Programación Orientada a Objetos

Joan Belkham Chindoy Hernández

ID: 971589

Instructor: William Matallana

Corporación Universitaria Minuto de Dios (UNIMINUTO)

Sede: Zipaquirá.

Estructura de Datos.

Programa: Ingeniería de Sistemas.

Tabla de Contenidos

Tabla de Contenidos **1**
Introducción **2**
Objetivos **3**
Objetivo General 3
Objetivos Específicos 3
Programación Orientada a Objetos **4**
¿Qué es? 4
¿Qué son las clases en Java? 4
Tipos de métodos en Java 6
Principios de la Programación Orientada a Objetos 7
¿Qué son los diagramas de clases? 8
Interacciones 9
¿Qué herramientas se pueden utilizar para realizar diagramas de clases? 10
¿Qué son las clases abstractas en Java? 11
¿Qué son las interfaces en Java? 11
Conclusiones **12**
Referencias Bibliográficas **13**

Introducción

La Programación Orientada a Objetos (POO) es un enfoque utilizado en el desarrollo de software que permite organizar el código de manera más organizada y reutilizable, mediante la creación de clases y objetos, los programadores pueden representar elementos del mundo real dentro de un sistema informático. Java es uno de los lenguajes de programación que utiliza este enfoque, lo que permite desarrollar aplicaciones de manera eficiente y ordenada. Además, los diagramas de clases ayudan a visualizar la estructura del código y la relación entre los componentes.

El objetivo de esta investigación es indagar acerca de los conceptos fundamentales de la POO en Java, como las clases, los diagramas de clases, los diferentes tipos de métodos y los principios básicos de esta. Además, se analizará el papel de las clases abstractas y las interfaces, explicando su importancia en el diseño de programas.

Objetivos

Objetivo General

Comprender los conceptos esenciales de la POO en Java y su importancia en la organización y desarrollo de aplicaciones.

Objetivos Específicos

1. Explicar de manera sencilla cómo funcionan las clases y objetos en Java y su papel en la construcción de programas.
2. Describir qué son los diagramas de clases, cómo ayudan a planificar un sistema y qué herramientas se pueden utilizar para crearlos.
3. Analizar la utilidad de las clases abstractas y las interfaces, y cómo contribuyen a mejorar la organización del código.

Programación Orientada a Objetos

¿Qué es?

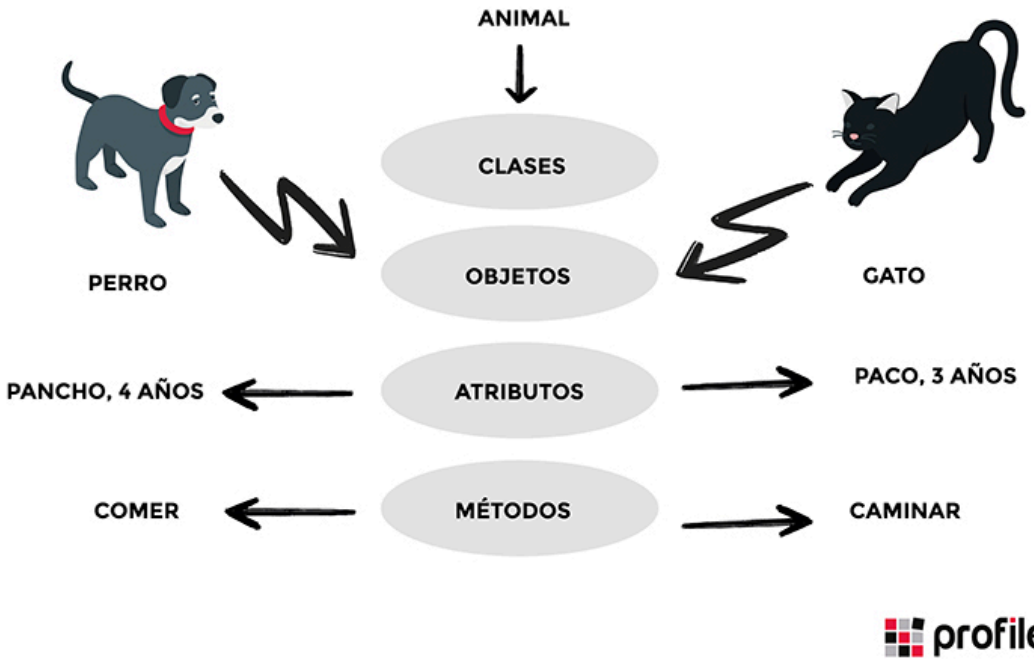
La POO es una forma de programar basada en clases y objetos, en lugar de centrarse solo en la lógica del código, organiza la información y las acciones en estructuras reutilizables llamadas clases, que sirven como moldes para crear objetos.

Antes, los lenguajes de programación se enfocaban más en la secuencia de instrucciones, pero con el tiempo surgieron nuevos enfoques, y la POO se volvió muy utilizada en lenguajes como Java, C# y Python. Es útil para desarrollar sistemas grandes porque permite pensar en términos de objetos y sus interacciones, en lugar de sólo funciones. Así , un programa se construye haciendo que varios objetos trabajen juntos, facilitando mucho la organización y reutilización.

¿Qué son las clases en Java?

En Java, las clases son muy importantes en la POO, ya que sirven como planos para crear objetos; Un objeto es una instancia de una clase, con características (atributos) y comportamientos definidos (métodos). Por ejemplo, en un juego de carreras, se podría tener una clase “Coche” que incluya atributos como modelo, velocidad máxima y potencia, además de métodos como acelerar, frenar y girar.

Imagen 1



Tomado de: https://profile.es/blog/que-es-la-programacion-orientada-a-objetos/#Clases_Objetos_Instancias

Conceptos Clave:

- Atributos: Representan las características del objeto.
- Métodos: Definen las acciones o comportamientos que pueden realizar los objetos.
- Constructores: Se usan para crear objetos y asignarles valores iniciales.
- Encapsulamiento: Controla el acceso a los atributos y métodos de una clase, usando modificadores como private, protected y public.
- Herencia: Permite que una clase herede atributos y métodos de otra.

- Instanciación: Es el proceso de crear un objeto a partir de una clase con el operador “new”.

Tipos de métodos en Java

En Java, los métodos son bloques de código que permiten ejecutar acciones dentro de una clase. Dependiendo de su propósito y funcionamiento, existen varios tipos de métodos:

1. Métodos Estáticos (Static Methods): Pertenecen a la clase en sí y no a sus instancias, se pueden llamar directamente desde la clase sin necesidad de crear un objeto. Son ideales para realizar tareas generales, como cálculos matemáticos o métodos de utilidad, ya que no dependen del estado de un objeto.

2. Métodos de Instancia (Instance Methods): Operan sobre instancias de una clase y pueden acceder a los atributos y otros métodos de la misma, se llaman a través de un objeto creado a partir de la clase. Son los más utilizados en POO, ya que permiten manipular los datos de un objeto de manera específica.

3. Métodos con Parámetros (Methods with Parameters): Son métodos que reciben valores al ser llamados. Estos parámetros permiten modificar el comportamiento del método según la información que se le pase, haciéndolo más flexible y reutilizable en diferentes situaciones.

4. Métodos con Valor de Retorno (Methods with Return Value): Después de ejecutarse, estos métodos devuelven un resultado, que puede ser de cualquier tipo de dato, como números, cadenas de texto u objetos. Son útiles cuando se necesita procesar información y devolver un resultado para su uso posterior.

5. Constructores (Constructors): Son métodos especiales utilizados para inicializar objetos al momento de su creación. Tienen el mismo nombre que la clase y pueden recibir

parámetros para establecer valores iniciales. Aunque no son métodos en el sentido tradicional, cumplen una función clave en la creación de instancias.

6. Métodos Getter y Setter (Getter and Setter Methods): Estos métodos permiten acceder y modificar los atributos privados de una clase de manera controlada. Los getters se utilizan para obtener el valor de un atributo, mientras que los setters permiten modificarlo. Son fundamentales para aplicar el principio de encapsulación y evitar el acceso directo a los atributos de un objeto.

Principios de la Programación Orientada a Objetos

La POO se basa en cuatro principios esenciales que permiten diseñar software estructurado, flexible y fácil de mantener, estos principios ayudan a organizar el código y gestionar la complejidad a medida que los sistemas crecen.

1. Abstracción: Permite modelar objetos del mundo real enfocándose solo en los aspectos más relevantes y ocultando detalles innecesarios. Esto facilita el desarrollo al simplificar la complejidad del sistema y mejorar la claridad del código.

2. Encapsulamiento: Protege los datos de un objeto evitando modificaciones directas desde fuera de la clase. Se logra a través de métodos específicos que controlan el acceso a los atributos, garantizando seguridad y facilidad de mantenimiento en el código.

3. Herencia: Este principio permite que una clase adquiera características y comportamientos de otra, facilitando la reutilización del código. La herencia permite estructurar las clases en jerarquías que reflejan relaciones naturales entre objetos.

4. Polimorfismo: El polimorfismo permite que un mismo método pueda tener diferentes comportamientos según el contexto en el que se use. Esto hace que el código sea más

flexible y escalable, permitiendo tratar diferentes objetos de manera uniforme sin necesidad de conocer su tipo exacto.

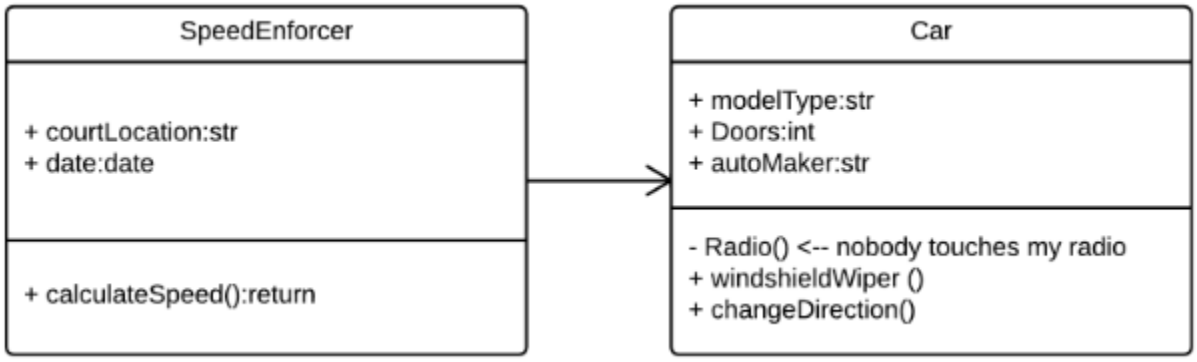
¿Qué son los diagramas de clases?

Son una de las herramientas más utilizadas del Lenguaje Unificado de Modelado (UML) y se utilizan para representar la estructura de un sistema de software. Son esenciales para modelar clases, sus atributos, métodos y las relaciones entre ellas, lo que facilita la organización y planificación del sistema.

El UML se ha convertido en un estándar en la POO, ya que permite visualizar la interacción entre clases y objetos dentro de un sistema. Cada clase en un diagrama de clases se representa mediante un rectángulo dividido en tres secciones:

- Sección superior: Contiene el nombre de la clase. Esta sección siempre es necesaria, ya sea que se esté hablando del clasificador o de un objeto.
- Sección central: Contiene los atributos de la clase. Esta sección se usa para describir cualidades de la clase. Esto solo es necesario al describir una instancia específica de una clase.
- Sección inferior: Incluye operaciones de clases (métodos). Esto está organizado en un formato de lista. Cada operación requiere su propia línea. Las operaciones describen cómo una clase puede interactuar con los datos.

Imagen 2



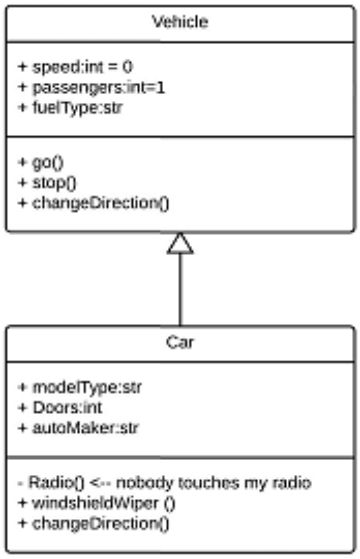
Tomado de: https://www.lucidchart.com/pages/es/tutorial-de-diagrama-de-clases-uml

Interacciones

Las interacciones en UML permiten modelar relaciones estructurales dentro del software, facilitando su desarrollo y mantenimiento al representar cómo se comunican y se relacionan las clases entre sí.

- Herencia: En un diagrama de clases, la herencia se representa con una línea continua que conecta la subclase con la superclase, terminando en una flecha cerrada y vacía apuntando hacia la clase principal.

Imagen 3



Tomado de: https://www.lucidchart.com/pages/es/tutorial-de-diagrama-de-clases-uml

Por ejemplo, si una clase "Vehículo" tiene atributos como velocidad, número de pasajeros y tipo de combustible, una subclase "Auto" heredará estos atributos y métodos como arrancar(), frenar() y cambiarDirección(), además de agregar sus propios atributos específicos como modelo, número de puertas y fabricante, junto con métodos adicionales como activarRadio(), usarLimpiaparabrisas() y regularTemperatura().

¿Qué herramientas se pueden utilizar para realizar diagramas de clases?

Las mejores herramientas encontradas fueron:

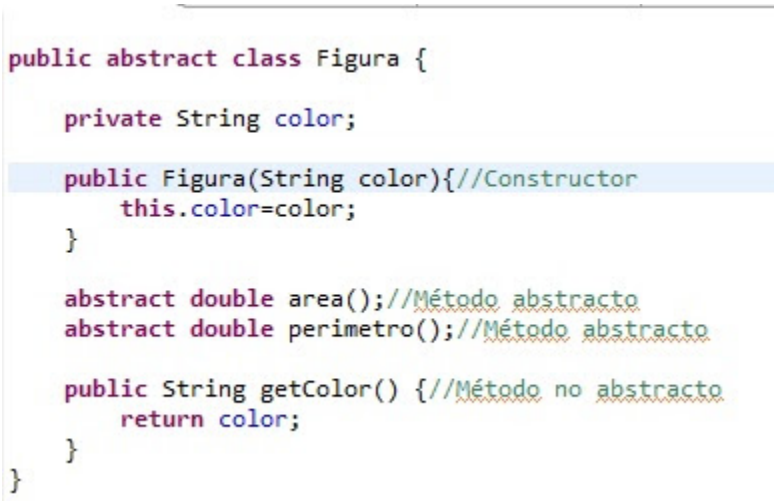
1. ClickUp: Ofrece una amplia gama de funcionalidades, incluyendo mapas mentales y pizarras para la creación de diagramas UML. Su interfaz intuitiva permite la colaboración en tiempo real y la integración con más de 1.000 aplicaciones.
2. Astah UML: Facilita la creación de diagramas UML profesionales y cuenta con una extensa biblioteca de plugins, también permite la generación de código, ingeniería inversa y se integra con plataformas como Confluence y yUML.
3. Draw.io: Herramienta gratuita basada en la web que permite crear diagramas UML de manera sencilla. No requiere instalación, admite trabajo offline y se integra con servicios reconocidos como Google Drive, OneDrive y GitHub.
4. Lucidchart: Plataforma en línea para diagramas UML colaborativos. Ofrece una amplia variedad de plantillas y formas, y se integra con herramientas como Google Drive, Slack y Confluence.
5. Microsoft Visio: Parte de la suite de Microsoft, ideal para crear diagramas profesionales. Ofrece muchas plantillas y formas, y se integra perfectamente con otras aplicaciones de Microsoft.

¿Qué son las clases abstractas en Java?

Son clases que no pueden instanciarse directamente y sirven como plantillas para otras clases. Se utilizan cuando se necesita definir una estructura común para un grupo de clases relacionadas.

Una clase abstracta puede contener atributos, métodos con implementación y abstractos (sin cuerpo); Los métodos abstractos establecen qué deben hacer las clases hijas, pero dejan que cada una defina cómo lo hará, se declaran con la palabra clave “abstract”.

Imagen 4



```
public abstract class Figura {

    private String color;

    public Figura(String color){//Constructor
        this.color=color;
    }

    abstract double area();//Método abstracto
    abstract double perimetro();//Método abstracto

    public String getColor() {//Método no abstracto
        return color;
    }
}
```

Tomado de: https://picarcodigo.blogspot.com/2012/10/clases-abstractas.html

¿Qué son las interfaces en Java?

Son estructuras que establecen un contrato que las clases deben cumplir. Solo pueden contener métodos abstractos (sin implementación) y constantes. A diferencia de las clases abstractas, las interfaces permiten que una clase implemente múltiples comportamientos, ya que Java no admite herencia múltiple con clases, pero sí permite múltiples interfaces, se declaran con la palabra clave “interface”.

Conclusiones

1. La Programación Orientada a Objetos facilita el desarrollo de software estructurado y reutilizable. Gracias a sus principios fundamentales, la POO permite organizar mejor el código y gestionar la complejidad de los sistemas a medida que evolucionan.
2. Los diagramas de clases son esenciales para modelar sistemas antes de su implementación, lo que permite visualizar la estructura y relaciones entre objetos, facilitando la comunicación entre desarrolladores y la planificación del software.
3. Las interacciones en UML, como la herencia, permiten modelar relaciones entre clases.
4. El uso de estos conceptos mejoran la escalabilidad y mantenibilidad del software, tanto la correcta aplicación de la POO, junto al modelado UML.

Referencias Bibliográficas

García, E. (2020, 1 27). *Diferencia entre clases abstractas e interfaces en Java*. códigofacilito. Retrieved 3 18, 2025, from https://codigofacilito.com/articulos/clases-abstractas-interfaces-java

INTRODUCCIÓN A LOS MÉTODOS EN JAVA: TIPOS Y FUNCIONAMIENTO. (2023, 8 30). Nascor formación. Retrieved 3 18, 2025, from https://cursosnascor.com/blog-detalle/introduccion-los-metodos-en-java-tipos-y-funcionamiento

Introducción a POO en Java: Objetos y clases. (2023, 10 27). OpenWebinars. Retrieved 3 18, 2025, from https://openwebinars.net/blog/introduccion-a-poo-en-java-objetos-y-clases/#qu%C3%A9-son-las-clases-en-java

¿Qué es la Programación Orientada a Objetos? (n.d.). Profile. Retrieved 3 18, 2025, from https://profile.es/blog/que-es-la-programacion-orientada-a-objetos/#Clases_objetos_e_instancias

10 Mejores Herramientas de Software para Diagramas UML en 2025 (Gratis/a y de Pago). (2024, 10 19). ClickUp. Retrieved 3 18, 2025, from https://clickup.com/es-ES/blog/69659/software-de-diagramas-uml

Tutorial de diagrama de clases UML. (n.d.). Lucidchart. Retrieved 3 18, 2025, from https://www.lucidchart.com/pages/es/tutorial-de-diagrama-de-clases-uml

Valencia, A. (2024, 5 9). *¿Qué es la Programación Orientada a Objetos (POO) y cuáles son sus principios fundamentales?* CodersLink. Retrieved 3 18, 2025, from https://coderslink.com/talento/blog/que-es-la-programacion-orientada-a-objetos-poo-y-cuales-son-sus-principios-fundamentales/#:~:text=A%20trav%C3%A9s%20de%20la%20encapsulaci%C3%B3n,a%20diferentes%20necesidades%20y%20contextos