Programación Orientada a Objetos (POO)

Universidad Tecnológica de Pereira

Programación 4

2023

¿Qué es la POO?

- El término **Programación Orientada a Objetos (POO)**, hoy en día ampliamente utilizado, es difícil de definir, ya que no es un concepto nuevo, sino que ha sido el desarrollo de técnicas de programación desde principios de la década de los setenta, aunque sea en la década de los noventa cuando ha aumentado su difusión, uso y popularidad. No obstante, se puede definir POO como una técnica o estilo de programación que utiliza objetos como bloque esencial de construcción.
- La **Programación Orientada a Objetos**, permite descomponer mas fácilmente un problema en subgrupos de partes relacionadas del problema. Entonces, utilizando el lenguaje se pueden traducir estos subgrupos a unidades autocontenidas llamadas objetos.

¿Qué es la POO?

- **La Programación Orientada a Objetos (POO)** es una metodología de diseño de software y un paradigma de programación que define los programas en términos de "clases de objetos", objetos que son entidades que combinan estado (es decir, datos) y comportamiento (esto es, procedimientos o métodos).
- **La programación orientada a objetos** es un programa con un conjunto de objetos, que se comunican entre ellos para realizar tareas y que es un modelo que representa un subconjunto del mundo real, tal fielmente como sea posible, de modo fácil y natural, donde los objetos van a tener características (atributos) y comportamientos (métodos). Que a diferencia de los lenguajes procedurales, en donde los datos y los procedimientos se encuentran separados y sin relación alguna.

¿Orígenes de la POO?

- **Etapa 1**. Lenguajes Ensambladores. La unidad de programación es la instrucción, compuesta de un operador y los operandos. El nivel de abstracción que se aplica es muy bajo.
- **Etapa 2**. Lenguajes de Programación: Fortran, Algol, Cobol. Los objetos y operaciones del mundo real se podían modelar mediante datos y estructuras de control separadamente. En esta etapa el diseño del software se enfoca sobre la representación del detalle procedimental y en función del lenguaje elegido. Conceptos como: refinamiento progresivo, modularidad, procedimientos y programación estructurada son conceptos básicos que se utilizan en esta etapa. Existe mayor abstracción de datos.

¿Orígenes de la POO?

- **Etapa 3.** Se introducen en esta etapa los conceptos de abstracción y ocultación de la información.
- **Etapa 4**. A partir de los años setenta se trabaja sobre una nueva clase de lenguajes de simulación y sobre la construcción de prototipos tales como Simula-70 y basado en parte de éste, el Smalltalk. En estos lenguajes, la abstracción de datos tiene una gran importancia y los problemas del mundo real se representan mediante objetos de datos a los cuales se les añade el correspondiente conjunto de operaciones asociados a ellos. Términos como Abstracción de datos, objeto, encapsulación entre otros, son conceptos básicos sobre la que se fundamenta la POO.

¿Para qué la POO?

- Para disminuir los errores y promociona la reutilización del código.
- Con el paradigma de POO lo que se busca es dejar de centrarse en la lógica pura de los programas, para comenzar a pensar en objetos, lo que forma la base de dicho paradigma. Esto ayuda bastante en sistemas grandes, pues en lugar de pensar en funciones, se piensa en las relaciones o interacciones de los distintos elementos del sistema.

¿Por qué la POO?

- Porque permite que el código sea reutilizable, organizado y sencillo de mantener. Y es que sigue el principio de desarrollo de software empleado por muchos programadores DRY (Don't Repeat Yourself), para no duplicar el código y crear programas eficientes. Asimismo, previene el acceso no deseado a los datos o la exposición de código propietario a través de la encapsulación y la abstracción.
- Convierte cosas complejas en estructuras simples reproducibles.
- Permite trabajar en equipo gracias al encapsulamiento, puesto que minimiza la posibilidad de duplicar funciones cuando distintas personas trabajan sobre un mismo objeto al mismo tiempo.
- Al estar la clase bien estructurada permite la corrección de errores en diversos lugares del código.
- Protege la información mediante la encapsulación, pues solo se puede acceder a los datos del objeto mediante propiedades y métodos privados.
- La abstracción nos permite construir sistemas más complejos y de un modo más sencillo y organizado.

Metodologías de POO

- Se han desarrollado metodologías, que tienen como una de sus funciones el lograr una mayor productividad en el desarrollo de los sistemas de información. Existen diferentes metodologías las cuales tienen diferentes fases y son precisamente dos de ellas en las que se enfatiza y son el Análisis y el Diseño:
  - OMT que es la Técnica del Modelado de Objetos, el cual a grandes rasgos cuenta con las siguientes cuatro fases: Análisis, Diseño del Sistema, Diseño de Objetos e Implementación.
  - UML más recientemente, el cual es un Lenguaje Unificado de Modelado y es una representación gráfica que sirve para modelar sistemas orientados a objetos, ya que permite manejar los elementos descritos en los apartados anteriores (por ejemplo los mensajes entre objetos, sincronización , etc.). Entre sus principales características se encuentran: su flexibilidad, cuenta con muchos elementos gráficos.

Lenguajes Orientados a Objetos

- Históricamente han ido surgiendo distintos paradigmas de programación. Por un lado, los lenguajes secuenciales como COBOL o procedimentales como Basic o C, se enfocan más en la lógica que en los datos. Por otro lado, otros más modernos como Java, C# y Python, usan paradigmas para definir los programas, siendo la POO la más popular.

Lenguajes Orientados a Objetos

- Entre los lenguajes orientados a objetos destacan los siguientes:
  - Smalltalk
  - Objective-C
  - C++
  - Ada 95
  - Java
  - Ocaml
  - Python
  - Delphi
  - Lexico (en castellano)
  - C#
  - Eiffel
  - Ruby
  - ActionScript
  - Visual Basic
  - PHP
  - PowerBuilder
  - Clarion
  - Simula67

Un poco de historia de la POO

- La Programación Orientación a Objetos (P.O.O.) surge en Noruega en 1967 con un lenguaje llamado Simula 67, desarrollado por Krinsten Nygaard y Ole-Johan Dahl, en el centro de cálculo noruego.
- Simula 67 introdujo por primera vez los conceptos de clases, corrutinas y subclases (conceptos muy similares a los lenguajes Orientados a Objetos de hoy en día).
- Uno de los problemas de inicio de los años setentas era que pocos sistemas lograban terminarse, pocos se terminaban con los requisitos iniciales y no todos los que se terminaban cumpliendo con los requerimientos se usaban según lo planificado. El problema consistía en cómo adaptar el software a nuevos requerimientos imposibles de haber sido planificados inicialmente.

Un poco de historia de la POO

- En los 70's científicos del centro de investigación en Palo Alto Xerox inventaron el lenguaje Small talk que fue el primer lenguaje Orientado a Objetos puro de los lenguajes Orientados a Objetos, es decir, únicamente utiliza clases y objetos.
- En los años 80's Bjarne Stroustrup de AT&T Labs., amplió el lenguaje C para crear C++ que soporta la programación Orientada a Objetos. En esta misma década se desarrollaron otros lenguajes Orientados a Objetos como Objective C, Common Lisp Object System (CIOS), object Pascal, Ada y otros.

Un poco de historia de la POO

- En el inicio de los 90's se consolida la Orientación a Objetos como una de las mejores maneras para resolver problemas. Aumenta la necesidad de generar prototipos más rápidamente (concepto RAD Rapid Aplication Developments). Sin esperar a que los requerimientos iniciales estén totalmente precisos.
- En 1996 surge un desarrollo llamado JAVA (extensión de C++). Su filosofía es aprovechar el software existente. Facilitar la adaptación del mismo a otros usos diferentes a los originales sin necesidad de modificar el código ya existente.
- En 1997-98 se desarrollan herramientas 'CASE' orientadas a objetos (como el diseño asistido por computadora).
- Actualmente la orientación a objetos parece ser el mejor paradigma, no obstante, no es una solución a todos los problemas. Trata de eliminar la crisis del software.

Desventajas de POO

- Pesar de que las ventajas de la programación orientada a objetos superan a las limitaciones de la misma, podemos encontrar algunas características no deseables en ésta.
- Limitaciones para el programador. No obstante que la tecnología orientada a objetos no es nueva, un gran porcentaje de programadores no están familiarizados con los conceptos de dicha tecnología. En otras palabras, la lógica de la programación estructurada sigue siendo predominante en la mayoría de los desarrolladores de software, después de haber revisado de forma breve los principios de la programación orientada a objetos.

Desventajas de POO

- Tamaño excesivo en las aplicaciones resultantes. La gran mayoría de los equipos de computo cuentan con capacidades tanto de almacenamiento como de memoria lo suficientemente buena como para ejecutar la mayoría de las aplicaciones que puedan desarrollarse con la tecnología orientada a objetos, sin embargo existen casos en los que lo anterior no se cumple. Una de las desventajas de la programación orientada a objetos es que cuando se heredan clases a partir de clases existentes se heredan de forma implícita todos los miembros de dicha clase aun cuando no todos se necesiten, lo que produce aplicaciones muy grandes que no siempre encajan en los sistemas con los que se disponga.
- Velocidad de ejecución: Esto tiene que ver, en cierto modo, con el punto anterior, una aplicación innecesariamente pesada en muchas ocasiones es más lenta de ejecutar que una aplicación conformada únicamente por los módulos necesarios.

C++

- El lenguaje de programación c++ fue el lenguaje que ayudó a potenciar la programación orientada a objetos, en una época dónde dominada la programación estructurada.

C++

- El lenguaje C++ fue creado por Bjarne Stroustrup, en el año 1979. Este programador danés empezó a trabajar en el lenguaje, que lo llamaba C con clases.
- Stroustrup comparó varios lenguajes para la creación del predecesor de C++. Por ejemplo, vio que Simula (un lenguaje POO del 1962), tenía buenas características para programar, pero era muy lento, y el lenguaje BCPL era rápido, pero de bajo nivel.
- Así que mejoró el lenguaje C con características de Simula. Se decidió por el lenguaje C porque:
  - es de uso general.
  - es rápido.
  - es portable.
  - es muy utilizado.

C++

- Aunque el lenguaje C++ se creó en 1979, tal y como hemos dicho, no es hasta 1983 que tiene su nombre definitivo, pasando de C con clases a C++. El nombre fue propuesto por Rick Mascitti, cuando se utilizó por primera vez fuera de un laboratorio científico.
- En ese tiempo también se añadieron nuevas características, como la herencia, la sobrecarga de funciones, y las funciones virtuales.
- C++ 2.0 se lanzó en 1989, con nuevas opciones como herencia múltiple (los lenguajes actuales de programación orientada a objetos solo permite una única herencia), clases abstractas, funciones estáticas y muchas más.
- Con C++11, aprobado como estándar ISO en 2011, dio un salto cualitativo gracias a los cambios en librerías, aunque la versión más reciente es la de C++17 aprobado en 2017.

Python

- Guido van Rossum ideó el lenguaje Python a finales de los 80 y comenzó a implementarlo en diciembre de 1989. En febrero de 1991 publicó la primera versión pública, la versión 0.9.0. La versión 1.0 se publicó en enero de 1994, la versión 2.0 se publicó en octubre de 2000 y la versión 3.0 se publicó en diciembre de 2008.
- Hasta 2018, el desarrollo de Python estaba dirigido personalmente por Guido van Rossum y bajo el paraguas de la fundación Python Software Foundation. En julio de 2018 Guido van Rossum anunció que dejaría de dirigir el desarrollo de Python. Desde 2019 el desarrollo de Python está dirigido por un consejo de dirección de cinco miembros elegidos entre los desarrolladores de Python y que se renueva anualmente.
- Las versiones de Python se identifican por tres números X.Y.Z

Versiones de Python

- X corresponde a las grandes versiones de Python (1, 2 y 3), incompatibles entre sí:
- Los principales cambios introducidos en Python 2 fueron las cadenas Unicode, las comprensiones de listas, las asignaciones aumentadas, los nuevos métodos de cadenas y el recolector de basura para referencias cíclicas.
- Los principales cambios introducidos en Python 3 fueron la separación entre cadenas Unicode y datos binarios, la función print(), cambios en la sintaxis, tipos de datos, comparadores, etc.
- Por el momento, no hay planes de crear una nueva versión Python 4, incompatible con las anteriores.

Versiones de Python

- Y corresponde a versiones importantes en las que se introducen novedades en el lenguaje pero manteniendo la compatibilidad (salvo excepciones).
- Desde la versión 2.0 hasta 2019, las versiones X.Y se publicaron aproximadamente cada año y medio y se han mantenido durante cinco años, excepto la versión 2.7, que se mantuvo durante diez años, hasta el 1 de enero de 2020 (aunque se publicó una versión final en abril de 2020).
- En 2019 se decidió pasar a publicar nuevas versiones X.Y anualmente, en octubre, manteniéndolas durante cinco años. Así, Python 3.9, 3.10 y 3.11 se publicaron en octubre de 2020, 2021 y 2022 respectivamente y Python 3.12 se publicará en octubre de 2023.

Versiones de Python

- Z corresponde a versiones menores que se publican durante el período de mantenimiento, en las que sólo se corrigen errores durante el primer año y fallos de seguridad en los cuatro restantes.
- La publicación de las versiones .Z es más irregular, porque el descubrimiento de fallos de seguridad puede provocar la necesidad de publicación de una nueva versión en cualquier momento, pero hay un calendario previsto de versiones Z.
- En principio, en el primer año después de la publicación de la versión inicial X.Y.0, se publican versiones .Z cada dos meses, tanto en forma de código fuente como de instaladores. Tras la publicación de la siguiente versión X.(Y+1).0 se publica una versión X.Y.Z proporcionando instaladores, pero las versiones .Z que se publican hasta completar los cinco de mantenimiento ya sólo se publican en forma de código fuente. Además, normalmente se publica una última versión X.Y.Z justo antes de que una versión X.Y deje de mantenerse.

POO en Python

- Python incluye las características siguientes para dar soporte a la programación orientada a objetos:
  - Creación de objetos basada en clases. Las clases son plantillas para la creación de objetos. Los objetos son estructuras de datos con el comportamiento asociado.
  - Herencia con polimorfismo. Python da soporte a la herencia individual y múltiple. Todos los métodos de instancias de Python son polimórficos y se pueden alterar temporalmente mediante subclases.
  - Encapsulación con ocultación de datos. Python permite ocultar los atributos. Cuando se ocultan los atributos, se puede acceder a los mismos desde fuera de la clase únicamente mediante los métodos de la clase. Las clases implementan métodos para modificar los datos.

El lenguaje Java

- Java nace en 1991 con el nombre “OAK”, posteriormente cambiado por Green por problemas legales, y finalmente con la denominación actual JAVA.
- El objetivo de java era crear un lenguaje de programación parecido a C++ en estructura y sintaxis, fuertemente orientado a objetos, pero con una máquina virtual propia. Esto se hizo bajo el principio, de poder ser usado bajo cualquier arquitectura “Write Once, Run Anywhere” (escríbelo una vez, ejecútalo en cualquier sitio).
- A día de hoy, podemos decir, que Java es uno de los lenguajes más importantes del mundo. Con una comunidad extendida en todos los componentes y más de 4 millones de desarrolladores, existen millones de dispositivos que lo usan.
- Además, tras el surgimiento de Android, Java se estableció como el lenguaje de programación para móviles más extendido del planeta.

¿Qué es una maquina virtual?

- Definición técnica: la JVM (máquina virtual en java) es la especificación de un programa de software que ejecuta código y proporciona el entorno de ejecución para ese código.
- Definición cotidiana: La JVM es la forma en que ejecutamos nuestros programas Java. Configuramos la JVM y luego confiamos en ella para administrar los recursos del programa durante la ejecución.

Para qué se utiliza la JVM

- La máquina virtual en Java tiene dos funciones principales:
  1. permitir que los programas Java se ejecuten en cualquier dispositivo o sistema operativo (conocido como el principio Escribir una vez, ejecutar en cualquier lugar)
  2. Administrar y optimizar la memoria del programa.
- Cuando se lanzó Java en 1995, todos los programas de computadora se escribían para un sistema operativo específico y el desarrollador de software administraba la memoria del programa.
- Entonces la JVM fue una novedad revolucionaria.

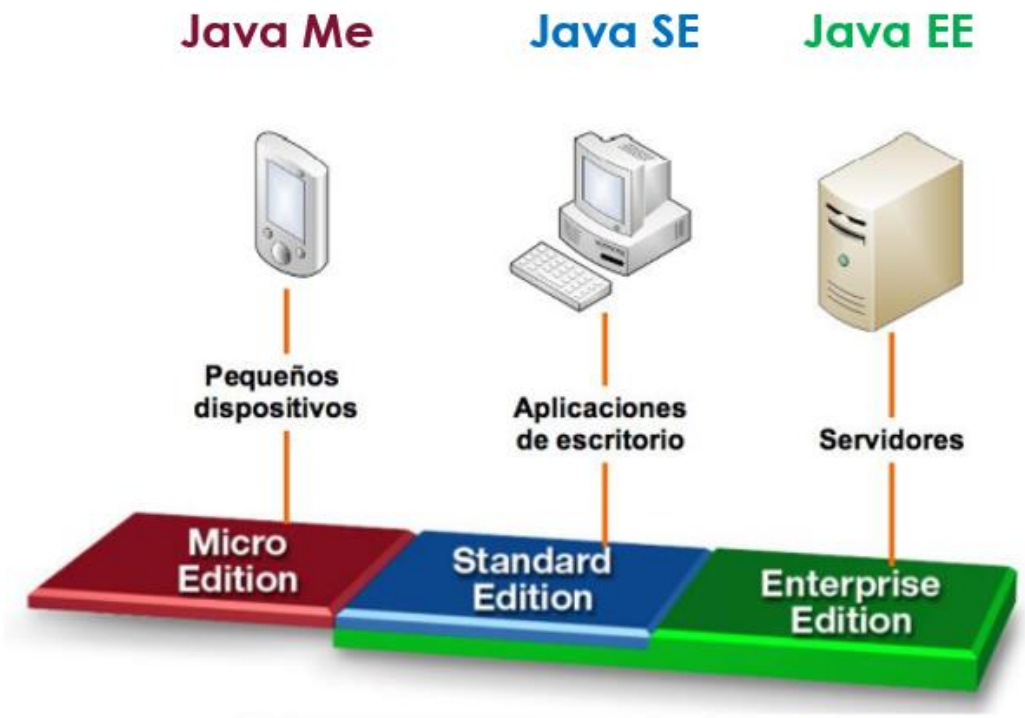
Versiones de Java

Versión	Release date	Public Updates	Support Until
JDK Beta	1995	?	?
JDK 1.0	January 1996	?	?
JDK 1.1	February 1997	?	?
J2SE 1.2	December 1998	?	?
J2SE 1.3	May 2000	?	?
J2SE 1.4	February 2002	October 2008	February 2013
J2SE 5.0	September 2004	November 2009	April 2015
Java SE 6	December 2006	April 2013	December 2018
Java SE 7	July 2011	April 2015	July 2022
Java SE 8 (LTS)	March 2014	At least May 2026 for OpenJDK	December 2030

Versiones de Java

Versión	Release date	Public Updates	Support Until
Java SE 9	September 2017	March 2018 for OpenJDK	N/A
Java SE 10	March 2018	September 2018 for OpenJDK	N/A
Java SE 11 (LTS)	September 2018	October 2024 for OpenJDK	September 2026
Java SE 12	March 2019	September 2019 for OpenJDK	N/A
Java SE 13	September 2019	March 2020 for OpenJDK	N/A
Java SE 14	March 2020	September 2020 for OpenJDK	N/A
Java SE 15	September 2020	March 2021 for OpenJDK	N/A
Java SE 16	March 2021	September 2021 for OpenJDK	N/A
Java SE 17 (LTS)	September 2021	September 2030 for Zulu	TBA

Plataformas Java



JVM, JRE, JDK

- Java Virtual Machine (JVM)
  - Es una abstracción de una máquina de cómputo, encargada de ejecutar los programas Java.
- Java Runtime Environment (JRE)
  - Es un paquete de software que contiene los artefactos requeridos para ejecutar un programa Java.
- Java Development Kit (JDK)
  - Es un superconjunto del JRE y contiene las herramientas para los programadores Java.