Computer Science

DEV.F:
DESARROLLAMOS(PERSONAS);

Temas

- POO.
- Clases y Objetos.
- JavaScript y la cadena de prototipos.
- Sugar sintaxis para clases y objetos.
- Catálogo de Hp / TBBT / Materias de sistemas.
- Contexto de **this**.
- Call, apply y bind.

Introducción a la POO

DEV.F:
DESARROLLAMOS(PERSONAS);

¿Qué es la programación orientada a objetos?

Es un paradigma de programación, es decir, un modelo o un estilo de programación que nos da unas guías sobre cómo trabajar con él.

- Se basa en el concepto de clases y objetos.
- Busca dejar de centrarnos en la lógica pura de los programas, para empezar a pensar en objetos.
- En vez de pensar en funciones, pensamos en las relaciones o interacciones de los diferentes componentes del sistema.
- Permite que el código sea reutilizable, organizado y fácil de mantener.

Principios de POO

- **Abstracción:** Todo en el mundo real es un objeto.
- **Encapsulamiento:** Agrupar elementos dentro de un mismo nivel de abstracción.
- **Polimorfismo:** Los objetos pueden tener distintos comportamientos.
- **Herencia:** Las características y funciones de un objeto padre son heredadas por los hijos.
- **Modularidad:** Se separa en archivos llamados clases, las funcionalidades.
- **Principio de ocultamiento:** Los objetos sólo deben acceder a la información por medio de getters y setters.

Conceptos

- **Clase:** Modelo o plantilla a partir del cual se crean objetos.
- **Objeto:** Elemento concreto del mundo real.
- **Atributo:** También se le conoce como característica o propiedad.
- **Método:** También se les conoce como función y son las acciones que el objeto de la clase puede realizar.

Diagrama UML de una clase

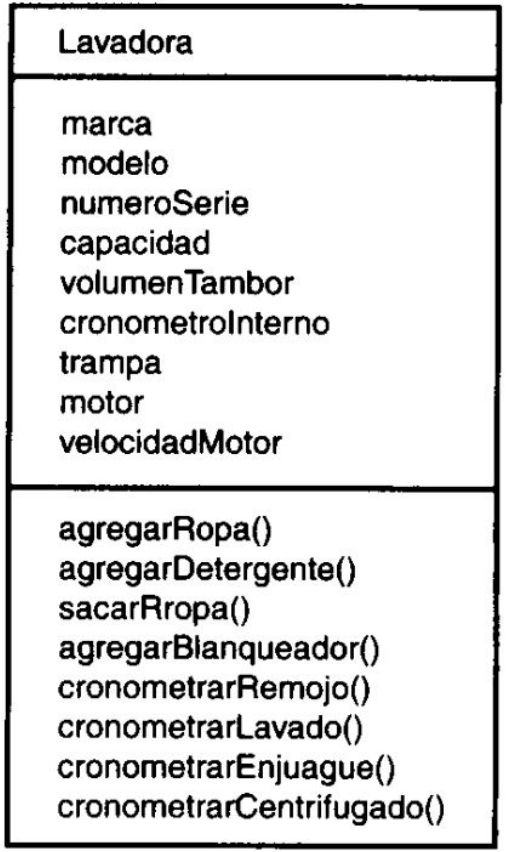
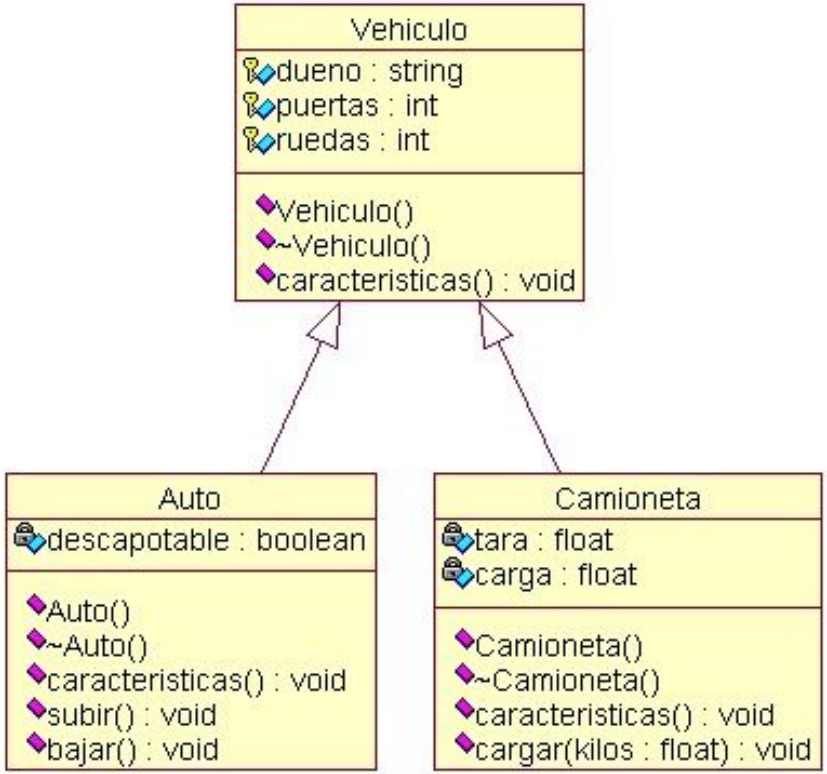


Diagrama UML jerarquía de clases



Práctica 1

- Realizar el diagrama de una jerarquía de clases con al menos dos niveles y 4 clases.

Clases y Objetos

DEV.F:
DESARROLLAMOS(PERSONAS);

Creación de objetos en JS

- const miObjeto = {

  name: ‘jose’

  app: ‘montoya’

  };

Práctica 2

- Utilizar iniciadores de objeto, funciones constructoras.

JavaScript y la cadena de prototipos

DEV.F:
DESARROLLAMOS(PERSONAS);

¿Qué es JS?

Es un lenguaje de programación interpretado usado para páginas web, y es usado en muchos entornos fuera del navegador, tal como Node.js. Está **basado en prototipos**, multiparadigma, de un solo hilo, dinámico, **con soporte para programación orientada a objetos**, imperativa y declarativa (por ejemplo programación funcional).

Principios de la cadena de prototipos

- **Prototipo:** Es homólogo a una clase.
- **Object:** El objeto de mayor jerarquía en JS.
- **Cadena de prototipos:** Simula el concepto de herencia mediante el objeto prototype.

Práctica 3

- Utilizar createObject, new y herencia prototipal.

Sugar Syntaxis para clases y objetos

DEV.F:: DESARROLLAMOS(PERSONAS);

Keywords para clases en JS.

- Class.
- Extends.
- Constructor.
- get y set.
- This

DEV.F

Práctica 4

Generar una jerarquía de clases con sugar syntaxis.

Práctica de Catalogo

DEV.F:
DESARROLLAMOS(PERSONAS);

Práctica 5

Contexto de this

DEV.F:
DESARROLLAMOS(PERSONAS);

Práctica

Realizar practicar sobre el contexto de this en:

- Global.
- En una función.
- En una función con strict mode.
- En objetos.
- Desde fuera de objeto.
- A partir de un objeto nuevo creado por una funcion Constructora.

Call, Apply y bind

DEV.F:
DESARROLLAMOS(PERSONAS);

Práctica

- Utilizar Call, Apply y bind, cambiar un prototipo (node list).
- Alcance de prototype y uso de createObject
- Funciones: hasownproperty, instanceof, typeof, getprototypeof