INSTITUTO SUPERIOR POLITÉCNICO CÓRDOBA

Introducción a la Programación Orientada a Objetos

Contenido

Programación Orientada a Objetos

La programación orientada a objetos (abreviada de ahora en más como POO), es un conjunto de reglas y principios de programación (o sea, un paradigma de programación) que busca representar las entidades u objetos del dominio (o enunciado) del problema dentro de un programa, de la forma más natural posible.

En el paradigma de programación tradicional o *estructurado*, que es el que hemos usado hasta aquí, el programador busca identificar los *procesos* (en forma de subproblemas o módulos) a fin de obtener los resultados deseados. Y esta forma de proceder no es en absoluto incorrecta: la estrategia de descomponer un problema en subproblemas es una técnica elemental de resolución de problemas que los programadores orientados a objetos siguen usando dentro de este nuevo paradigma. Entonces, ¿a qué viene el paradigma de la POO?

La programación estructurada se basa en descomponer procesos en subprocesos y programando cada uno como ***rutina***, ***función***, **o** ***procedimiento*** (todas formas de referirse al mismo concepto) Sin embargo, esta forma de trabajar resulta difícil al momento de desarrollar sistemas *realmente* grandes o de mucha complejidad. Es decir que, no alcanza dividir el problema a resolver en subproblemas cuando el sistema es muy complejo dado que, se torna difícil de visibilizar y hasta realizar las más pequeñas modificaciones del código fuente lo que hace, casi imposible replantear el sistema para agregar nuevas funcionalidades o resolver fallos.

La *POO* significa una nueva visión en la forma de programar, buscando aportar claridad y naturalidad en la manera en que se plantea un problema. Ahora, el objetivo no es identificar los procesos sino que además, se trata de identificar *actores*: las *entidades* u *objetos* que aparecen en el escenario o el dominio del problema. En este punto es importante agregar que, esos objetos no sólo tienen datos asociados sino que también, presentan comportamientos capaces de ejecutar.

Un buen ejemplo son los videojuegos. Por ejemplo, el juego Mario Bros. tiene varios personajes, Mario Bros. y Luigi.

Figura 1: Luigi y Mario Bros

Estos personajes son objetos virtuales y tienen sus propias características y comportamientos ya que, representan objetos del mundo real. Es decir que, estos personajes son capaces de saltar y caminar e interactuar con otros objetos tales como: el escenario y las monedas a medida que avanza la partida.

Ventajas de la POO

- Es la forma natural de entender la realidad y por ende, más fácil de representar/modelar.
- Permite comprender el código fuente y resolver más fácilmente problemas de gran complejidad.
- Facilita el mantenimiento y escalabilidad de las aplicaciones.
- Es más adecuado para la construcción de entornos GUI.
- Fomenta la reusabilidad y provee un gran impacto sobre la productividad y confiabilidad.

Objetos

Para comprenderlo veamos la siguiente imagen y luego intentemos responder ¿Cuáles son los objetos que se pueden abstraer para ver televisión? ¿Cómo podrías describirlos?

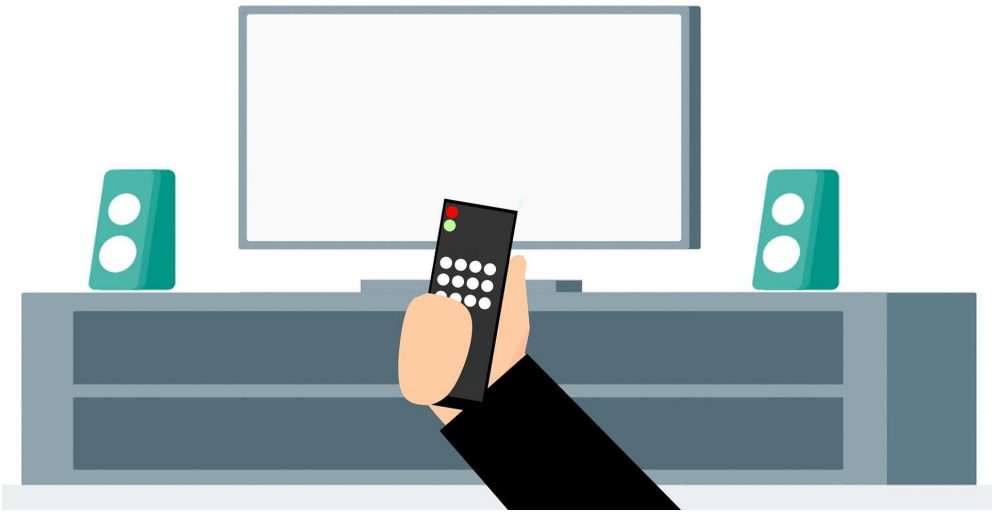

Figura 2: Ver la televisión.

En el problema planteado se pueden identificar tres elementos: la persona, el televisor y el control remoto. Cada elemento posee sus propias características y comportamientos. En la POO a estos elementos se los conoce bajo el nombre de OBJETOS, a las características o estados que identifican a cada objeto ATRIBUTOS y, a los comportamientos o acciones, MÉTODOS.

Entonces podemos resumir:

ELEMENTO	**DESCRIPCIÓN**
Persona	Tiene sus propios atributos: Apellido, Nombre, Altura, género, Color ojos, Cabello, etc. Y tiene un comportamiento: Ver , escuchar, hablar, correr, saltar, etc.
Control Remoto	Tiene sus propios atributos: Tamaño, color, tipo, batería, etc. Y tiene un comportamiento: Enviar señal, codificar señal, cambiar canal, aumentar volumen, ingresar a menú, prender TV, etc.
Televisor	Tiene sus propios atributos: pulgadas, tipo, número parlantes, marca , etc. Y tiene un comportamiento: Decodificar señal, prender, apagar, emitir señal, emitir audio, etc.

Del ejemplo anterior podemos inferir el concepto de objeto:

Un objeto es un elemento que contiene un estado (atributos) y un comportamiento

(métodos) que realiza por sí solo o bien, interactuando con otros objetos.

Sin embargo, además de las características (atributos) y acciones (métodos)… ¿Qué otras características podrías mencionar? Observa la siguiente imagen para analizar..

Figura 4: Televisor

De acuerdo a la imagen anterior podemos decir que un objeto además, se identifica por un nombre o identificador único que lo diferencia de los demás (en este caso puede ser el nro de serie), un estado (encendido, apagado), un tipo (televisor), nos abstrae dejándonos acceder sólo a las funciones encendido, apagado, cambiar canal, etc. y, por supuesto tiene un tiempo de vida.

En base a lo expuesto anteriormente podemos expresar las características generales de los objetos en POO:

- Se identifican por un nombre o identificador único que los diferencia de los demás.
- Poseen estados.
- Poseen un conjunto de métodos.
- Poseen un conjunto de atributos.
- Soportan el encapsulamiento (nos deja ver sólo lo necesario).
- Tienen un tiempo de vida.
- Son instancias de una clase (es de un tipo).

Para hacer eso, los lenguajes de programación orientados a objetos (como TypeScripts) usan descriptores (plantillas) de entidades conocidas como *clases*.

Clases

Una *clase* es la descripción de una entidad u objeto de forma tal que pueda usarse como plantilla para crear muchos objetos que respondan a dicha descripción. Para establecer analogías, se puede pensar que una clase se corresponde con el concepto de *tipo de dato* de la programación estructurada tradicional, y los objetos creados a partir de la clase (llamados *instancias* en el mundo de la *POO*) se corresponden con el concepto de *variable* de la programación tradicional. Así como el *tipo* es uno solo y describe la forma que tienen todas las muchas *variables* de ese tipo, la *clase* es única y describe la forma y el comportamiento de los muchos *objetos* de esa clase.

Para describir objetos que responden a las mismas características de forma y comportamiento, se definen las *clases*.

Veamos el siguiente ejemplo:

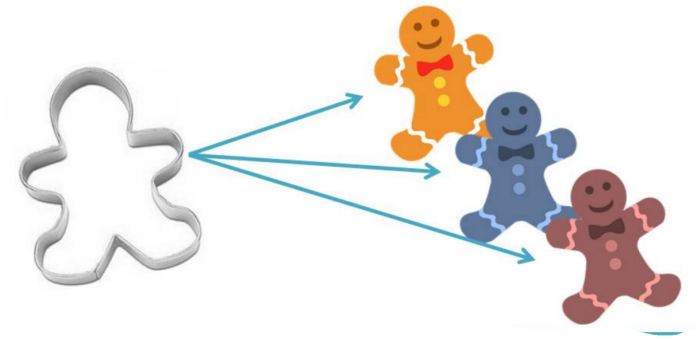

Figura 5: Molde de galletitas
Fuente: https://platzi.com/clases/1629-java-oop/21578-abstraccion-que-es-una-clase/

En el mismo podemos observar un molde para hacer galletitas (la clase) y el resultado que consiste en una o más galletas (objeto o instancia de clase).

Características generales de las clases en POO.

- Poseen un alto nivel de abstracción.
- Se relacionan entre sí mediante jerarquías.
- Los nombres de las clases deben estar en singular.

Representación gráfica de clases

La representación gráfica de una o varias clases se realiza mediante los denominados Diagramas de Clase. Para ello, se utiliza la notación que provee el Lenguaje de Modelación Unificado (UML, ver **www.omg.org**), a saber:

- Las clases se denotan como rectángulos divididos en tres partes. La primera contiene el nombre de la clase, la segunda contiene los atributos y la tercera los

métodos.

- Los modificadores de acceso a datos y operaciones, a saber: público, protegido y privado; se representan con los símbolos +, # y – respectivamente, al lado derecho del atributo. (+ público, # protegido, - privado).

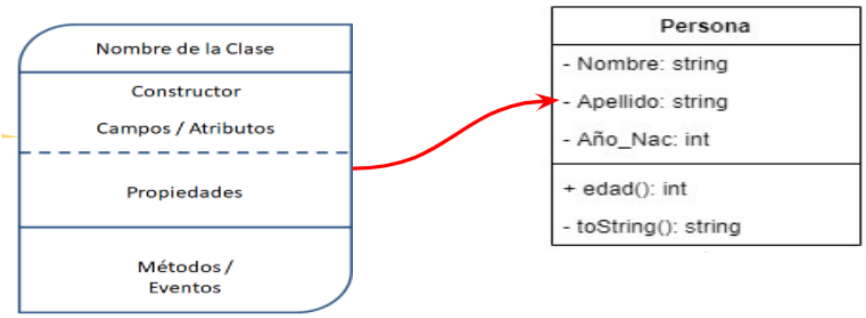

Figura 6: Estructura de una clase (Diagrama de clases)

Relación entre clases y objetos

Algorítmicamente, podemos entender a las clases como PLANTILLAS que describen objetos. Su objeto es definir nuevos tipos conformados por atributos y operaciones. Es decir que, las clases son una especie de molde de fábrica, en base al cual son construidos los objetos.

Por su parte, los objetos son instancias individuales de una clase. Durante la ejecución de un programa sólo existen los objetos, no clases.

A continuación enumeramos algunos puntos a saber:

- La declaración de una variable de una clase NO crea el objeto.
- La creación de un objeto, debe ser indicada explícitamente por el programador (instanciación). Es decir, así como inicializamos las variables con un valor, deberemos iniciar los objetos sólo que, para los objetos lo realizaremos a través de un método especial. El CONSTRUCTOR.

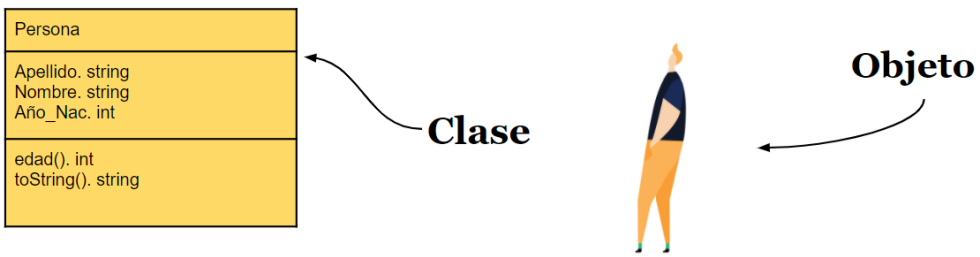

Figura 7: Relación entre clases y objetos.

Referencias

Apuntes Programa Clip - Felipe Steffolani

https://www.freecodecamp.org/

Laura Álvarez, Helmer Avendaño, Yeison García, Sebastián Morales,
Edwin Bohórquez, Santiago Hernandez, Sebastián Moreno, Cristian Orjuela.
Programación Orientada a Objetos.
https://ferestrepoca.github.io/paradigmas-de-programacion/poo/poo_teoria/concepts.html

https://www.typescriptlang.org/

https://barcelonageeks.com/composicion-de-funciones-en-python/

111Mil. Módulo Programación Orientada a Objetos.
https://github.com/111milprogramadores/apuntes/blob/master/Programacion%20Orientada%20a%20Objetos/Apunte%20Teorico%20de%20Programacion%20OO.pdf