Clase 4 y 5: Clases Propiedades y objetos

Sitio:	Centro de E-Learning - UTN.BA	Imprimido por:	Nilo Crespi
Curso:	Curso de Backend Developer - Turno Noche	Día:	Friday, 23 de January de 2026, 10:16
Libro:	Clase 4 y 5: Clases Propiedades y objetos		

Descripción

Objetivos de la clase:

- Comprender el concepto de clases y su importancia en la programación orientada a objetos.
- Identificar las propiedades y métodos de una clase y cómo se utilizan para definir la estructura de un objeto.
- Aprender a crear objetos a partir de una clase utilizando el constructor y a inicializar sus propiedades.
- Entender el concepto de herencia de clases y cómo se utiliza para reutilizar código y definir relaciones entre objetos.
- Aprender a crear subclases que hereden propiedades y métodos de una superclase y a agregar nuevas funcionalidades.

Tabla de contenidos

1. Introducción

En esta clase, exploraremos el concepto de clases en la programación orientada a objetos, un pilar fundamental para estructurar código de manera organizada y reutilizable. Aprenderemos cómo las clases funcionan como plantillas para la creación de objetos, definiendo sus propiedades y métodos. También analizaremos cómo los objetos representan instancias concretas de una clase y cómo se utilizan en la práctica mediante constructores para inicializar sus valores.

Además, profundizaremos en la herencia de clases, un mecanismo que permite la reutilización de código y la creación de relaciones entre objetos. Veremos cómo una subclase puede extender una superclase para heredar sus propiedades y métodos, añadiendo nuevas funcionalidades cuando sea necesario. A través de ejemplos prácticos, comprenderemos cómo aplicar estos conceptos en la resolución de problemas cotidianos en el desarrollo de software.

2. Programación orientada a objetos (POO)

Programación orientada a objetos (POO)

Es un paradigma de programación que se basa en el concepto de objetos, que contienen datos y métodos que manipulan esos datos.

Nociones básicas

- **Clases: En TypeScript, las clases se utilizan para definir objetos y sus comportamientos. Una clase es una plantilla para crear objetos que tienen propiedades y métodos comunes. Para definir una clase en TS, utilizamos la palabra clave class, seguida del nombre de la clase y sus propiedades y métodos.**
- **Objetos: Un objeto es una instancia de una clase, que tiene sus propias propiedades y métodos. En TS, se pueden crear objetos usando la palabra clave "new".**
- **Herencia: es un mecanismo que permite crear una nueva clase a partir de una clase existente, heredando sus propiedades y métodos. En TypeScript, se utiliza la palabra clave extends para crear una clase hija que hereda de una clase padre.**
- **Encapsulamiento: se refiere a la ocultación de la información interna de un objeto y la exposición sólo de la información necesaria para su uso externo. En TS, podemos utilizar los modificadores de acceso public, private y protected para definir la visibilidad de las propiedades y métodos de una clase.**

Clases

Las clases son un conjunto de datos que definen la estructura de los objetos y sus métodos.

Propiedades

Las propiedades son las características que definen los objetos, como su nombre, edad, color, etc.

Objetos

Los objetos son las instancias de las clases, es decir, son los datos concretos que se crean a partir de la plantilla definida por la clase.

Por ejemplo, se podría definir una clase "Persona" con propiedades como nombre, edad, dirección, etc. y métodos como saludar, caminar, etc. Luego, se podrían crear objetos de esta clase y llamar a los métodos correspondientes.

Ejemplo

Definir la clase "Persona" con propiedades como nombre, edad, dirección, etc. y métodos como saludar, caminar, etc.

Luego, crear objetos de esta clase y llamar a los métodos correspondientes.

Constructor

El constructor es un método especial que se utiliza para crear objetos a partir de una clase. El constructor se ejecuta automáticamente cuando se crea un objeto a partir de una clase y se utiliza para inicializar las propiedades del objeto. Veamos un ejemplo:

3. Herencia de clases

Herencia de clases

La herencia es un concepto importante en la programación orientada a objetos que permite que una clase herede propiedades y métodos de otra clase. La clase que hereda se llama subclase o clase hija, mientras que la clase de la que se hereda se llama superclase o clase padre. La subclase puede agregar nuevas propiedades y métodos o modificar los existentes. Veamos un ejemplo:

Ejemplo herencia

```
class Vehiculo {
   ….    }

   mostrarDetalles() {
        console.log(`Marca: ${this.marca}, Modelo: ${this.modelo}, Año ${this.anio},
Color: ${this.color}`);
   }
}
class Coche extends Vehiculo {
   acelerar() {
        console.log("Acelerando...");
   }
}

const coche1 = new Coche("Ford", "Mustang", 2022, "Rojo");
coche1.mostrarDetalles();
coche1.acelerar();
```

Entre todos

Pensemos en las siguientes situaciones

Sistema de gestión de pedidos de una tienda

Para nuestro sistema de gestión de pedidos, necesitamos implementar la funcionalidad de calcular el total del pedido a partir de los precios y cantidades de los productos, y permitir agregar nuevos productos al pedido.

- **Crear una clase llamada "Pedido" con las siguientes propiedades: "id", "fecha", "total" y "productos" (un arreglo de objetos "Producto"). Además, debemos crear una clase llamada**

"Producto" que tenga las siguientes propiedades: "nombre", "precio" y "cantidad".

- **Agregar un método llamado "calcularTotal" a la clase "Pedido" que calcule el total del pedido a partir de los precios y cantidades de los productos, y un método llamado "agregarProducto" que permita agregar un nuevo producto al pedido.**
- **Finalmente, crea un objeto a partir de la clase "Pedido" y muestre por consola el total del pedido y los productos que contiene.**

4. ¡A practicar!

Espacio de práctica

1. **Crea una clase llamada "Producto" que tenga las siguientes propiedades: "nombre", "precio" y "cantidad". Luego, crea un objeto a partir de la clase "Producto" y muestra en consola el nombre del producto y su precio.**
2. **Crea una clase llamada "Persona" que tenga las siguientes propiedades: "nombre", "edad" y "profesión". El constructor de la clase debe recibir como parámetros el nombre y la edad, y la propiedad "profesión" debe inicializarse con un valor por defecto de "Desconocido". Luego, crea un objeto a partir de la clase "Persona" y muestra en consola sus propiedades.**
3. **Crea una clase llamada "Animal" que tenga la propiedad "nombre" y el método "mover". Luego, crea una subclase llamada "Perro" que herede de la clase "Animal" y tenga un método adicional llamado "ladrar". Crea un objeto a partir de la clase "Perro" y muestra en consola y/o navegador su nombre y que está moviéndose y ladrando.**