

M4 Programación orientada a objetos FIN A

ACTIVIDAD 1

|  |  |
| --- | --- |
| **Tutor:** | **Victor Madera Hernández** |
| **Estudiante:** | **José Ramón Ibáñez Posadas** |
| **Matricula:** | **BNL098377** |

|  |  |
| --- | --- |
| Monterrey, Nuevo León | Viernes, 12 de Abril de 2024 |

INTRODUCCIÓN

En el mundo de la programación, existen varios paradigmas que guían la forma en que diseñamos y desarrollamos software. El paradigma secuencial se basa en la ejecución paso a paso de instrucciones, ideal para algoritmos simples y lineales, aunque carece de reutilización de código. Por otro lado, el paradigma modular divide el programa en módulos independientes y reutilizables, promoviendo la colaboración entre equipos y facilitando el mantenimiento del software. En contraste, la programación orientada a objetos modela el software en términos de objetos que interactúan entre sí, permitiendo la reutilización de código a través de la encapsulación, herencia y polimorfismo, lo que facilita la representación del mundo real en el software.

Es crucial comprender las definiciones de clase y objeto, así como sus componentes, como parte fundamental de la programación orientada a objetos. Un mapa mental puede ser una herramienta poderosa para visualizar estas relaciones de manera clara y concisa, facilitando la comprensión y el diseño de sistemas complejos.

Al aplicar estos conceptos a objetos específicos como el tren y la moto, es importante identificar sus atributos y métodos. Los atributos representan las características de un objeto, como el modelo, la velocidad máxima o el tipo de combustible, mientras que los métodos son las acciones que el objeto puede realizar, como arrancar, acelerar o frenar.

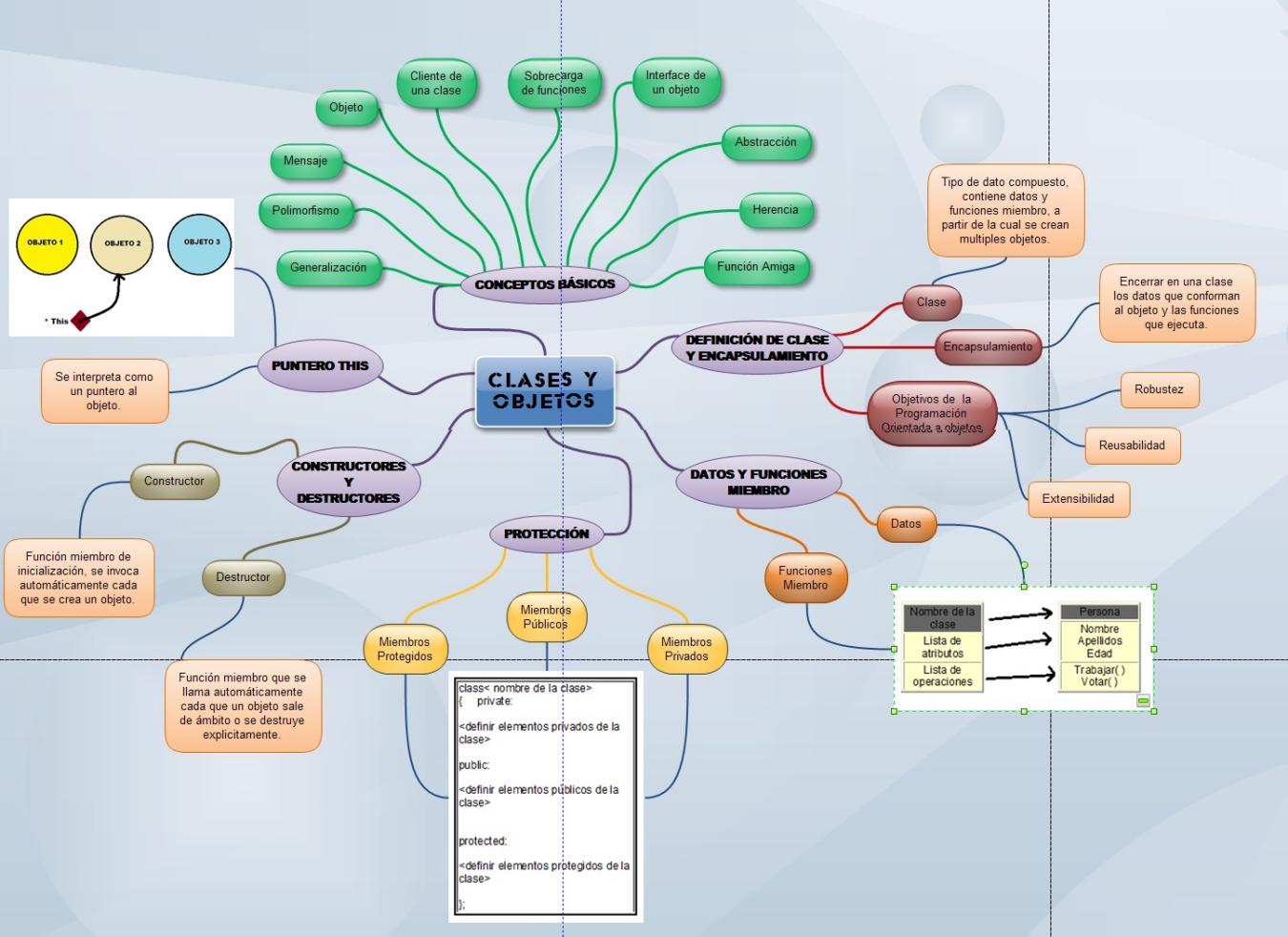
En resumen, conocer y entender los paradigmas de programación, así como los conceptos fundamentales de la programación orientada a objetos, junto con la aplicación práctica de atributos y métodos en objetos específicos, son elementos esenciales para desarrollar software robusto y eficiente en la actualidad.

DESARROLLO

CUADRO COMPARATIVO

|  |  |  |  |
| --- | --- | --- | --- |
| **Paradigma** | **Función** | **Características** | **Ventajas** |
| **Paradigma Secuencial** | Secuencialmente ejecutar instrucciones | - Ejecución línea por línea | - Fácil de entender y seguir el flujo de ejecución |
|  |  | - Basado en instrucciones y flujo de control lineal | - Útil para algoritmos simples y procedimientos lineales |
|  |  | - No hay concepto de reutilización de código | - Ideal para situaciones donde se requiere un proceso paso a paso sin complejidad estructural |
| **Paradigma Modular** | Dividir el programa en módulos | - Módulos independientes y reutilizables | - Facilita la reutilización de código y el mantenimiento del software |
|  |  | - Abstracción de funcionalidades en módulos | - Permite la colaboración entre equipos de desarrollo, cada uno trabajando en un módulo específico |
|  |  | - Encapsulamiento de datos y funciones en módulos | - Facilita la comprensión del código al dividirlo en partes más pequeñas y manejables |
| **Paradigma Orientado a Objetos** | Modelar el software como objetos | - Clases y objetos como entidades principales | - Reutilización de código a través de la herencia y la composición |
|  |  | - Encapsulamiento de datos y comportamiento en objetos | - Facilita el diseño y la representación del mundo real en el software |
|  |  | - Herencia y polimorfismo para compartir y extender | - Promueve la modularidad y la flexibilidad en el diseño del software |
|  |  | - Abstracción de datos y comportamiento en clases | - Mejora la mantenibilidad y la escalabilidad del software al dividirlo en componentes cohesivos |

MAPA MENTAL



TABLA

|  |  |
| --- | --- |
| **Objeto: Moto** |  |
| **Atributos** | **Descripción** |
| *Marca* | *La marca de la moto (por ejemplo, Yamaha, Honda, etc.)* |
| *Modelo* | *El modelo específico de la moto (por ejemplo, MT-07, CBR600RR, etc.)* |
| *Color* | *El color de la moto* |
| *Velocidad máxima* | *La velocidad máxima que puede alcanzar la moto en km/h* |
| *Precio* | *El precio de la moto en la moneda correspondiente* |
| **Métodos** | **Descripción** |
| *Arrancar* | *Permite encender la moto* |
| *Acelerar* | *Aumenta la velocidad de la moto* |
| *Frenar* | *Reduce la velocidad de la moto* |
| *Apagar* | *Apaga la moto* |
| **Objeto: Tren** |  |
| **Atributos** | **Descripción** |
| *Modelo* | *El modelo específico del tren (por ejemplo, TGV, Shinkansen, etc.)* |
| *Número de vagones* | *La cantidad de vagones que tiene el tren* |
| *Capacidad de pasajeros* | *La cantidad máxima de pasajeros que puede transportar el tren* |
| *Velocidad máxima* | *La velocidad máxima que puede alcanzar el tren en km/h* |
| *Tipo de combustible* | *El tipo de combustible que utiliza el tren (por ejemplo, diésel, eléctrico, etc.)* |
| **Métodos** | **Descripción** |
| *Arrancar* | *Pone en marcha el tren* |
| *Acelerar* | *Aumenta la velocidad del tren* |
| *Frenar* | *Reduce la velocidad del tren* |
| *Cargar pasajeros* | *Permite que los pasajeros suban al tren* |
| *Descargar pasajeros* | *Permite que los pasajeros bajen del tren* |

CONCLUSIÓN

En conclusión, podemos destacar tres ideas principales de lo mencionado anteriormente:

1. Flexibilidad y modularidad: Los paradigmas de programación, ya sea secuencial, modular u orientado a objetos, ofrecen diferentes enfoques para el diseño de software. La modularidad permite dividir el programa en componentes independientes y reutilizables, lo que facilita la colaboración entre equipos y el mantenimiento del código a largo plazo.

2. Abstracción y representación del mundo real: La programación orientada a objetos, en particular, permite modelar el software en términos de objetos que reflejan entidades del mundo real. Este enfoque proporciona una forma intuitiva y eficaz de representar y manipular conceptos complejos, facilitando la comprensión y el desarrollo de sistemas de software.

3. Importancia de los conceptos fundamentales: Comprender las definiciones de clase, objeto, atributos y métodos es esencial para la programación orientada a objetos. Estos conceptos forman la base sobre la cual se construyen los sistemas de software, y su dominio permite a los desarrolladores crear soluciones más robustas y escalables.

En resumen, al combinar los diferentes paradigmas de programación con un sólido entendimiento de los conceptos fundamentales de la programación orientada a objetos, los desarrolladores pueden diseñar y desarrollar software de manera más eficiente, adaptable y orientada a las necesidades del mundo real.

BIBLIOGRAFÍA

