TYPESCRIPT

Programación orientada a objetos

Emilio Sánchez
2º ASIR – SGBD
24/10/2021

Contenido

1. Definición

La programación orientada a objetos (también llamada **POO**) es un paradigma de lenguaje de programación que emplea el concepto de objetos en sus interacciones con el fin de desarrollar programas informáticos.

La POO es diferente de la programación estructurada tradicional, en la que los datos y los procedimientos están separados y sin relación, ya que lo único que se busca es el procesamiento de unos datos de entrada para obtener otros de salida.

2. Características de los objetos

- Un objeto es una entidad que tiene un estado, un método (comportamiento) y una identidad. Un objeto es una abstracción de algún hecho o ente del mundo real, con atributos que representan sus características o propiedades, y métodos que emulan su comportamiento o actividad. Todas las propiedades y métodos comunes a los objetos se encapsulan o agrupan en clases.
- El estado de un objeto son los atributos a los que se les asignan valores (datos). La única manera de modificar el estado de un objeto es a través de sus métodos.
- El método o comportamiento son los mensajes a los que podrá responder dicho objeto. En otras palabras, las funciones u operaciones que se pueden realizar con dicho objeto. Los objetos tienen mecanismos de interacción (llamados métodos) que permiten comunicarse entre sí. Esta comunicación favorece el cambio de estado del objeto.
- La identidad de un objeto es lo que lo diferencia del resto de los objetos. Es un identificador.

3. Características de POO

- **Abstracción:** La abstracción a objetos expresa las características esenciales de un objeto, las cuales distinguen al objeto de los demás (la abstracción genera la ilusión de simplicidad). Además de distinguir entre los objetos provee límites conceptuales. Entonces se puede decir que la encapsulación separa las características esenciales de las no esenciales dentro de un objeto. Si un objeto tiene más características de las necesarias los mismos resultarán difíciles de usar, modificar, construir y comprender.
- **Encapsulamiento:** significa reunir todos los elementos que pueden considerarse pertenecientes a una misma entidad, al mismo nivel de abstracción. Esto permite aumentar la cohesión de los componentes del sistema.
- **Principio de ocultación:** ocultamiento del estado del objeto, es decir, de los datos que pertenecen a un objeto de manera que sólo se pueda cambiar mediante los métodos (u operaciones) definidas para ese objeto. Lo único visible de un objeto para el resto es su interfaz, es decir, los métodos que pueden utilizarse. El aislamiento protege a las propiedades de un objeto contra su modificación por quien no tenga derecho a acceder a ellas; solamente los propios métodos internos del objeto pueden acceder a su estado.

- **Modularidad:** propiedad que permite dividir una aplicación en partes (módulos) más pequeños. Cada módulo debe intentar ser lo más independiente posible de la aplicación.
- **Polimorfismo:** el polimorfismo se refiere a la propiedad por la que es posible enviar mensajes sintácticamente iguales a objetos de tipos distintos. El único requisito que deben cumplir los objetos que se utilizan de manera polimórfica es saber responder al mensaje que se les envía.
- **Herencia:** Las clases se relacionan entre sí y permiten formar una jerarquía de clasificación. Los objetos heredan propiedades y métodos (comportamientos) de las clases a las que pertenecen. La herencia organiza y facilita el polimorfismo y el encapsulamiento, permitiendo a los objetos ser definidos y creados como tipos especializados de objetos preexistentes. Estos pueden compartir (y extender) su comportamiento sin tener que volver a implementarlo.

4. En TypeScript…

El formato para definir clases en TypeScript es el siguiente:

```
class Point {}
```

A continuación se introducen los datos dentro de la clase:

```
class Point {
  x: number;
  y: number;
}
```

```
class Point {
  x = 0;
  y = 0;
}
```

4.1 Subclases

Una de las ventajas que nos da POO es poder crear subclases. Estas subclases toman todos los valores de la clase superior, pero pudiendo añadirle más sin que pertenezcan a la clase superior.

Como podemos ver en el ejemplo se crea una clase animal común para todos y luego una subclcase perro.

El perro hereda todo lo de la clase animal, pero al mismo tiempo podrá definir dentro suya nuevos objetos que no tenga la clase animal. Así se pueden definir parámetros generales para muchas clases distintas sin que choquen entre ellas.

```
class Animal {
  move() {
    console.log("Moving along!");
  }
}

class Dog extends Animal {
  woof(times: number) {
    for (let i = 0; i < times; i++) {
      console.log("woof!");
    }
  }
}
```

4.2 Modificadores de acceso

Cada objeto de la clase puede encontrarse en 3 estados dependiendo de cómo queramos manipular los datos:

- **Public:** nos permite manipular el objeto sin ninguna restricción.
- **Private:** solo nos permite manipular el objeto dentro de la clase.
- **Protected:** nos permite manipular el dato dentro de la clase y en sus subclases.