

Índice de contenido

Programación Orientada a Objetos

1. ¿Qué es la programación orientada a objetos?

Es un paradigma de programación que se basa en el concepto de “objetos”, los cuales son entidades que contiene campos (atributos/variables) y comportamientos (métodos/funciones) relacionados.

Los objetos son capaces de interactuar entre sí para realizar sus tareas y también pueden modificar los valores contenidos en sus atributos.

Los lenguajes de programación contienen objetos prediseñados (llamados librerías o paquetes) permitiendo que el desarrollo de software se realice de manera más sencilla, eficaz y rápida; Además, muchos de estos lenguajes permiten al usuario la creación de sus propios objetos y librerías.

Si bien los conceptos de objetos, atributos, métodos y clases pueden ser bastante confusos al principio, a lo largo de esta clase iremos abordando cada término individualmente para explicar todos los conceptos de manera más clara.

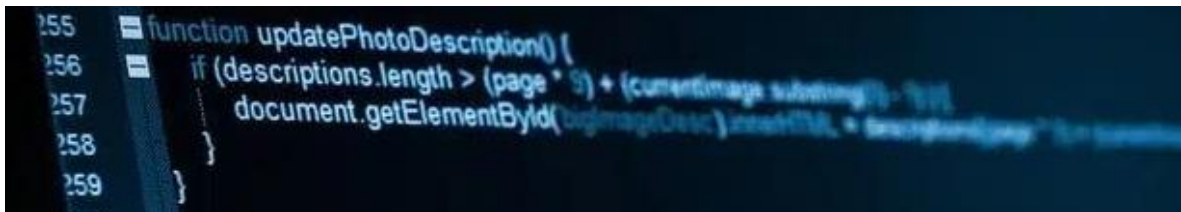
2. ¿Cómo surgió la programación orientada a objetos?

Antes de la década de los '90s el paradigma de programación más utilizado era la programación estructurada.

En la programación estructurada tradicional los datos y procedimientos están separados y sin relación, ya que su principal función es el procesamiento de unos datos de entrada para obtener otros de salida.

En la programación estructurada se emplean principalmente funciones que procesan datos. Si bien en programas pequeños y medianos esto no representa ningún inconveniente y hace el proceso de creación de software lineal y sencillo; cuando se presenta un proyecto de software muy grande (por ejemplo, un sistema bancario para un banco internacional con millones de clientes) la programación estructurada comienza a mostrar limitaciones en términos de mantenimiento, escalabilidad y reutilización de código.

En la programación estructurada de los años 80, no era estrictamente necesario que todo el código fuente estuviera en un mismo archivo, aunque era una práctica común debido a las limitaciones de las herramientas y entornos de desarrollo de la época. Imagínate todo el código fuente de un sistema bancario en un mismo archivo. Estamos hablando de miles y miles de líneas de código (si no cientos de miles) que debían ser revisadas en caso de encontrarse errores o fallas en el sistema. A esto también podemos agregarle el hecho de que se construía el software entre muchos desarrolladores que lo más probable tenían diferentes maneras de programar.



La programación orientada a objetos por su parte traía muchas ventajas a la hora de implementar software tales como:

- **Modularidad**: La programación orientada a objetos traía un enfoque más modular y flexible para abordar la complejidad de los sistemas de software, lo que hacía más adecuada para aplicaciones de mayor tamaño y complejidad.
- **Reutilización de código**: La programación orientada a objetos promueve la reutilización de código a través de conceptos como la herencia. Esto permite a los desarrolladores reutilizar componentes de software existente en vez de tener que escribir todo desde cero.
- **Abstracción**: La programación orientada a objetos se basa en el concepto de modelado del mundo real, lo que hace más fácil de entender para los programadores. Al representar entidades del mundo real utilizando atributos y métodos, la POO facilita transformar los requisitos del software a código.

Durante la década de los '80s la programación orientada a objetos se fue convirtiendo en el paradigma de programación dominante, en gran parte gracias a avances tecnológicos significativos en el campo del desarrollo de software, en gran parte debido a la influencia del lenguaje de programación C++.

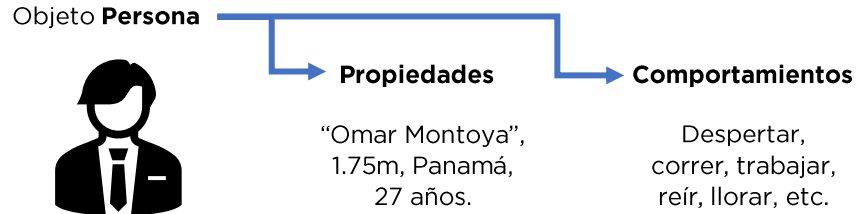
En un principio C++ era solo una extensión del lenguaje de programación C. C solo tenía soporte para programación estructurada por lo que se creó la extensión C++ para proporcionar un lenguaje de programación que permitiera a los desarrolladores utilizar programación orientada a objetos.

Debido al surgimiento de C++, otros lenguajes de programación fueron agregando características de programación orientada a objetos. Sin embargo, la adición de estas características a lenguajes que no fueron diseñados inicialmente para ellas condujo a problemas de compatibilidad y la capacidad de mantenimiento del código.

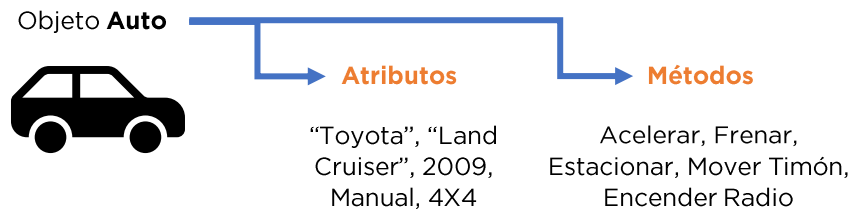
Para resolver este problema se crearon lenguajes de programación robustos basados en la orientación a objetos. La competencia fue ganada por Java que salió al mercado en 1995 por Sun Microsystems (ahora le pertenece a Oracle), primordialmente gracias a su máquina virtual Java.

3. ¿Qué es un Objeto?




Los objetos son todas las cosas que tienen propiedades y comportamientos. Las propiedades son las características de nuestros objetos. Los comportamientos serán todas las operaciones que nuestros objetos pueden realizar.


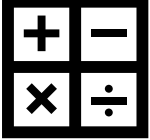



En la POO las propiedades son llamadas atributos; mientras que los comportamientos son llamados métodos.



Los objetos pueden ser tanto físicos (cosas que podemos tocar) o conceptuales (cosas que no son reales en el mundo físico, pero pueden ser conceptos o ideas).

Tipo de Objeto	Objeto	Atributos	Métodos
Físico	Televisor 	"Sony", 50", OLED, 2024.	Encender, apagar, cambiar Canal, bajar Volumen
Físico	Libro 	"Java como programar", 10ma edición, "Deitel y Deitel", "Pearson"	Abrir Libro, Pasar Página, Imprimir, cerrarLibro
Físico	Consola 	"Nintendo", "Switch", 2022, "Amarillo", 10GB.	Encender, Apagar, Iniciar Juego, Descargar Juego, Bloquear

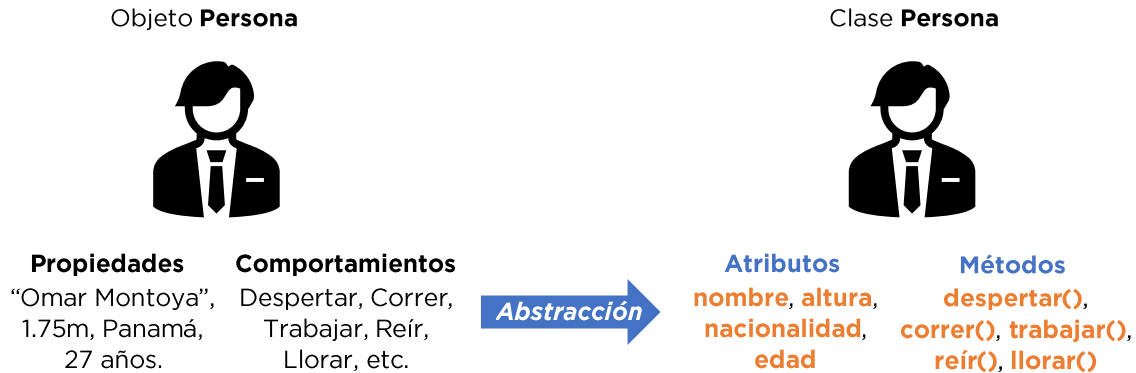
Conceptual	Sesión de Usuario 	"codewithomarm", "123456", "admin", 30 min, 03/05/2024	Iniciar Sesión, Cerrar Sesión, Crear Publicación, Eliminar Sesión.
Conceptual	Suma 	10, 20, 30	Agregar primer numero, agregar segundo numero, calcular resultado, mostrar resultado
Conceptual	Reporte 	"Omar Montoya", "Workforce", "Marzo 2024", "Reporte de Asistencia"	Crear Reporte, Actualizar Reporte, Cerrar Reporte, Enviar Reporte

Los objetos que utilizemos en nuestros programas java serán tan sencillos o complejos como nuestro programa lo necesite. En la programación orientada a objetos, los objetos funcionan como engranajes que harán posible que nuestro software funcione y ejecute sus funciones.

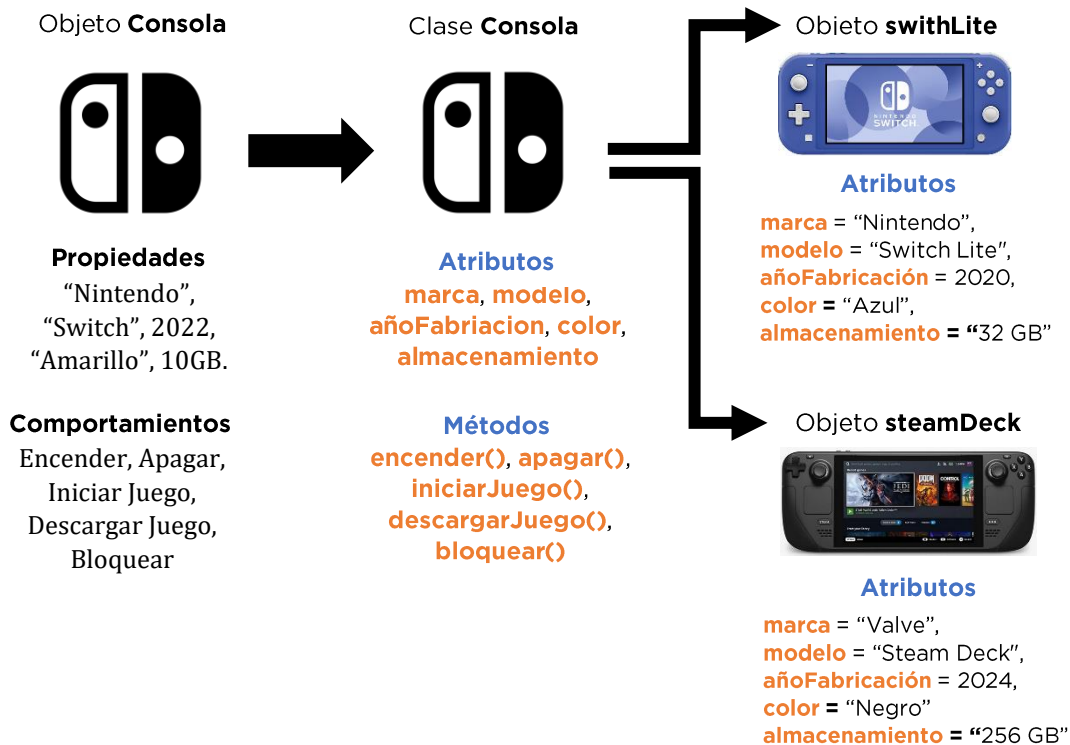
4. ¿Qué es una Clase?

Las Clases son los modelos o plantillas sobre las cuales construimos nuestros objetos, es decir, las clases son los “moldes” que nos permiten generar objetos.

Para la creación de clases se utiliza la abstracción. La Abstracción se trata de analizar objetos de forma independiente, sus propiedades, características y comportamientos, para abstraer su composición y generar un modelo, lo que traducimos a código como clases. Los atributos serán nombrados utilizando nombres de sustantivos y los métodos utilizando nombres de verbos o sustantivos + verbos.



Una vez creada la Clase, tenemos el molde para crear la cantidad de objetos que nuestro programa necesite:



5. Principios de la Programación Orientada a Objetos

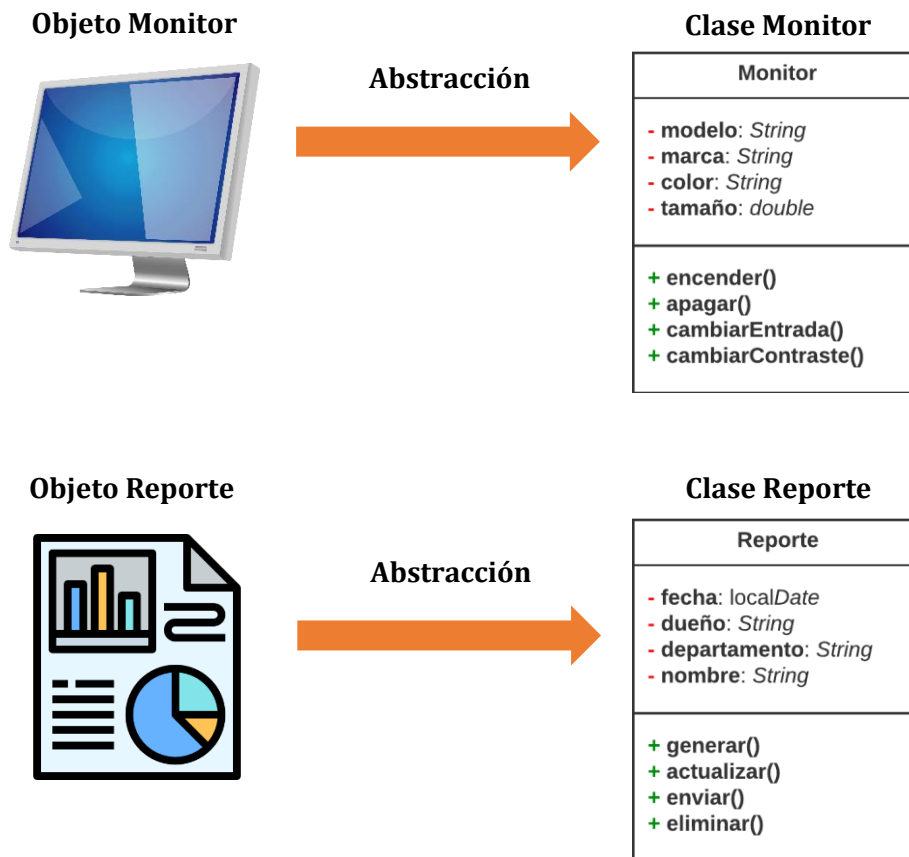
Abstracción

Permite representar objetos del mundo real en un programa de manera simplificada, capturando solo los detalles relevantes para la aplicación.

Define las características esenciales de un objeto y sus comportamientos. Cada objeto en el sistema sirve como modelo que puede realizar trabajos, cambiar sus características y comunicarse con otros objetos.

Puede abstraer objetos del mundo real ya sean físicos (persona, auto, monitor) o también pueden ser conceptuales (sesión de usuario, reporte, suma).

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

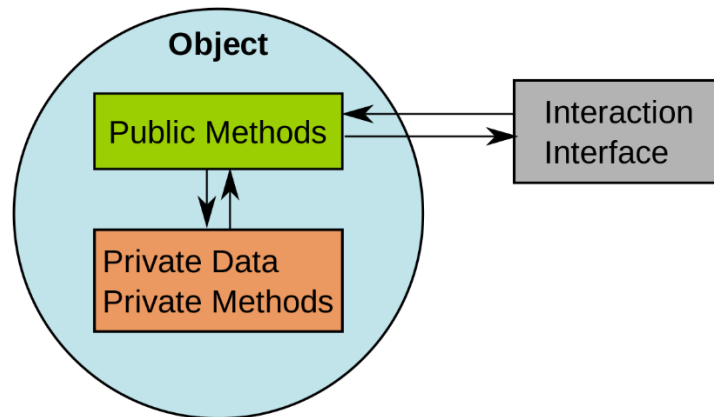
La encapsulación es un mecanismo para reunir datos y métodos dentro de una estructura ocultando la implementación del objeto, es decir, impidiendo el acceso a los datos por cualquier medio que no sean los servicios propuestos. La encapsulación permite, por tanto, garantizar la integridad de los datos contenidos en el objeto. Por lo tanto, si queremos

proteger la información contra modificaciones inesperadas, debemos recurrir al principio de encapsulación.

En Java, como en muchos lenguajes de programación orientados a objetos, las clases, los atributos y los métodos tienen niveles de accesibilidad, que indican en qué circunstancias se puede acceder a estos elementos.

El usuario de una clase no tiene por qué saber cómo se estructuran los datos del objeto, es decir, su implementación. Así, prohibiendo al usuario modificar directamente los atributos, y obligándole a utilizar las funciones definidas para modificarlos (denominadas interfaces), conseguimos garantizar la integridad de los datos. Por ejemplo, se puede asegurar que el tipo de datos proporcionados se ajusta a las expectativas, o que los datos están en el rango esperado.

La encapsulación permite definir niveles de visibilidad para los elementos de la clase. Estos niveles de visibilidad definen los derechos de acceso a los datos en función de si se accede a ellos mediante un método de la propia clase, de una clase heredada o de cualquier otra clase.

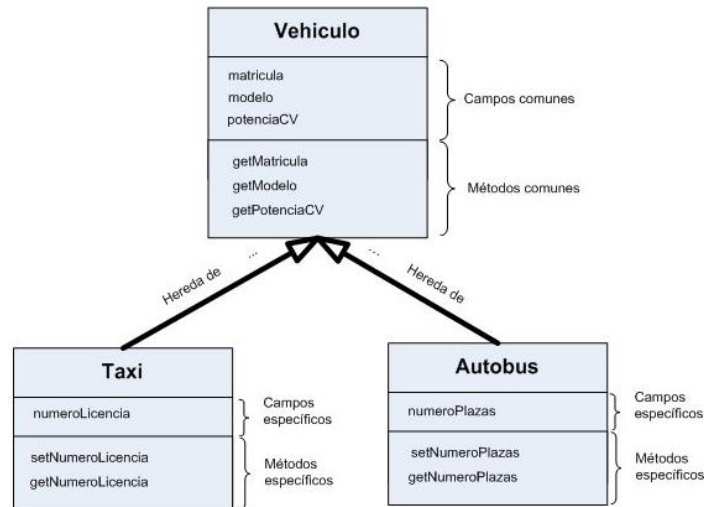


Los niveles de accesibilidad los veremos a detalles en la Clase 5.

Herencia

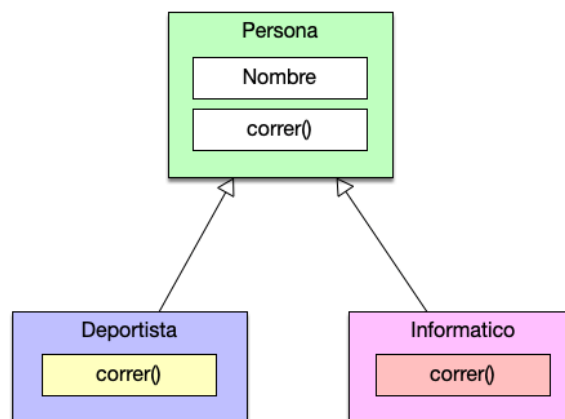
Permite crear nuevas clases (objetos) basadas en clases existentes, heredando sus características y comportamientos. Esto fomenta la reutilización del código y la creación de jerarquías de clases.

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. 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; siendo de alta complejidad técnica por lo cual suele recurrirse a la herencia virtual para evitar la duplicación de datos.



Polimorfismo

Permite que un objeto pueda presentar diferentes formas, es decir, que pueda comportarse de distintas maneras dependiendo del contexto. Esto se logra mediante el uso de métodos con el mismo nombre, pero con implementaciones diferentes en las clases derivadas.



En este ejemplo tenemos tres clases distintas en las que tenemos el método *correr()*. A pesar de que el método se llama igual en las tres clases, el desarrollador tiene la posibilidad de definir comportamientos distintos para cada una de las clases.