Taller de conceptos y principios de programación orientada a objetos.

GA4-220501095-AA2-EV01.

Aprendiz:

Valentina Vargas Sanchez

Angela Gualdron Dulcey Henry

Andres Morales Garzón

Instructor:

Yerman Augusto Hernández Saenz

CENTRO DE SERVICIOS FINANCIEROS

SENA- REGIONAL DISTRITO CAPITAL

ANALISIS Y DESARROLLO DEL SOFTWARE

FICHA: 2627062

2023

Introducción

La Programación Orientada a Objetos es un enfoque de programación en el cual el código se organiza en clases y objetos. Permite diseñar aplicaciones complejas de manera más sencilla y cercana a la vida real. Sus características incluyen la abstracción, encapsulamiento, herencia, polimorfismo, cohesión y acoplamiento. Las clases son plantillas para crear objetos. Los objetos tienen métodos para realizar acciones y los eventos permiten la comunicación entre objetos y usuarios. Los atributos son propiedades de los objetos. En resumen, la Programación Orientada a Objetos es un enfoque poderoso para modelar el mundo real en código y facilitar el desarrollo y mantenimiento de aplicaciones.

Programación Orientada a Objetos.

Es una forma especial de programar, en que se organiza el código en unidades denominadas clases, en las cuales se crean objetos que se relacionan entre sí para lograr un objetivo. Se podría decir que esta forma de programar es una forma cercana a como expresaríamos la cosas en la vida real, permite diseñar la aplicación más complejas sin que el código sea complicado. Sus características son: Abstracción, encapsulamiento, cohesión, acoplamiento, herencia y polimorfismo.

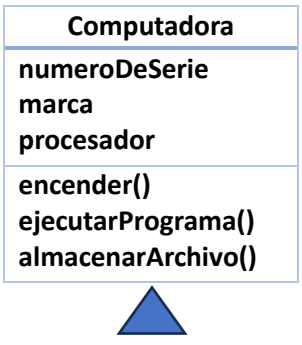
Clases y Objetos.

Los objetos se crean a partir de las clases y las clases sirven como platillas para crear objetos. Ejemplo: Cuando se piensa en la palabra “Persona”, no se refiere a una persona en particular, solo a la idea de la persona. Una persona tiene un nombre, una edad, etc. Esa es la clase, la plantilla.

Ahora cuando se piensa en personas específicas como tu mama, papa, hermanos, etc. Esos son objetos. Para cada persona se puede llenar la plantilla de datos: nombre, edad, etc.

Herencia.

La herencia nos permite reutilizar el código en cada clase heredando o extendiendo características de un objeto a sus “descendientes”. Por ejemplo: Una laptop sigue siendo una computadora que tienes todas las funciones, pero podríamos agregar una característica como la duración de la batería. En este caso una laptop sigue siendo una computadora, tiene todos sus atributos y métodos, pero se agrega un atributo y un método a la definición original.



Laptop
duracionBateria
asignarPrestamo()

Métodos.

Son aquellas funciones que permite manejar el objeto y que no rinden un tipo de servicio durante el transcurso del programa. Determinan también como van a responder el objeto cuando recibe un mensaje.

Por ejemplo: Una clase “Automóvil” podría tener métodos como “Encender”, “Apagar” y “Acelerar”. Cada objeto “Automóvil” tendría acceso a estos métodos, permitiéndoles realizar acciones específicas.

Eventos.

Es una acción que se desencadena como resultado de una interacción entre un objeto y el usuario o sistema. Es decir, que un evento es una respuesta a una acción. Por ejemplo: Cuando un usuario hace clic en un botón en una aplicación, se produce un evento.

Son importante ya que hace que los objetos se comuniquen entre si y con el usuario. Los eventos pueden ser utilizados para hacer que una aplicación responda a acciones de un usuario o para notificar al usuario de cierto cambios o estados.

Atributos.

Son propiedades que pueden asumir los objetos dentro de una clase. Los objetos de una clase tienen los mismos atributos, pero sus valores pueden diferir.

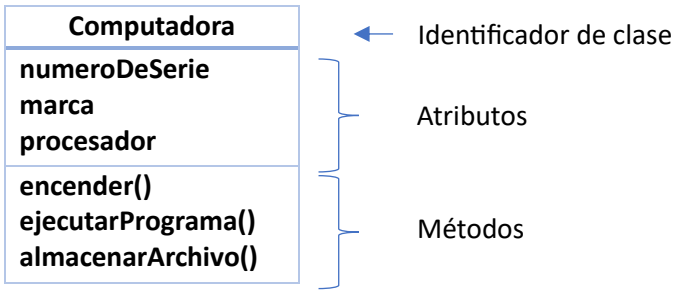
Los atributos son descripciones de datos que son comunes a los objetos de una clase. Por ejemplo: “Tamaño”, “Peso”, o “Color”. Se pueden asignar valores a estos atributos como el atributo “Color” puede tener el valor “Azul” o “Magenta”. Todos los objetos de una clase tienen los mismos atributos, pero pueden cambiar el valor. El siguiente objeto puede tener el valor “Rojo” para el atributo “Color”.

Abstracción.

La abstracción se trata de identificar las características claves. Se enfoca en lo externo de los elementos lo permite conocer su comportamiento esencial de su implementación, es decir que hay un énfasis en el “¿Qué es?” y “Que hace” algo y no en cómo debe implementarse o cómo funciona algo.

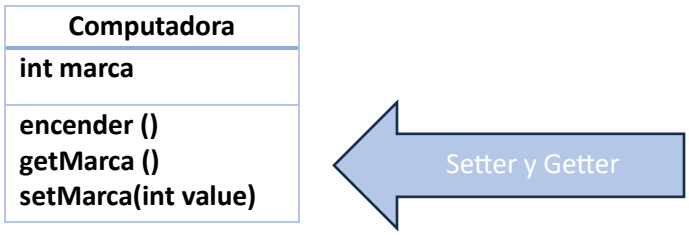
Se representa mediante clases que contienen métodos y atributos. Ejemplo:

Las características de un equipo de cómputo son: La marca, color, número de serie, cantidad de memoria instalada, capacidad del disco duro, etc.



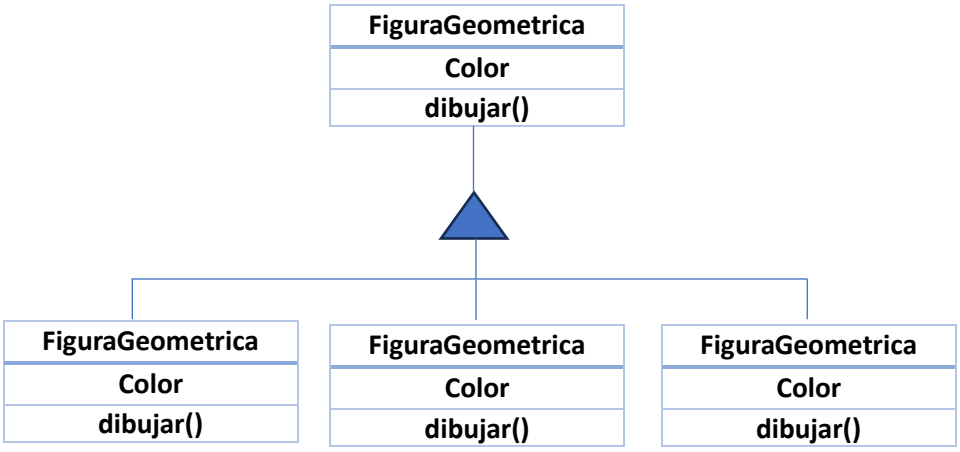
Encapsulamiento.

Permite agrupar datos y operaciones que se relaciones bajo una misma lógica, sirve para no trabajar los datos como elementos independientes y dispersos. También acostumbra a proteger la información o el estado de los atributos para que no se puedan ver o modificar sin el mecanismo adecuado. Para ellos se usa métodos para recuperar información (getters) y asegurar que la información proporcionada sea consistente con el objeto, y poder asignar (setters) un nuevo valor y verificar que no afecte la integridad del objeto.



Polimorfismo.

Es la característica que permite que una abstracción tome diferentes formas o comportamientos según el contexto. Si las clases de una superclase tienen un método con la misma definición, reaccionaran de la manera adecuada cuando reciban el mismo mensaje. Ejemplo:



Conclusiones.

En conclusión, la Programación Orientada a Objetos es una metodología poderosa que nos permite organizar el código en unidades llamadas clases, crear objetos que se relacionan entre sí y diseñar aplicaciones de manera más intuitiva y eficiente. Sus características como la abstracción, encapsulamiento, herencia, polimorfismo, cohesión y acoplamiento nos brindan herramientas para modelar el mundo real en código y facilitar el desarrollo y mantenimiento de software. La POO nos permite reutilizar código, mejorar la estructura del programa y fomentar la comunicación entre objetos, creando aplicaciones más flexibles y manejables.