Características más relevantes de la POO

Alejandro Héctor González

Las características más relevantes de la POO son:

Abstracción

La abstracción consiste en aislar un elemento de su contexto o del resto de los elementos que lo acompañan y responder a la pregunta: "¿qué hace ese elemento?". El proceso de abstracción permite seleccionar las características relevantes dentro de un conjunto e identificar comportamientos comunes para definir nuevos tipos de entidades en el mundo real. Puntualmente cuando uno define un objeto, la abstracción permite denotar las características esenciales del mismo, donde se capturan sus comportamientos. Cada objeto en el sistema sirve como modelo de un "agente" abstracto que puede realizar trabajo, informar y cambiar su estado, y "comunicarse" con otros objetos en el sistema sin revelar "cómo" se implementan estas características.

Encapsulamiento

Se denomina encapsulamiento al ocultamiento del estado, es decir, en el caso de la POO de los datos miembro de un objeto de manera que solo se pueda cambiar mediante las operaciones definidas para ese objeto. Esto permite aumentar la cohesión de los componentes del sistema (grado en que los elementos de un módulo permanecen juntos; mide la fuerza de la relación entre las piezas de funcionalidad dentro de un módulo dado).

Modularidad

Se denomina "modularidad" a la propiedad que permite subdividir una aplicación en partes más pequeñas llamadas módulos, cada una de las cuales debe ser tan independiente como sea posible de la aplicación en sí y de las restantes partes. Estos módulos se pueden compilar por separado, pero tienen conexiones con otros módulos.

Ocultamiento

Cada objeto está aislado del exterior, es un módulo natural, y cada tipo de objeto expone una "interfaz" que especifica a otros objetos cómo pueden interactuar con los objetos de la clase. 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. Esto asegura que otros objetos no puedan cambiar el estado interno de un objeto de manera inesperada, eliminando efectos secundarios e interacciones inesperadas. En POO hace referencia a que los atributos privados de un objeto no pueden ser modificados ni obtenidos a no ser que se haga a través del paso de un mensaje (invocación a métodos).

ventaja

Herencia

La herencia es uno de los mecanismos de los lenguajes de programación orientada a objetos basados en clases, por medio del cual una clase se deriva de otra de manera que extiende su funcionalidad. La clase de la que se hereda se suele denominar clase base, clase padre, superclase.

Las clases no se encuentran aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen.

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.
O, dicho de otro modo, las referencias y las colecciones de objetos pueden contener objetos de diferentes tipos, y la invocación de un comportamiento en una referencia producirá el comportamiento correcto para el tipo real del objeto referenciado. Cuando esto ocurre en "tiempo de ejecución", esta última característica se llama asignación dinámica.

Recolección de basura

La recolección de basura (garbage collection) es la técnica por la cual el entorno de objetos se encarga de destruir automáticamente, y por tanto desvincular la memoria asociada, los objetos que hayan quedado sin ninguna referencia a ellos. Esto significa que el programador no debe preocuparse por la asignación o liberación de memoria, ya que el entorno la asignará al crear un nuevo objeto y la liberará cuando nadie lo esté usando.
En la mayoría de los lenguajes híbridos que se extendieron para soportar el Paradigma de Programación Orientada a Objetos como C++ u Object Pascal esta característica no existe y la memoria debe liberarse expresamente.

Método estático

Así como una clase puede tener atributos estáticos, Java también permite definir métodos estáticos que se crean independientemente a la definición de objetos.

Un método estático puede llamarse sin tener que crear un objeto de dicha clase.

Igual que los atributos estáticos, un método estático tiene ciertas restricciones:

- No puede acceder a los atributos de la clase (salvo que sean estáticos)
- No puede utilizar el operador this, ya que este método se puede llamar sin tener que crear un objeto de la clase.
- Puede llamar a otro método siempre y cuando sea estático.

Si recordamos cada vez que creamos un programa en Java debemos especificar el método main:

*public **static** void main(String[] args)*

El método main es estático para que la máquina virtual de Java pueda llamarlo directamente sin tener que crear un objeto de la clase que lo contiene.

Bibliografía

Barnes, David; Kölling, Michael (2013). *Programación orientada a objetos con JAVA usando BlueJ*. Editorial Prentice Hall .

Blasco Francisco (2019) .Programación Orientada a Objetos en JAVA. Editorial Ra-Ma. España
Booch Grady. (1996). *Análisis y Diseño Orientado a Objetos con Aplicaciones*. Editorial Addison Wesley.

Llobet Azpitarte Rafael, et al (2009). *Introducción a la Programación Orientada a Objetos con Java.* Primera edición.Rev. 1.0.1.. Departamento de Sistemas Informáticos y Computación. Universidad Politécnica de Valencia. ISBN: 978-84-613-0411-0

Lopez Roman Leobardo (2011). *Programación estructurada y orientada a objetos*. Editorial: Alfa Omega Grupo editor.

Montero Roberto Miguel (2017). *Java 9 (Guías Prácticas)* Editorial Anaya Publicaciones Generales. España. ISBN 10: 844153943X / ISBN 13: 9788441539433.

Pagés Mariano (2018). *El Paradigma de Objetos a tu Alcance: Aprendiendo a resolver problemas, pensando en objetos*. Ebook Kindle.

Román Martínez, Elda Quiroga (2002). *Estructuras de datos: referencia práctica con orientación a objetos***.** Editorial Thomson