo exterior no es de utilidad. Los objetos deben relacionarse.

Los objetos interactúan entre ellos mediante mensajes.

Cuando un objeto A quiere que otro objeto B ejecute una de sus funciones o procedimientos (métodos de B), el objeto A manda un mensaje al objeto B.

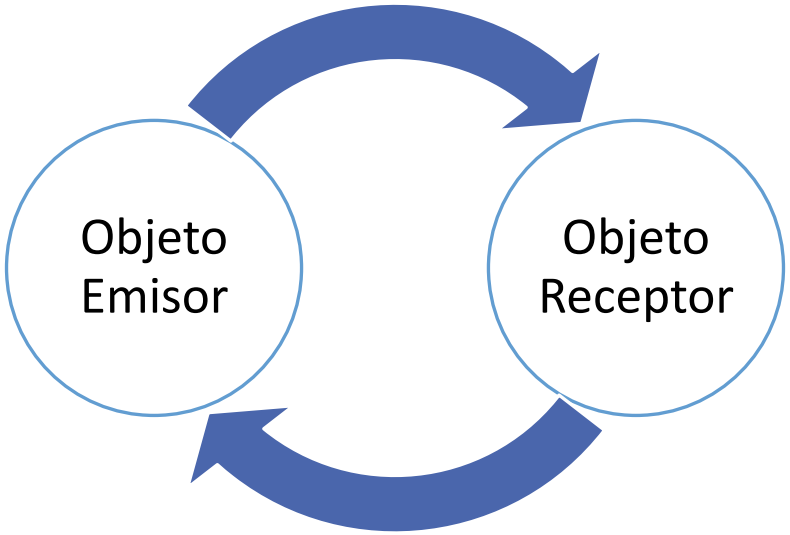


Mensajes

En un mensaje siempre hay un receptor, lo cual no ocurre en una llamada a procedimiento.

Un mensaje consta de 3 partes:

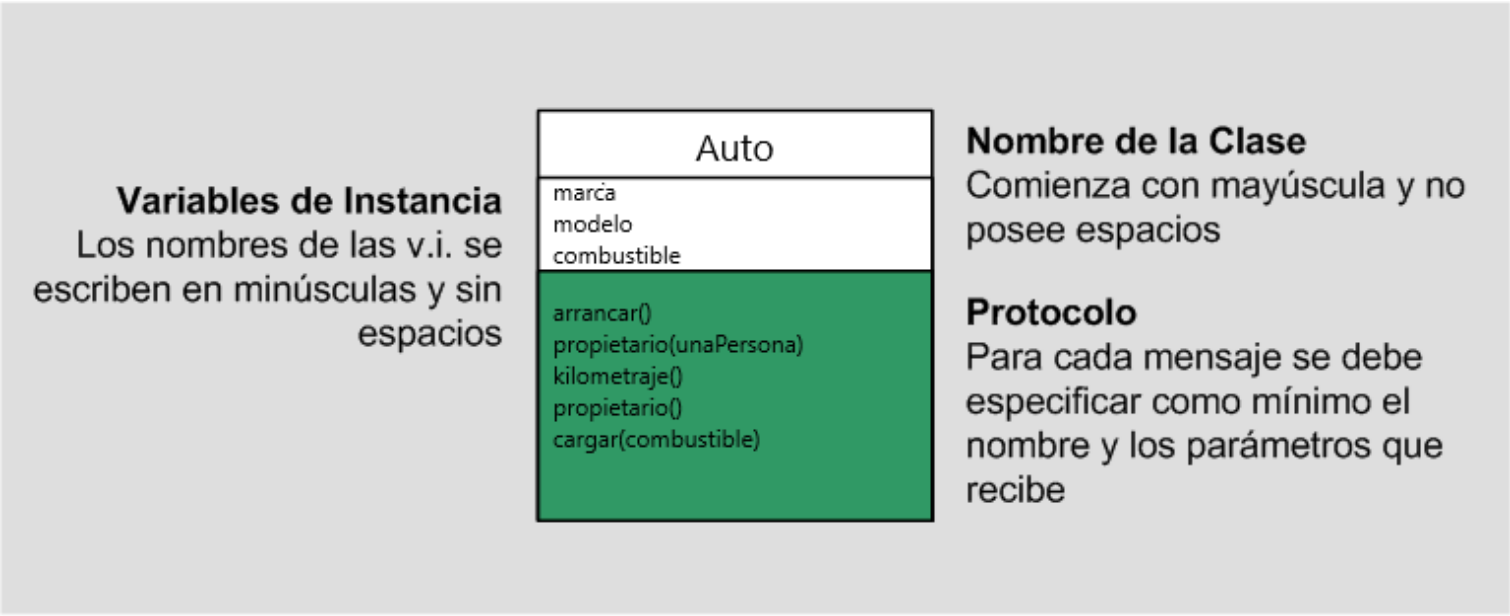
1. Identidad del receptor: Nombre del objeto que contiene el método a ejecutar.
2. Nombre del método a ejecutar: Solo los métodos declarados públicos.
3. Lista de Parámetros que recibe el método (cero o más parámetros)



ESPECIFICACIÓN DE CLASES

Las clases se especifican por medio de un nombre, el estado o estructura interna que tendrán sus instancias y los métodos asociados que definen el comportamiento

Gráficamente:



Formas de Conocimientos

Para que un objeto conozca a otro lo debe poder nombrar. Decimos que se establece una ligadura (binding) entre un nombre y un objeto.

Algunas de las formas de conocimiento o tipos de relaciones entre objetos son

- Conocimiento Interno: Variables de instancia.
- Conocimiento Externo: Parámetros.

Además, existe una forma de conocimiento especial: las seudo variables.

Variables de Instancia

Define una relación entre un objeto y sus atributos.

Se definen explícitamente como parte de la estructura de la clase.

La relación dura tanto tiempo como viva el objeto, o se cambie explícitamente.

Parámetros

Se refiere a los parámetros de un mensaje.

El nombre de la relación se define explícitamente en el nombre del método.

La relación de conocimiento dura el tiempo que el método se encuentra activo.

La ligadura entre el nombre y el objeto no puede alterarse durante la ejecución del método.

Constructores

Los constructores son funciones miembro especiales que sirven para inicializar un objeto de una determinada clase al mismo tiempo que se declara.

Los constructores son especiales por varios motivos:

- Tienen el mismo nombre que la clase a la que pertenecen.
- No tienen tipo de retorno, y por lo tanto no retornan ningún valor.
- No pueden ser heredados.
- Por último, deben ser públicos, no tendría ningún sentido declarar un constructor como privado, ya que siempre se usan desde el exterior de la clase, ni tampoco como protegido, ya que no puede ser heredado.

Destructores

Los **destructores** son funciones miembro especiales que sirven para eliminar un objeto de una determinada clase.

Al igual que los constructores, los destructores también tienen algunas características especiales:

- También tienen el mismo nombre que la clase a la que pertenecen.
- No tienen tipo de retorno, y por lo tanto no retornan ningún valor.
- No tienen parámetros.
- No pueden ser heredados.
- Deben ser públicos, no tendría ningún sentido declarar un destructor como privado, ya que siempre se usan desde el exterior de la clase, ni tampoco como protegido, ya que no puede ser heredado.
- No pueden ser sobrecargados, lo cual es lógico, puesto que no tienen valor de retorno ni parámetros, no hay posibilidad de sobrecarga.

La Seudo Variable: THIS

Para mandarle un mensaje a un objeto hay que poder nombrarlo.

¿Cómo hace un objeto para mandarse un mensaje a sí mismo?

Seudo variable: como una variable ordinaria pero:

- No se declara
- No puede modificarse

Java y C# : this

Smalltalk : self

Con esta seudo variable el objeto puede hacer referencia a sí mismo.

Clases en C++. Ejemplo

```
class MiClase
{
        int var1; //Variable de clase
        const double var2 = 3.14159;  //Variable de clase

        void cambiarVar1(int a); //Declaración de un método de la clase
        double calcularArea(const double& x, const double& y); //Declaración de un método de la clase
};

void MiClase::cambiarVar1(int a)  //Definición del método por fuera de la clase
{
        var1 = a;
}

double MiClase::calcularArea(const double& x, const double& y) //Definición del método por fuera de la clase
{
        return x*y*var2;
}
```

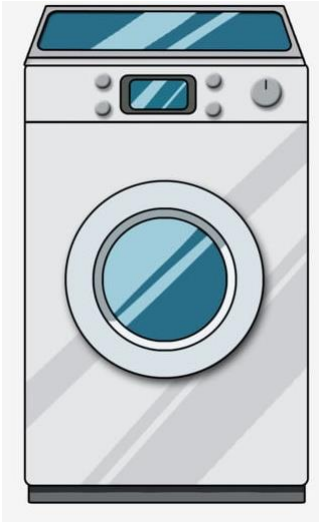
```
int main()
{
        MiClase miObjeto; //Declarando un objeto de la clase

        miObjeto.cambiarVar1(5); //Accediendo a un miembro con el operador punto
        double var = miObjeto.calcularArea(34.6, 23.9); //Accediendo a un miembro con el operador punto
}
```

Defina la clase (atributos y métodos) para cada uno de los elementos e instancie 1 objeto de cada uno

1

2



3

fin 1ra parte...