Conceptos básicos programación orientada a objetos

Breve descripción:

En este componente formativo se abordan los conceptos clave del paradigma orientado a objetos, entre los que se encuentran la abstracción, el encapsulamiento, la modularidad, la jerarquía y el polimorfismo. Adicionalmente, se introducen los conceptos de clases y objetos desde la perspectiva de este paradigma.

Abril 2024

Grupo de Ejecución de la Formación Virtual

Tabla de contenido

Introducción

¿Sabías que algunos de los elementos fundamentales que se usan para la descripción de un algoritmo en un lenguaje de programación, son los identificadores? Estos pueden ser variables o constantes, dependiendo del comportamiento de los valores que registramos en ellos.

Estos identificadores se componen principalmente de tres elementos: nombre del identificador, valor almacenado y posición física de la memoria donde se encuentra. Para poder utilizar identificadores, se debe realizar el proceso de declaración del identificador, donde se define el tipo de dato asociado, el cual indica el tamaño a reservar en la memoria y el nombre de dicho identificador.

Cuando el problema a desarrollar requiere gran cantidad de información y adicionalmente, la información se compone de diversos tipos de datos, el mecanismo de identificadores se vuelve obsoleto, no solo por la gran cantidad de estructuras de memoria que se deberán construir, sino porque su gestión también será más compleja.

Al tener problemas cada vez más complejos y grandes cantidades de datos, se requieren mejores mecanismos para gestionar los datos y es a partir de este escenario, que surge la programación orientada a objetos, como un paradigma que busca aproximar el manejo de la información, desde la forma como es procesada internamente en los programas a como realmente se representa en la vida cotidiana.

A lo largo del componente formativo, se profundizará sobre los conceptos en los que se fundamenta el paradigma y luego se examinarán en detalle algunos de los elementos centrales que se utilizan para implementar el paradigma. Bienvenido.

1. Paradigma orientado a objetos

El paradigma orientado a objetos es un concepto que inició en los años 70, pero que se logró estandarizar alrededor de 1997 con la aparición del Lenguaje Unificado de Modelado UML, teniendo presente que la palabra objetos no hace referencia necesariamente a componentes gráficos de un sistema de información (Rumbaugh, et al., 2004).

Según Booch et al. (2007), para que un sistema o modelo sea orientado a objetos debe tener los siguientes elementos:

- Abstracción
- Encapsulamiento
- Modularidad
- Jerarquía
- Polimorfismo

Abstracción

La abstracción es un proceso natural del ser humano y consiste esencialmente en la identificación de características claves según el contexto y conocimiento de las personas. De tal forma, se constituye en el medio mediante el cual se enfrentan los problemas complejos para extraer las características esenciales y posteriormente, representar dichas características en objetos.

La abstracción se enfoca en la vista externa de los elementos, lo cual permite separar su comportamiento esencial de su implementación e ignorar detalles internos. Mediante la abstracción hay un énfasis en el qué es y qué hace y no en el cómo debe implementarse o cómo funciona algo. Aguilar, 2021

Analicemos el siguiente ejemplo, sobre el proceso de abstracción.

En diferentes escenarios se aplican ciertos niveles de abstracción, por ejemplo, a la hora de decidir qué teléfono celular se desea comprar.

- **Edad adulta**

  Una persona de negocios de edad adulta, puede interesarle la disponibilidad de comunicación por medio de telefonía móvil, así como la conectividad a las redes de datos y la facilidad con la que pueda conectarse a cualquiera de esas redes.

- **Adolescente**

  Probablemente una persona más joven, por ejemplo, un adolescente, podría interesarle más las capacidades para soportar diferentes tipos de archivos multimedia y la velocidad con la que se puedan ejecutar juegos.

- **Desarrollador**

  Una persona que desarrolla "software" para móviles, podría interesarle más otras características como el tipo de sistema operativo, la operatividad con los sensores disponibles, y cosas de ese estilo.

Podemos concluir que lo que se busca ahora, es llevar ese mismo proceso para entender la esencia de los problemas complejos a desarrollar, y construir, a partir de esas abstracciones, que representan todos los elementos involucrados.

Encapsulamiento

El encapsulamiento es el proceso que permite agrupar datos y también operaciones que relacionan estos datos bajo una misma unidad lógica.

La encapsulación además, trae consigo algo llamado ocultamiento de datos, que consiste en la posibilidad de separar los elementos esenciales con los que se opera algo (interfaz – externa y pública) y de los detalles de su implementación (interna y privada). El encapsulamiento es un proceso que se debe realizar luego del proceso de abstracción. Aguilar, 2021

A continuación, presentamos un ejemplo de encapsulamiento.

Un control remoto, agrupa varios elementos que tienen funcionamientos particulares pero que en conjunto permiten la manipulación a distancia de un televisor.

Adicionalmente, como usuarios de un control remoto interactuamos con él mediante su interfaz externa y pública, es decir, sus botones de colores, cada botón tiene un identificador que es conocido para nosotros y mediante el cual manipulamos el televisor.

Como usuarios de un control remoto no es esencial su implementación (interna y privada), es decir, cómo interactúan las ondas de luz de baja frecuencia o las señales que son decodificadas por los aparatos electrónicos.

Modularidad

La modularidad es la propiedad que permite dividir una aplicación o sistema en partes más pequeñas, idealmente deben ser muy independientes (bajo acoplamiento) y altamente funcionales (alta cohesión).

Dividir un sistema en partes más pequeñas, tiene varios beneficios, entre los cuales se pueden listar:

- Se produce la complejidad. Los problemas más pequeños son más fáciles de resolver que los problemas grandes.
- Cada módulo puede ser ejecutado y probado de forma independiente, lo cual facilita la corrección de errores y permite que un problema pueda ser desarrollado por varias personas en forma paralela.

Como ejemplo de modularidad, tenemos:

Un vehículo se puede construir a partir del desarrollo de cada una de sus partes de forma independiente, e incluso, por fabricantes diferentes: puertas, ventanas, motor, chasis, asientos, llantas, etc.

Luego, en un proceso que la mayoría de las veces es automatizado, se unen estas partes para producir el vehículo.

Cuando el vehículo es reparado u optimizado, se puede cambiar cada una de sus partes, de forma individual.

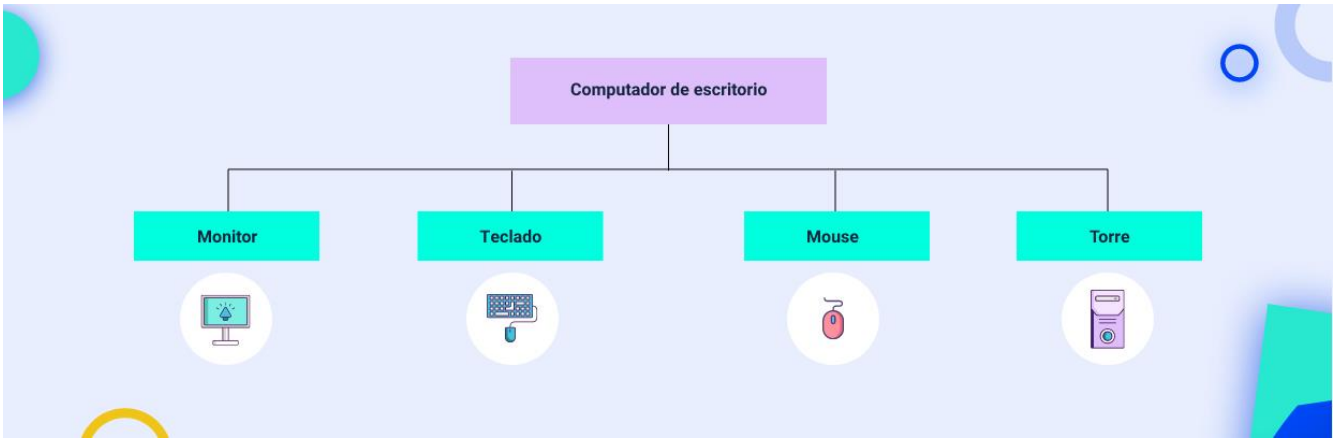
Jerarquía

Las jerarquías son clasificaciones y ordenaciones de las abstracciones de un problema (Aguilar, 1998), permiten comprender de forma general la estructura de un sistema y favorecen, en algunos casos, la reutilización de características y comportamientos de las abstracciones definidas.

Las jerarquías que se producen desde el punto de vista de la estructura de objetos se les conoce como agregación y composición, y las jerarquías que se producen desde las estructuras de clases se les conoce como generalización y especialización.

Conozcamos la jerarquía de un computador de escritorio:

Figura 1. Jerarquía de un computador de escritorio



Dependiendo del nivel de dependencia que existe entre la abstracción que agrupa y las abstracciones que son partes, se definen relaciones de composición o agregación.

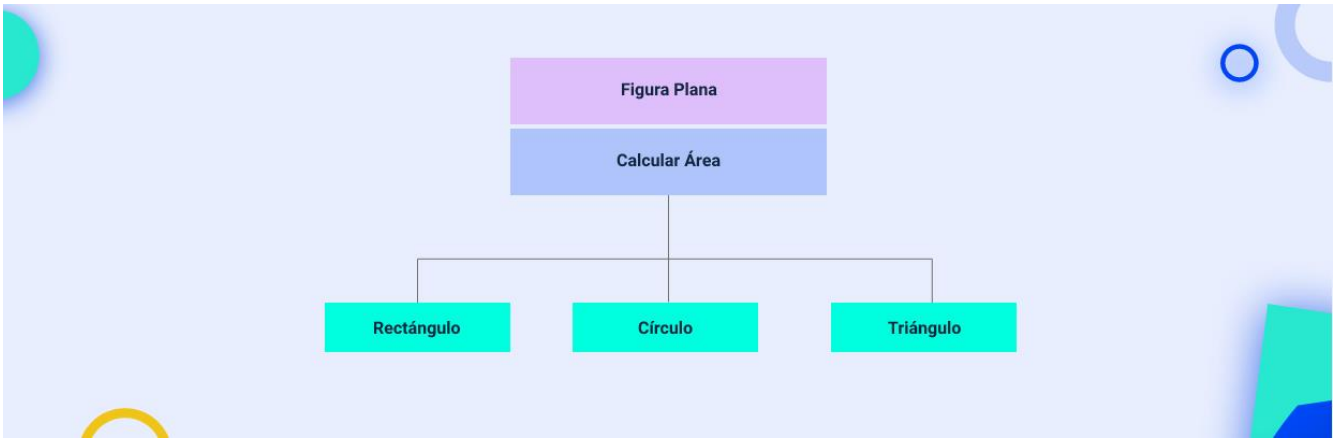
La representación de las uniones entre las abstracciones de una jerarquía usa símbolos diferentes para diferenciar lo que son jerarquías de clases y jerarquías de objetos. Esto se detalla en el tema de diagrama de clases.

Polimorfismo

Es la característica que permite que una abstracción tome diferentes formas o diferentes comportamientos según el contexto. Esta característica se potencia en compañía de las jerarquías vistas anteriormente.

En la siguiente figura, se muestra un ejemplo de una jerarquía sobre figuras planas y adicionalmente, se indica que hay un comportamiento llamado Calcular Área, es decir, todas las figuras planas pueden calcular su área.

Figura 2. Ejemplo de una jerarquía sobre figuras planas



En la jerarquía de clases del ejemplo anterior, las abstracciones rectángulo, círculo y triángulo heredan las características de la abstracción figura plana incluyendo el comportamiento de Calcular Área. Sin embargo, este comportamiento tiene diferentes formas de calcularse dependiendo qué abstracción de la jerarquía es la que la utilice:

- Para el caso de la abstracción rectángulo, el cálculo de área se obtiene luego de multiplicar el valor de la base y la altura de la figura.
- Para la abstracción círculo, se debe multiplicar la constante π (Pi) por el valor del radio al cuadrado.
- Para la abstracción triángulo se calcula diferente, en este caso, el cálculo del área se obtiene luego de multiplicar el valor de la base por la altura de la figura y dividir el valor entre dos.

2. Clases y objetos

Las clases y objetos son el corazón de la programación orientada a objetos y en la construcción de estos elementos, se refleja cada una de las características del paradigma orientado a objetos revisado en el apartado anterior. Algunas veces parece que ambos elementos se traslapan y algunas otras veces se suele hablar de forma indiscriminada de estos dos elementos, por lo cual, a continuación, se ampliará la revisión para comprender sus características y diferencias.

Un objeto en el paradigma orientado a objetos representa un objeto del mundo real, el objeto debe ser especializado y solo se encarga de una tarea.

Un objeto puede ser:

- Cosas tangibles como un avión, un auto o un televisor.
- Roles o papeles como un gerente, un cliente o un vendedor.
- Organizaciones como una empresa o un departamento.
- Interacciones como transacciones o contratos.
- Incidentes como vuelos, sucesos o accidentes.
- Lugares como muelles, carreteras, etc.

Los objetos se comunican con otros objetos por medio de mensajes y su estructura se describe en la siguiente tabla.

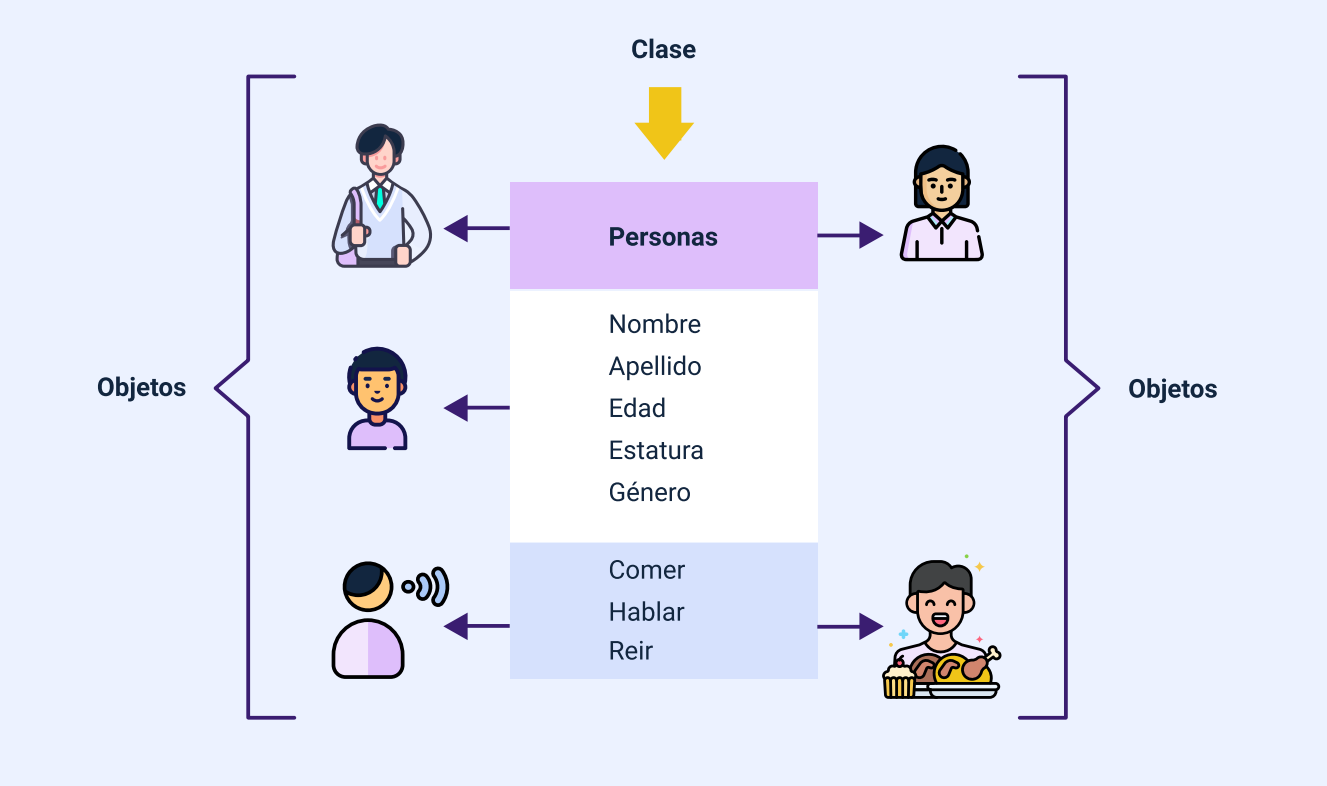
Tabla 1. Estructura de un objeto

Elemento	Descripción
Identidad (nombre).	Representa las características del objeto. Son los parámetros que lo definen y lo diferencian de objetos del mismo tipo.

Elemento	Descripción
Atributos y propiedades (estado).	Representa las características del objeto. Son los parámetros que lo definen y lo diferencian de objetos del mismo tipo.
Métodos.	Representan el comportamiento del objeto. Acciones que realizan o manejan los datos.

Ahora, una clase es una abstracción o visión generalizada de un conjunto de objetos que tienen características (atributos y propiedades) y métodos iguales o similares. Es decir, una clase representa las características comunes de un conjunto de objetos, como se muestra en la figura que se encuentra a continuación.

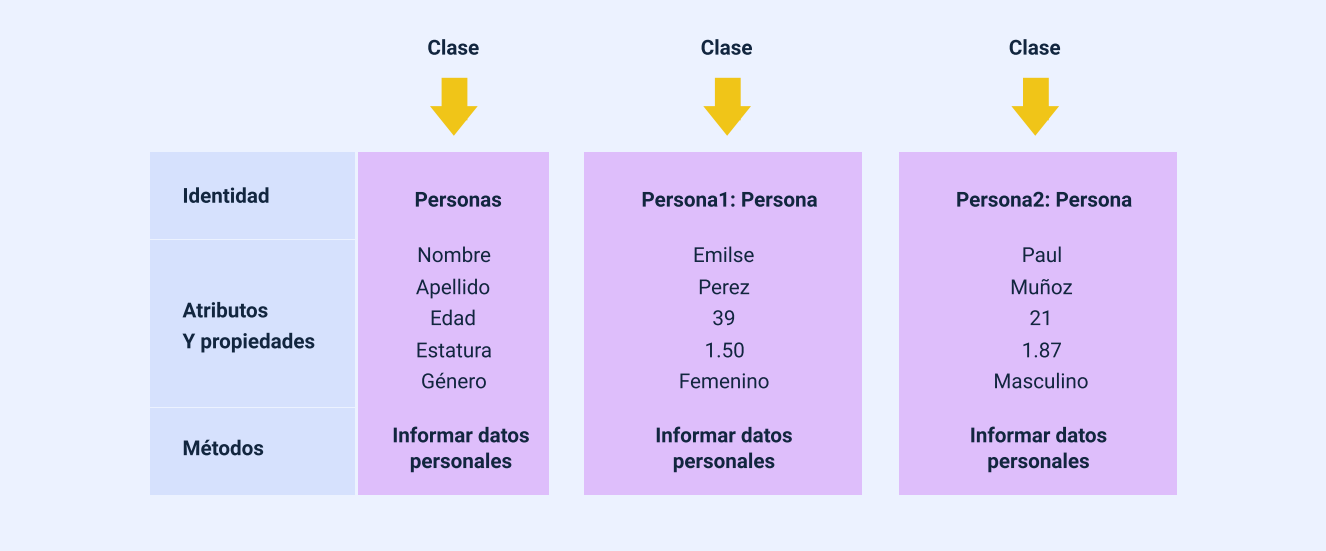
Figura 3. Ejemplo clases y objetos



Una clase es un tipo de dato definido por el usuario, no existe en forma concreta, debido a que es solo una abstracción. A partir de la definición de una clase, se pueden construir objetos. Las clases son equivalentes a los tipos de datos y los objetos son los identificadores que se crean a partir de este tipo particular de dato.

Así, una clase puede servir para definir (instanciar) objetos. Un objeto es una instancia de una clase totalmente independiente de otros objetos de la misma clase. La estructura de los objetos corresponderá a la estructura definida por la clase desde la cual se crean, así como se observa en la siguiente figura.

Figura 4. Estructura de una clase y objetos



Atributos

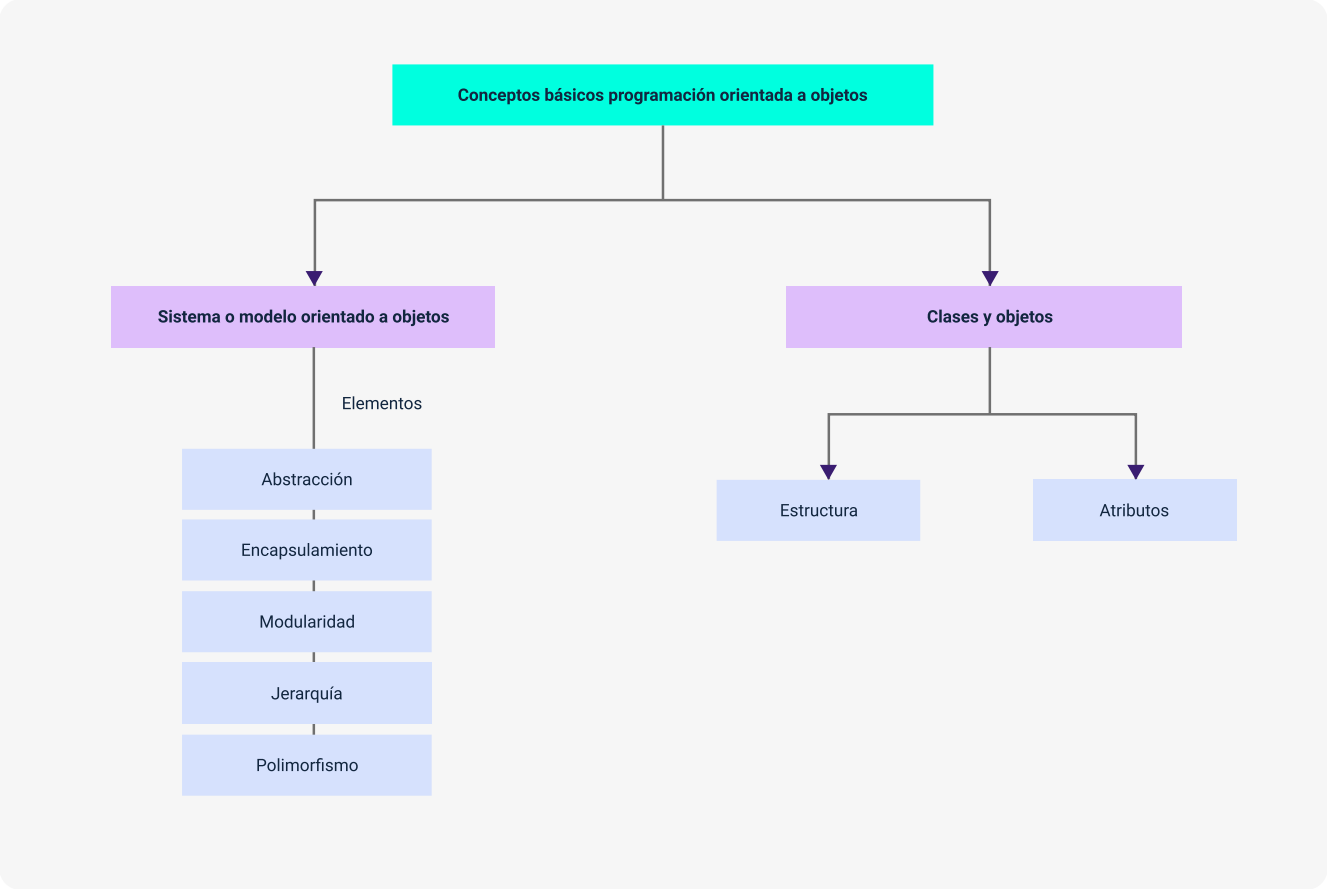
Los atributos de los objetos que se definen en las clases, cuando se describen en lenguaje de modelado formal o en un lenguaje de programación, se estructuran de forma similar a como se definen los identificadores, es decir, cada atributo se expresa con un tipo de dato y nombre del identificador.

Adicionalmente, tanto los atributos como los métodos tendrán una especificación, en la que se indica si corresponden a elementos públicos (interfaces visibles) o elementos privados (implementación interna), lo que representa una aplicación de la propiedad del encapsulamiento.

a) Lo primero por hacer, es por medio de la abstracción, identificar de la información existente, cuáles son los posibles objetos y las posibles características claves para tener en cuenta al realizar un registro de matrícula y la información esencial relacionada con los estudiantes.
b) Puede ser útil también, verificar de estudiantes reales el tipo de información que requiere la institución.
c) Luego de esto, se empieza a construir las plantillas (clases) con la estructura de los elementos identificados. Posteriormente, estas clases definidas, permitirán instanciar objetos que corresponden a la información concreta con la que se manipulará el sistema.

Síntesis

A continuación, se presenta una síntesis de la temática estudiada en el componente formativo:



Material complementario

Tema	Referencia	Tipo de material	Enlace del recurso
Paradigma orientado a objetos	EDteam. (2019). ¿Qué es la programación orientada a objetos? La mejor explicación en español.	YouTube	https://youtu.be/DlphYPc_HKk
Paradigma orientado a objetos	Soy Dalto. (2019). Programación orientada a objetos explicada en 10 minutos.	YouTube	https://youtu.be/uNlB7141umY

Glosario

Algoritmo: secuencia de pasos ordenados y finitos que describen un conjunto de acciones para resolver un problema.

Paradigma: ejemplo o modelo de algo. En el caso del paradigma orientado a objetos corresponde a un conjunto de características y formas en la que se representa la información.

Referencias bibliográficas

Aguilar, J. L. (2021). Fundamentos de programación. McGraw Hill Education.

Aguilar, L. J. (1998). Programación orientada a objetos. McGraw Hill Education.

Booch, G., Maksimchuk, R. A., Engle, M. W., Conallen, J., Young, B. J. & Houston, K. A. (2007). Object-oriented Analysis and Design with Applications. Addison-Wesley.

Rumbaugh, J., Jacobson, I., & Booch, G. (2004). El lenguaje unificado de modelado. Addison-Wesley.

Samuel. (2020). ¿Qué es la POO?. [Web log post] Fundamentos de las POO. http://micanalsamuelc.blogspot.com/2017/03

Créditos

Nombre	Cargo	Centro de Formación y Regional
Milady Tatiana Villamil Castellanos	Líder del Ecosistema	Dirección General
Olga Constanza Bermudez Jaimes	Responsable de Línea de Producción	Centro de Servicios de Salud - Regional Antioquia
Zulema Yidney León Escobar	Experta temática	Centro de Teleinformática y Producción Industrial - Regional Cauca
Jonathan Guerrero Astaiza	Experto temático	Centro de Teleinformática y Producción Industrial - Regional Cauca
Ana Catalina Córdoba Sus	Evaluadora Instruccional	Centro de Servicios de Salud - Regional Antioquia
Carlos Julián Ramírez Benítez	Diseñador de Contenidos Digitales	Centro de Servicios de Salud - Regional Antioquia
Edgar Mauricio Cortés García	Desarrollador Fullstack	Centro de Servicios de Salud - Regional Antioquia
Edgar Mauricio Cortés García	Actividad Didáctica	Centro de Servicios de Salud - Regional Antioquia
Luis Gabriel Urueta Álvarez	Validador de Recursos Educativos Digitales	Centro de Servicios de Salud - Regional Antioquia
Margarita Marcela Medrano Gómez	Evaluador para Contenidos Inclusivos y Accesibles	Centro de Servicios de Salud - Regional Antioquia
Daniel Ricardo Mutis Gómez	Evaluador para contenidos inclusivos y accesibles	Centro de Servicios de Salud - Regional Antioquia