Instituto Nacional de Aprendizaje

Programación de Aplicaciones Informáticas

Módulo: Programación Orientada a Objetos

Portafolio 04

Conceptos de Programación Orientada a Objetos

Profesor: Luis Alonso Bogantes.

Estudiante: Mariel Rojas

*Fecha: 10/05/2023*

# Objetos

Los objetos son un ente indeterminado, que se crean a partir de clases, y las clases encapsula datos y comportamientos relacionados. A los objetos también se les conoce como instancias de clase, y una clase básicamente es una plantilla o molde para crear objetos y define las propiedades y métodos que los objetos de esa clase tendrán.

(Los objetos pueden interactuar entre sí y con el entorno que los rodea mediante la ejecución de métodos, que son funciones definidas en la clase. Los métodos pueden modificar los datos del objeto o retornar un resultado que depende de los datos del objeto.)

# Propiedades

Una propiedad es un valor que pueden tener diferentes tipos de datos, como números, texto, booleanos, fechas, entre otros, estos valores se encuentran asociado a un objeto determinado y describen alguna característica de ese objeto.

Una propiedad es una variable que se define dentro de una clase y que se asocia a cada instancia de esa clase (objeto).

# Atributos

Un atributo se relaciona a una variable que se define dentro de una clase y se asocia a cada instancia de esa clase (objeto describen las características o propiedades de los objetos que se crearán a partir de la clase y pueden tener diferentes tipos de datos, como números, texto, booleanos, fechas, entre otros.

## Diferencia entre un Atributo y una Propiedad

La principal diferencia entre un atributo y una propiedad es que la propiedad puede encapsular un atributo y/o proporcionar una funcionalidad adicional, mientras que un atributo solo define una variable dentro de una clase. Sin embargo, en algunos contextos, los términos atributo y propiedad pueden ser utilizados indistintamente para referirse a una variable asociada a un objeto.

# Estado

El estado se refiere a la condición actual de un objeto en un momento dado. El estado de un objeto es definido por los valores de sus atributos en un momento determinado, dichos atributos pueden tener diferentes tipos de datos, como números, texto, booleanos, fechas, entre otros.

El estado de un objeto puede cambiar a medida que se ejecutan las operaciones en el programa. Por ejemplo, si se crea un objeto "Coche" con un atributo "velocidad" que tiene un valor inicial de cero, su estado cambiará a medida que se acelera o se frena el coche. La velocidad del coche será el estado actual del objeto en cada momento.

# Métodos

Un método es una función que se define dentro de una clase y que puede ser invocada en una instancia de esa clase (objeto). Los métodos permiten a los objetos realizar tareas determinadas o realizar operaciones en sus propios datos. Un método puede acceder a los atributos del objeto y manipularlos según sea necesario. Los métodos también pueden tener parámetros y devolver valores como cualquier otra función.

# Clase

Una clase es una plantilla o un molde que se utiliza para crear objetos, dicha clase se encarga de describir las características y el comportamiento de un grupo de objetos relacionados. Especifica los atributos que tienen los objetos de esa clase y los métodos que pueden realizar esos objetos, cada vez que se crea un objeto de esa clase, se está creando una instancia de esa clase, que tiene sus propios valores de atributos y puede realizar los métodos definidos en la clase.

Por ejemplo, si se está desarrollando una aplicación para una tienda en línea, se podría definir una clase "Producto" que tenga atributos como "nombre", "precio", "descripción", "cantidad en stock", etc., y métodos como "añadir al carrito" y "eliminar del carrito".

# Instanciación

La instanciación se refiere al proceso de crear una instancia de una clase, es decir, la creación de un objeto a partir de una clase existente. Cuando se instancia una clase, se reserva memoria para almacenar los datos del objeto y se inicializan los valores de los atributos según se haya definido en la clase. La instanciación permite la creación de múltiples objetos de la misma clase, cada uno con sus propios valores de atributos y capacidad para realizar los métodos definidos en la clase.

# Evento

Un evento es una señal o notificación que se genera en un programa en respuesta a una acción específica, como un clic de ratón, una pulsación de tecla o la finalización de una tarea. Los eventos se generan por componentes u objetos del programa y se envían a otros componentes u objetos para que realicen una acción correspondiente

# Herencia

La herencia se puede definir como la creación de nuevas clases a partir de otras ya existentes. La idea detrás de la herencia es que una clase nueva (denominada clase derivada o subclase) puede heredar las características y el comportamiento de una clase existente (denominada clase base o superclase), y luego agregar o modificar su propio comportamiento.

Cuando se define una clase derivada, se especifica qué clase base se va a utilizar como punto de partida. La clase derivada puede heredar todas las propiedades y métodos de la clase base, incluyendo sus atributos y métodos públicos, protegidos y privados. La clase derivada también puede agregar sus propios atributos y métodos, y redefinir o modificar los existentes.

La herencia también permite el polimorfismo, que es la capacidad de las clases derivadas de ser tratadas como objetos de la clase base. Esto significa que se pueden crear métodos que aceptan como argumento objetos de la clase base, pero que pueden trabajar con cualquier objeto derivado de esa clase base. Esto hace que el código sea más flexible y fácil de mantener, ya que se pueden agregar nuevas clases derivadas sin tener que modificar el código existente.

# Abstracción

La abstracción se describe como capacidad de separar los detalles complejos de implementación de una entidad, para que solo se exponga su funcionalidad esencial y relevante. Es decir, se trata de mostrar solo los aspectos importantes de un objeto o entidad y ocultar los detalles internos que no son relevantes para su uso.

# Encapsulamiento

El encapsulamiento se especifica como la técnica de ocultar la complejidad interna de un objeto y restringir el acceso directo a sus datos y métodos. El objetivo del encapsulamiento es lograr evitar que los datos sean modificados accidentalmente o mal utilizados, promueve el modularidad y la reutilización del código.

Al utilizar el encapsulamiento, se pueden crear objetos que son más seguros y confiables, ya que se limita el acceso directo a sus datos y métodos. Además, el encapsulamiento facilita la reutilización del código, ya que los objetos encapsulados se pueden utilizar como componentes independientes en diferentes partes del programa.

El encapsulamiento se logra mediante el uso de modificadores de acceso, que son palabras clave que se utilizan para especificar qué partes de un objeto son visibles y accesibles desde fuera de la clase

# Ocultamiento

El ocultamiento se conoce como al principio de limitar el acceso a los detalles de implementación de una clase, de manera que solo se exponga una interfaz pública que permita interactuar con la clase de forma controlada. El ocultamiento se logra mediante el uso de la encapsulación y los modificadores de acceso, que permiten definir qué partes de una clase son visibles y accesibles desde fuera de la misma. Esto significa que los atributos y métodos internos de la clase se mantienen ocultos y solo pueden ser accedidos a través de la interfaz pública definida por la clase.

# Modificadores de acceso:

Público: los miembros públicos de una clase son visibles y accesibles desde cualquier parte del programa. Esto incluye los métodos que se pueden llamar desde fuera de la clase y los atributos que se pueden acceder y modificar directamente.

Protegido: los miembros protegidos de una clase son visibles y accesibles solo desde dentro de la clase y sus subclases. Esto permite que las subclases hereden y modifiquen los atributos y métodos de la clase base, sin permitir que otros objetos accedan directamente a ellos.

Privado: los miembros privados de una clase son visibles y accesibles solo desde dentro de la clase. Esto significa que los objetos externos no pueden acceder ni modificar directamente los atributos y métodos privados de la clase.

# Modularidad

Se entiende como la capacidad de dividir un programa en componentes más pequeños y manejables, conocidos como módulos, que pueden ser desarrollados, probados, modificados y reutilizados de manera independiente.

# Polimorfismo

Es la capacidad de objetos de distintas clases para responder al mismo mensaje o llamado de método, de manera que se comporten de manera similar, pero con una implementación específica para cada clase. En otras palabras, el polimorfismo permite que objetos de diferentes clases compartan una misma interfaz o comportamiento.

Diagrama

Descripción generada automáticamente

# Ejemplos :

Se crea una clase llamada "Persona" que tenga los siguientes atributos: nombre, apellido, edad, número de identificación.

Para crear un objeto "persona" a partir de esta clase, se crea un objeto "persona1" a partir de la clase "Persona" y a dicha persona se le proporcionan valores para cada uno de los atributos de la clase al momento de crear el objeto. Entonces básicamente seria así : persona1 = Persona( “Mariel”, “Rojas”, “23”, “208030487”)

## Herencia :

Se supone que se necesita una clase "Estudiante" que se hereda de la clase "Persona" y agrega un nuevo atributo "curso", entonces básicamente crea una subclase Estudiante a partir de la clase persona. Luego se crea un objeto "estudiante1" de la clase "Estudiante" y accedimos tanto a los atributos de la superclase "Persona" como a los de la subclase "Estudiante”.

estudiante1 = Estudiante( “Mariel”, “Rojas”, “23”, “208030487”,”Programación Orientada a Objetos”)