PRINCIPIOS DE LA PROGRAMACIÓN ORIENTADA A OBJETOS:

1. Encapsulamiento: protección de datos y ocultación de información

El encapsulamiento es el principio que permite proteger los datos y ocultar la información interna de un objeto, de manera que solo se pueda acceder a ellos a través de métodos públicos y seguros.

Los datos y métodos privados de un objeto no están disponibles para otros objetos, lo que garantiza la integridad de la información y evita que sea alterada o corrompida accidental o intencionalmente.

Cada objeto es responsable de su propia información y de su propio estado, la única forma de que la información se pueda modificar es mediante los mismos métodos del objeto. por lo tanto, los atributos internos de un objeto deberían ser inaccesibles desde fuera, pudiéndolos modificar solo llamando a las funciones correspondientes.

Con esto conseguimos mantener a salvo el estado de usos indebidos que puedan ocasionar resultados inesperados.

2. Abstracción: simplificación y claridad de conceptos

La abstracción es el principio que permite simplificar y clarificar los conceptos y comportamientos complejos mediante la creación de modelos o representaciones abstractas y simplificadas.

Los modelos abstractos se centran en los aspectos esenciales y relevantes del objeto o sistema, y eliminan los detalles irrelevantes o confusos. De esta manera, se simplifica el diseño y se mejora la comprensión y mantenibilidad del programa.

Este principio se puede definir como las características específicas el objeto, los mismos que lo distinguen de los demás tipos de objetos,

La abstracción consiste en separar un elemento de su contexto o del resto de elementos que lo acompañan. Es un principio por el cual se descarta toda aquella información que no resulta relevante en un contexto en particular, enfatizando en alguno de los detalles o propiedades de los objetos. Depende principalmente del interés del observador, permitiendo abstracciones muy diferentes de la misma realidad.

3. Herencia: reutilización de código y especialización de objetos

La herencia es el principio que permite crear nuevas clases a partir de clases existentes, reutilizando el código y los comportamientos de sus ancestros. La nueva clase se conoce como subclase o derivada, mientras que la clase original se llama superclase o base.

La herencia permite crear objetos especializados a partir de objetos más generales, y añadir o modificar sus propiedades y comportamientos de manera independiente. Además, reduce la duplicación de código y aumenta la eficiencia y la legibilidad del programa.

4. Polimorfismo: flexibilidad y compatibilidad de objetos

En informática el polimorfismo es considerado uno de los elementos más importantes dentro la programación orientada a objetos POO, ya que su definición está fuertemente ligada a la Herencia.

El polimorfismo es el principio que permite a los objetos responder de diferentes maneras a un mismo mensaje o método, según su tipo o contexto. Es decir, un objeto puede comportarse de forma distinta según la situación, sin necesidad de conocer su tipo específico.

El polimorfismo aumenta la flexibilidad y la compatibilidad de los objetos, ya que permite interactuar con ellos de manera genérica y predecible, sin tener que conocer los detalles internos de su implementación. También permite extender y modificar las funcionalidades del programa de forma modular y escalable.

Como definición el polimorfismo es la habilidad de un objeto de realizar una acción de diferentes maneras, utilizando métodos iguales que se implementen de forma diferente en varias clases.

COMPARATIVO ENTRE LA PROGRAMACION FUNCIONAL Y LA ORIENTADA A OBEJTOS:

La programación funcional y la programación orientada a objetos son dos paradigmas de programación muy populares y utilizados en el desarrollo de software. La programación funcional se centra en funciones y matemáticas evita el estado mutable, mientras que la programación orientada a objetos se centra en objetos que tienen propiedades y comportamientos.

Combinar estos dos paradigmas puede tener varios beneficios. En este artículo, exploraremos algunos de ellos.

1. Código más modular y reutilizable

La programación orientada a objetos se basa en la encapsulación, que implica agrupar datos y métodos relacionados en un objeto. Esto hace que el código sea más modular y reutilizable. La programación funcional también se enfoca en el modularidad, pero utiliza funciones puras para lograrlo. Las funciones puras no tienen efectos secundarios y siempre producen el mismo resultado para los mismos argumentos de entrada.

Al combinar la programación orientada a objetos y la programación funcional, podemos crear objetos que contienen funciones puras como métodos. Esto significa que podemos tener un código más modular y reutilizable que es fácil de probar y mantener.

2. Código más expresivo y legible

La programación funcional se enfoca en funciones que transforman datos de entrada en datos de salida. Esto significa que el código puede ser más expresivo y legible. La programación orientada a objetos también puede ser legible, pero puede requerir más código para lograr la misma funcionalidad. Cuando combinamos estos dos paradigmas, podemos tener un código más expresivo y legible que es fácil de entender y mantener.

3. Mejor manejo de estado

La programación funcional se enfoca en evitar el estado mutable, lo que significa que los datos no cambian después de que se crean. La programación orientada a objetos también puede manejar el estado, pero puede llevar a problemas de funcionamiento y concurrencia.

Al combinar la programación orientada a objetos y la programación funcional, podemos tener objetos que mantienen su estado, pero solo pueden cambiarlo a través de métodos que son funciones puras. Esto significa que podemos tener un mejor manejo del estado en nuestro código, lo que puede ayudar a evitar errores y hacer que nuestro código sea más seguro.

4. Mayor flexibilidad en el diseño de la arquitectura

La programación orientada a objetos y la programación funcional tienen diferentes enfoques en la forma en que se estructura el código. La programación orientada a objetos se centra en objetos que tienen propiedades y comportamientos, mientras que la programación funcional se enfoca en funciones puras que transforman datos de entrada en datos de salida.

Al combinar estos dos paradigmas, podemos tener una mayor flexibilidad en el diseño de la arquitectura de nuestro código.