Programación Front End (nivel inicial)

Ing. Leonardo Torres, Lic. Gustavo Lafuente

FACUTLAD DE INGENIERÍA
Universidad Nacional de la Pampa

Argentina programa 4.0

Conceptos fundamentales

POO - Programación Orientada a Objetos

Es un **paradigma** de programación que viene a innovar la **forma de obtener resultados**. Los objetos se utilizan como metáfora para emular las entidades reales del negocio a modelar.

Muchos de los **objetos prediseñados** de los lenguajes de programación actuales permiten la agrupación en bibliotecas o librerías, sin embargo, muchos de estos lenguajes permiten al usuario la creación de sus **propias bibliotecas**.

Se centra en los objetos que los desarrolladores quieren manipular en lugar de enfocarse en la lógica necesaria para manipularlos. Este enfoque de programación es adecuado para programas que son grandes, complejos y se actualizan o mantienen activamente.

Es una teoría que suministra la base y el modelo para resolver problemas.

POO - Elementos

Clases: Las clases pueden ser definidas como un molde que contendrá todas las características y acciones con las cuales podemos construir N cantidad de objetos.

Propiedades: Las propiedades son las características de una clase, tomando como ejemplo la clase humanos, las propiedades podrían ser: nombre, el género, la altura, color de cabello, color de piel, etc.

Métodos: Los métodos son las acciones que una clase puede realizar, siguiendo el mismo ejemplo anterior, estas podrían ser: caminar, comer, dormir, soñar, respirar, nadar, etc.

Objetos: Son aquellos que tienen propiedades y comportamientos, estos pueden ser físicos o conceptuales.

Por ejemplo: El objeto “Arnell”, quien es una instancia de la clase humanos.

POO - Pilares

Abstracción: Es cuando separamos los datos de un objeto para luego generar un molde (una clase).

Encapsulamiento: Lo puedes utilizar cuando deseas que ciertos métodos o propiedades sean inviolables o inalterables.

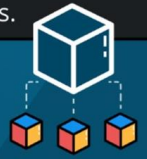
Un ejemplo del encapsulamiento podría ser una cuenta de banco, donde el usuario no puede simplemente aumentar su balance de dinero, si no que debe depender de unos métodos previamente validados para aumentar dicho balance (depósitos, transferencias, etc).

Herencia: Nos permite crear nuevas clases a partir de otras. Si tuviéramos una clase “Autos” y quisiéramos crear unas clases “Auto deportivo” o “Auto clásico”, podríamos tomar varias propiedades y métodos de la clase “Autos”. Esto nos da una jerarquía de padre e hijo.

Polimorfismo: Proviene de Poli = muchas, morfismo = formas. Se utiliza para crear métodos con el mismo nombre pero con diferente comportamiento.

¿QUÉ ES LA PROGRAMACIÓN ORIENTADA A OBJETOS?

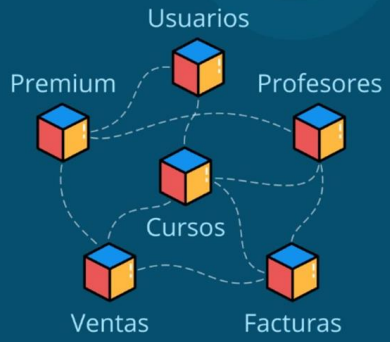
Es un paradigma de programación que organiza las funciones en entidades llamadas objetos.



- **Los objetos se crean a partir de una plantilla llamada clase**. Cada objeto es una instancia de su clase.

- Los objetos tienen **datos (atributos)** y **funcionalidades (métodos).**

- En una aplicación los objetos están separados **pero se comunican entre ellos.**



ATRIBUTOS

```
Nombres
Apellidos
Correo
Contraseña
Premium
```

MÉTODOS

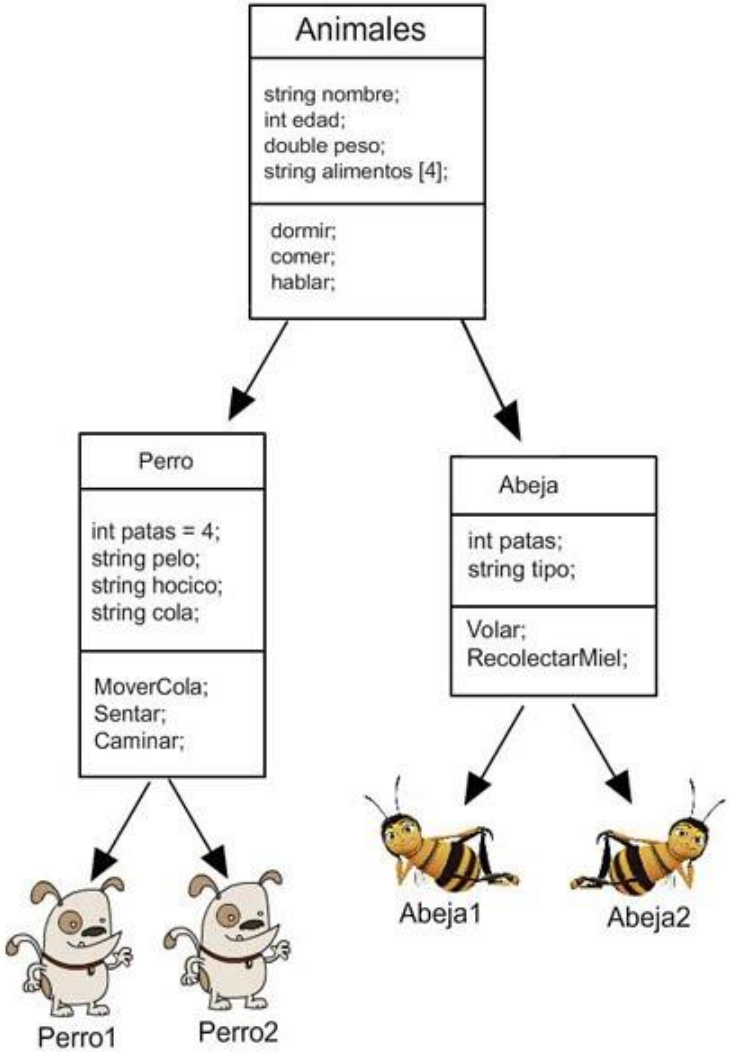
```
Editar perfil
Iniciar sesión
Cerrar sesión
Cambiar contraseña
Pasar a premium
```

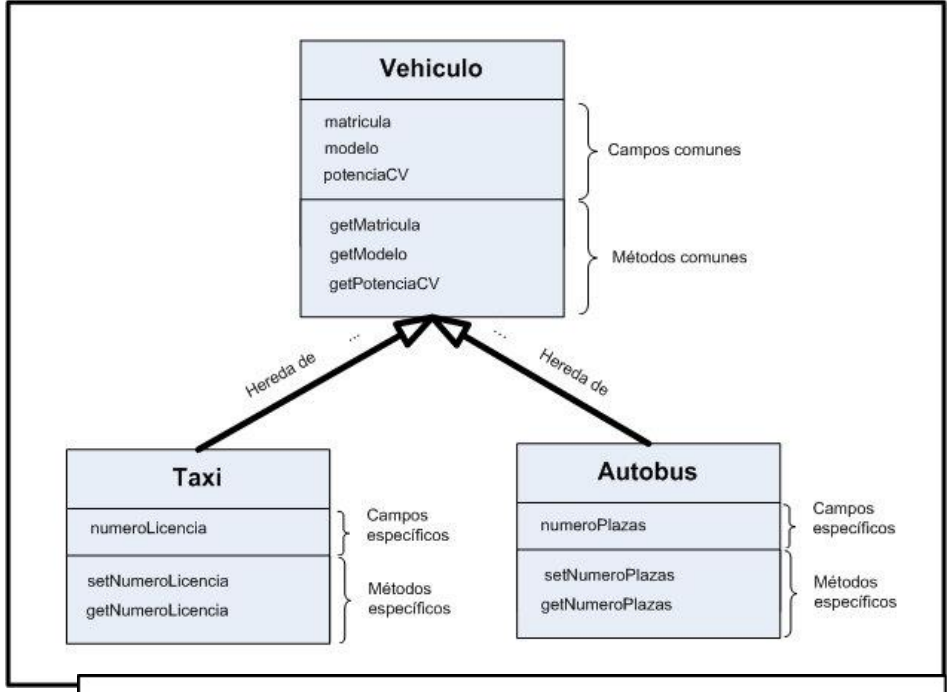
Puedes programar con este paradigma **en la mayoría de lenguajes.**

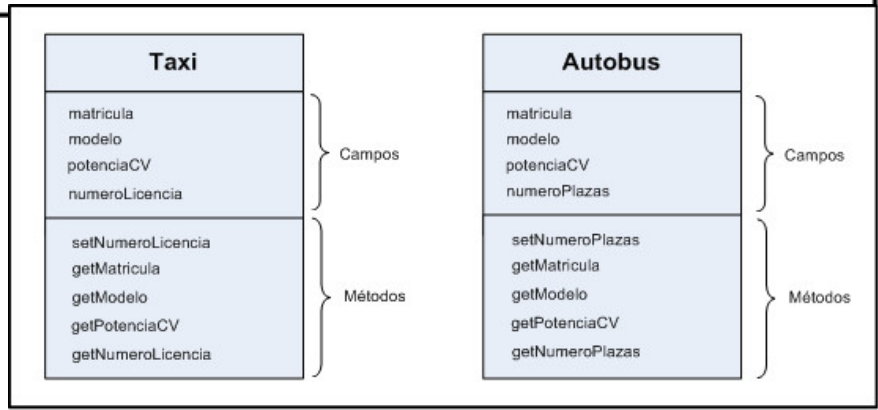
Aprende a crear aplicaciones usando la POO en:

ed.team/cursos

EDteam


Animales
string nombre;
int edad;
double peso;
string alimentos [4];
dormir;
comer;
hablar;
Perro
int patas = 4;
string pelo;
string hocico;
string cola;
MoverCola;
Sentar;
Caminar;
Abeja
int patas;
string tipo;
Volar;
RecolectarMiel;
Perro1
Perro2
Abeja1
Abeja2


Vehiculo
matricula
modelo
potenciaCV
Campos comunes
getMatricula
getModelo
getPotenciaCV
Métodos comunes
Hereda de
Hereda de
Taxi
numeroLicencia
Campos específicos
setNumeroLicencia
getNumeroLicencia
Métodos específicos
Autobus
numeroPlazas
Campos específicos
setNumeroPlazas
getNumeroPlazas
Métodos específicos


Taxi
matricula
modelo
potenciaCV
numeroLicencia
Campos
setNumeroLicencia
getMatricula
getModelo
getPotenciaCV
getNumeroLicencia
Métodos
Autobus
matricula
modelo
potenciaCV
numeroPlazas
Campos
setNumeroPlazas
getMatricula
getModelo
getPotenciaCV
getNumeroPlazas
Métodos

POO - JS

Esto permite que en JavaScript podamos trabajar en un estilo de **POO mediante prototipos**, que es la manera en la que JavaScript nos acerca al manejo de objetos mediante clases de un modo similar al que ya conocemos de otros lenguajes como por ejemplo puede ser el caso de **JAVA**.

Las clases son por decirlo de algún modo simple "**funciones especiales**", como las expresiones y declaraciones de funciones. Su sintaxis tiene dos componentes: expresiones de clases y declaraciones de clases.

Una expresión de clase es otra manera de definir una clase. Existen dos tipos, **nombradas o anónimas**.

```
class Rectangulo {
  constructor(alto, ancho) {
    this.alto = alto;
    this.ancho = ancho;
  }
}
```

```
// Anonima
let Rectangulo = class {
  constructor(alto, ancho) {
    this.alto = alto;
    this.ancho = ancho;
  }
};
```

```
// Nombrada
let Rectangulo2 = class Rectangulo2 {
  constructor(alto, ancho) {
    this.alto = alto;
    this.ancho = ancho;
  }
};
```

```
class patata {
  constructor(tipo) {  // Constructor
    this.patatanombre = tipo;
  }
}
mycar = new patata("Red Pontiac");
console.log(mycar.patatanombre);
```

```
class Rectangulo {
  constructor (alto, ancho) {
    this.alto = alto;
    this.ancho = ancho;
  }
  // Getter
  get area() {
     return this.calcArea();
   }
  // Método
  calcArea () {
    return this.alto * this.ancho ;
  }
}

const cuadrado = new Rectangulo (10, 10);

console.log(cuadrado.area); // 100
```

```
class Animal {
  constructor(nombre) {
    this.nombre = nombre;
  }

  hablar() {
    console.log(this.nombre + ' emite un ruido.')
  }
}

class Gato extends Animal {
  hablar() {
    console.log(this.nombre + ' maulla.');
  }
}
```

this

Es un valor complejo en Javascript. Su valor, lejos de ser intuitivo, se determina según la forma en que se invoque a la función que lo encierra, algo que no siempre es correctamente entendido o aprovechado.

La palabra clave this tiene en Javascript un comportamiento diferente al de otros lenguajes pero por lo general, **su valor hace referencia al propietario de la función que la está invocando o en su defecto**, al objeto donde dicha función es un método.

Cuando no estamos dentro de una estructura definida, esto es un objeto con métodos, el propietario de una función es siempre el contexto global. En el caso de los navegadores web, tenemos que recordar que dicho objeto es **window**

```
console.log( this === window );  // true

function test(){
  console.log( this === window);
}

test(); // true
```

```
let yo = {
  nombre: 'yeison',
  edad: 22,
  hablar: function () {
    console.log(this.nombre);
  }
};

yo.hablar(); // yeison
```

```
let decirNombre = function(obj) {
  obj.hablar = function() {
  console.log(this.nombre);
  };
};

const Mateo = {
  nombre: 'Mateo',
  edad: 22
};

const juan = {
  nombre: 'Juan',
  edad: 25
};

decirNombre(juan);
decirNombre(Mateo);

juan.hablar(); // Juan
Mateo.hablar(); // Mateo
```

```
let Persona = function (nombre, edad, madre) {
  return {
    nombre: nombre,
    edad: edad,
    hablar: function() {
      console.log(this.nombre);
    },
    madre: {
      nombre: madre,
      hablar: function() {
        console.log(this.nombre);
      }
    }
  };
};

const ana = Persona('Ana', 30, 'Clara');

ana.hablar(); // Ana
ana.madre.hablar(); // Clara
```

link interesantes

conceptos

https://desarrolloweb.com/articulos/499.php

javascript

https://developer.mozilla.org/es/docs/Learn/JavaScript/Objects/Object-oriented_JS

https://medium.com/@jmz12/usando-clases-en-javascript-e07f0e25c67d

Programación Front End 2023

FACUTLAD DE INGENIERÍA
Universidad Nacional de la Pampa

Argentina programa 4.0