Programación orientada a objetos

La **programación orientada a objetos** o **POO** (**OOP** según sus siglas en inglés) es un paradigma de programación que usa objetos y sus interacciones, para diseñar aplicaciones y programas informáticos. Está basado en varias técnicas, incluyendo herencia, abstracción, polimorfismo y encapsulamiento. Su uso se popularizó a principios de la década de los años 1990. En la actualidad, existe variedad de lenguajes de programación que soportan la orientación a objetos.

Introducción

Los objetos son entidades que tienen un determinado *estado*, *comportamiento (método)* e *identidad*:

- El *estado* está compuesto de datos o informaciones , será uno o varios atributos a los que se habrán asignado unos valores concretos (datos).
- El *comportamiento* está definido por los métodos o mensajes a los que sabe responder dicho objeto, es decir, qué operaciones se pueden realizar con él.
- La *identidad* es una propiedad de un objeto que lo diferencia del resto, dicho con otras palabras, es su identificador (concepto análogo al de identificador de una variable o una constante).

Un objeto contiene toda la información que permite definirlo e identificarlo frente a otros objetos pertenecientes a otras clases e incluso frente a objetos de una misma clase, al poder tener valores bien diferenciados en sus atributos. A su vez, los objetos disponen de mecanismos de interacción llamados métodos, que favorecen la comunicación entre ellos. Esta comunicación favorece a su vez el cambio de estado en los propios objetos. Esta característica lleva a tratarlos como unidades indivisibles, en las que no se separa el estado y el comportamiento.

Los **métodos (comportamiento)** y **atributos (estado)** están estrechamente relacionados por la propiedad de conjunto. Esta propiedad destaca que una clase requiere de métodos para poder tratar los atributos con los que cuenta. El programador debe pensar indistintamente en ambos conceptos, sin separar ni darle mayor importancia a alguno de ellos. Hacerlo podría producir el hábito erróneo de crear clases contenedoras de información por un lado y clases con métodos que manejen a las primeras por el otro. De esta manera se estaría realizando una **programación estructurada camuflada** en un lenguaje de programación orientado a objetos.

La POO difiere 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. La programación estructurada anima al programador a pensar sobre todo en términos de procedimientos o funciones, y en segundo lugar en las estructuras de datos que esos procedimientos manejan. En la programación estructurada solo se escriben funciones que procesan datos. Los programadores que emplean POO, en cambio, primero definen objetos para luego enviarles mensajes solicitándoles que realicen sus métodos por sí mismos.

Origen

Los conceptos de la programación orientada a objetos tienen origen en Simula 67, un lenguaje diseñado para hacer simulaciones, creado por Ole-Johan Dahl y Kristen Nygaard del Centro de Cómputo Noruego en Oslo. En este centro, se trabajaba en simulaciones de naves, que fueron confundidas por la explosión combinatoria de cómo las diversas cualidades de diferentes naves podían afectar unas a las otras. La idea surgió al agrupar los diversos tipos de naves en diversas clases de objetos, siendo responsable cada clase de objetos de definir sus *propios* datos y comportamientos. Fueron refinados más tarde en Smalltalk, desarrollado en Simula en Xerox PARC (cuya primera versión fue escrita sobre Basic) pero diseñado para ser un sistema completamente dinámico en el cual los objetos se podrían crear y modificar "sobre la marcha" (en tiempo de ejecución) en lugar de tener un sistema basado en programas estáticos.

La programación orientada a objetos se fue convirtiendo en el estilo de programación dominante a mediados de los años ochenta, en gran parte debido a la influencia de C++, una extensión del lenguaje de programación C. Su dominación fue consolidada gracias al auge de las Interfaces gráficas de usuario, para las cuales la programación orientada a objetos está particularmente bien adaptada. En este caso, se habla también de programación dirigida por eventos.

Las características de orientación a objetos fueron agregadas a muchos lenguajes existentes durante ese tiempo, incluyendo Ada, BASIC, Lisp, Pascal, entre otros. La adición de estas características a los lenguajes que no fueron diseñados inicialmente para ellas condujo a menudo a problemas de compatibilidad y en la capacidad de mantenimiento del código. Los lenguajes orientados a objetos "puros", por su parte, carecían de las características de las cuales muchos programadores habían venido a depender. Para saltar este obstáculo, se hicieron muchas tentativas para crear nuevos lenguajes basados en métodos orientados a objetos, pero permitiendo algunas características imperativas de maneras "seguras". El Eiffel de Bertrand Meyer fue un temprano y moderadamente acertado lenguaje con esos objetivos pero ahora ha sido esencialmente reemplazado por Java, en gran parte debido a la aparición de Internet, y a la implementación de la máquina virtual de Java en la mayoría de navegadores. PHP en su versión 5 se ha modificado, soporta una orientación completa a objetos, cumpliendo todas las características propias de la orientación a objetos.

Conceptos fundamentales

La programación orientada a objetos es una forma de programar que trata de encontrar una solución a estos problemas. Introduce nuevos conceptos, que superan y amplían conceptos antiguos ya conocidos. Entre ellos destacan los siguientes:

- **Clase**: definiciones de las propiedades y comportamiento de un tipo de objeto concreto. La instanciación es la lectura de estas definiciones y la creación de un objeto a partir de ellas.
- **Herencia:** (por ejemplo, herencia de la clase C a la clase D) Es la facilidad mediante la cual la clase D hereda en ella cada uno de los atributos y operaciones de C, como si esos atributos y operaciones hubiesen sido definidos por la misma D. Por lo tanto, puede usar los mismos métodos y variables publicas declaradas en C. Los componentes registrados como "privados" (private) también se heredan, pero como no pertenecen a la clase, se mantienen escondidos al programador y sólo pueden ser accedidos a través de otros métodos públicos. Esto es así para mantener hegemónico el ideal de OOP.
- **Objeto**: entidad provista de un conjunto de propiedades o atributos (datos) y de comportamiento o funcionalidad (métodos) los mismos que consecuentemente reaccionan a eventos. Se corresponde con los objetos reales del mundo que nos rodea, o a objetos internos del sistema (del programa). Es una instancia a una clase.
- **Método**: Algoritmo asociado a un objeto (o a una clase de objetos), cuya ejecución se desencadena tras la recepción de un "mensaje". Desde el punto de vista del comportamiento, es lo que el objeto puede hacer. Un método puede producir un cambio en las propiedades del objeto, o la generación de un "evento" con un nuevo mensaje para otro objeto del sistema.
- **Evento**: Es un suceso en el sistema (tal como una interacción del usuario con la máquina, o un mensaje enviado por un objeto). El sistema maneja el evento enviando el mensaje adecuado al objeto pertinente. También se puede definir como evento, a la reacción que puede desencadenar un objeto, es decir la acción que genera.
- **Mensaje**: una comunicación dirigida a un objeto, que le ordena que ejecute uno de sus métodos con ciertos parámetros asociados al evento que lo generó.
- **Propiedad o atributo**: contenedor de un tipo de datos asociados a un objeto (o a una clase de objetos), que hace los datos visibles desde fuera del objeto y esto se define como sus características predeterminadas, y cuyo valor puede ser alterado por la ejecución de algún método.
- **Estado interno**: es una variable que se declara privada, que puede ser únicamente accedida y alterada por un método del objeto, y que se utiliza para indicar distintas situaciones posibles para el objeto (o clase de objetos).

No es visible al programador que maneja una instancia de la clase.
- **Componentes de un objeto**: atributos, identidad, relaciones y métodos.
- **Identificación de un objeto**: un objeto se representa por medio de una tabla o entidad que esté compuesta por sus atributos y funciones correspondientes.

En comparación con un lenguaje imperativo, una "variable", no es más que un contenedor interno del atributo del objeto o de un estado interno, así como la "función" es un procedimiento interno del método del objeto.

Características de la POO

Existe un acuerdo acerca de qué características contempla la "orientación a objetos", las características siguientes son las más importantes:

- **Abstracción**: denota las características esenciales de un objeto, 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. Los procesos, las funciones o los métodos pueden también ser abstraídos y cuando lo están, una variedad de técnicas son requeridas para ampliar una abstracción.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. La abstracción es clave en el proceso de análisis y diseño orientado a objetos, ya que mediante ella podemos llegar a armar un conjunto de clases que permitan modelar la realidad o el problema que se quiere atacar.
- **Encapsulamiento**: Significa reunir a 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. Algunos autores confunden este concepto con el principio de ocultación, principalmente porque se suelen emplear conjuntamente.
- **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. Al igual que la encapsulación, los lenguajes soportan la Modularidad de diversas formas.
- **Principio de ocultación**: Cada objeto está aislado del exterior, es un módulo natural, y cada tipo de objeto expone una *interfaz* a otros objetos que especifica 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 pueden cambiar el estado interno de un objeto de maneras inesperadas, eliminando efectos secundarios e interacciones inesperadas. Algunos lenguajes relajan esto, permitiendo un acceso directo a los datos internos del objeto de una manera controlada y limitando el grado de abstracción. La aplicación entera se reduce a un agregado o rompecabezas de objetos.
- **Polimorfismo**: comportamientos diferentes, asociados a objetos distintos, pueden compartir el mismo nombre, al llamarlos por ese nombre se utilizará el comportamiento correspondiente al objeto que se esté usando. 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 tardía* o *asignación dinámica*. Algunos lenguajes proporcionan medios más estáticos (en "tiempo de compilación") de polimorfismo, tales como las plantillas y la sobrecarga de operadores de C++.
- **Herencia**: las clases no están 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. 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. Esto suele hacerse habitualmente agrupando los objetos en *clases* y estas en *árboles* o *enrejados* que reflejan un comportamiento común. Cuando un objeto hereda de más de una clase se dice que hay *herencia múltiple*.

- **Recolección de basura**: la recolección de basura o *garbage collector* 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 desasignarse manualmente.

Resumen

La programación orientada a objetos es un paradigma que utiliza objetos como elementos fundamentales en la construcción de la solución. Surge en los años 70. Un objeto es una abstracción de algún hecho o ente del mundo real que tiene atributos que representan sus características o propiedades y métodos que representan su comportamiento o acciones que realizan. Todas las propiedades y métodos comunes a los objetos se encapsulan o se agrupan en clases. Una clase es una plantilla o un prototipo para crear objetos, por eso se dice que los objetos son instancias de clases.

Lenguajes orientados a objetos

Simula (1967) es aceptado como el primer lenguaje que posee las características principales de un lenguaje orientado a objetos. Fue creado para hacer programas de simulación, en donde los "objetos" son la representación de la información más importante. Smalltalk (1972 a 1980) es posiblemente el ejemplo canónico, y con el que gran parte de la teoría de la programación orientada a objetos se ha desarrollado.

Entre los lenguajes orientados a objetos se destacan los siguientes:

- ABAP -> SAP Lenguaje orientado a eventos
- ABL Lenguaje de programación de OpenEdge de Progress Software
- ActionScript
- ActionScript 3
- Ada
- C++
- C#
- Clarion
- Clipper (lenguaje de programación) (Versión 5.x con librería de objetos Class(y))
- D
- Object Pascal (Embarcadero Delphi)
- Gambas
- Harbour
- Eiffel
- Fortran 90/95
- Java
- JavaScript (la herencia se realiza por medio de la programación basada en prototipos)
- Lexico (en castellano)
- Objective-C
- Ocaml
- Oz
- R

- Perl (soporta herencia múltiple. La resolución se realiza en preorden, pero puede modificarse al algoritmo linearization C3 por medio del módulo Class::C3 [1] en CPAN)
- PHP (a partir de su versión 5)
- PowerBuilder
- Python
- Ruby
- Smalltalk (Entorno de objetos puro)
- Magik (SmallWorld)
- Vala
- VB.NET
- Visual FoxPro (en su versión 6)
- Visual Basic 6.0
- Visual DataFlex
- Visual Objects
- XBase++
- Lenguaje DRP
- Lenguaje de programación Scala (lenguaje usado por Twitter) http://www.scala-lang.org/page.jsp

Muchos de estos lenguajes de programación no son puramente orientados a objetos, sino que son híbridos que combinan la POO con otros paradigmas.

Al igual que C++ otros lenguajes, como OOCOBOL, OOLISP, OOPROLOG y Object REXX, han sido creados añadiendo extensiones orientadas a objetos a un lenguaje de programación clásico.

Un nuevo paso en la abstracción de paradigmas de programación es la Programación Orientada a Aspectos (POA). Aunque es todavía una metodología en estado de maduración, cada vez atrae a más investigadores e incluso proyectos comerciales en todo el mundo.

Enlaces externos

- Qué es la programacion orientada a objetos [2]

Referencias

[1] http://search.cpan.org/perldoc?Class::C3
[2] http://www.desarrolloweb.com/articulos/499.php

Fuentes y contribuyentes del artículo

Programación orientada a objetos *Fuente*: http://es.wikipedia.org/w/index.php?oldid=57438432 *Contribuyentes*: Abajo estaba el pez, Airunp, Akira 999, Aleator, Alejolp, Alhen, Alvaro qc, Andre Engels, Andreasmperu, Angel GN, Angus, Anonimato1990, Anthemfor182, Antur, Ascánder, AstroNomo, Barcex, Biasoli, Bitman28, Byj2000, Camilo, Caravena, Carmin, Carocbax, Carok, Chfiguer, Comae, Cosmox, Cratón, Crescent Moon, David0811, Dem, Developer, Diegusjaimes, Dj Merlin, Dodo, Dreitmen, ESTEBAN ESPINOZA, Edgar, Edmenb, Eduardosalg, Edwardcetera, Egaida, El Padrino, El mago de la Wiki, Elwikipedista, Emiduronte, Erfil, FJJW, Farisori, Foundling, Funflin, Furti, GermanX, HUB, Hprmedina, Humberto, Icvav, Identy, Ing moi, Isha, Ivan.Romero, J053d, JMPerez, JavierCantero, Jecanre, Jesuja, Jkbw, Jmvkrecords, JoaquinFerrero, Josuemb, Jstitch, Kavanagh, Kved, Lasneyx, Lentucky, Leonardo Tadei, Leonpolanco, Locos epraix, Lopezpablo 87, Maaksz2, MadriCR, Mahadeva, Manawo, Mansoncc, ManuelGR, Manwë, MarcoAurelio, Mariano12 1989, Marsal20, Matdrodes, MatiasBellone, Milyka, Moriel, Mpeinadopa, Muro de Aguas, Murven, Nesita02, Netito777, Nicop, Niqueco, Nulain, Otermin, Otherox, Pablo323, PabloCastellano, Pan con queso, Pedro Felipe, Pedrovicenterosero, Penavarro09, Penguino, Poco a poco, Prision3ro, Pólux, Queninosta, Raul2010, Raulshc, Raymonddoyle, Rosarino, Sauron, SergioVares, Serser, Sirpuppet, Soynatan, Spc, Spirit-Black-Wikipedista, SuperBraulio13, SuperJoe, Superbenja, Superzerocool, Tano4595, Technopat, Tirithel, Tokvo, Tomatejc, Tonchizerodos, Tosin2627, Tute, Txo, Vitamine, Vivero, Vubo, Wilfredor, Wrdhat, X.Cyclop, Xsm34, Yeza, ZeruGiran, conversion script, host-200-76-49-134.block.alestra.net.mx, Ál, 714 ediciones anónimas

Licencia