**Laboratorio de Programación Orientada a Objetos**

Práctica 9:

UML.



#### Equipo No:\_\_3\_\_

#### Integrantes:

|  |  |
| --- | --- |
| N.L. | Nombre |
| 1 | Antonio Martínez Rodrigo. |
| 3 | Briseño Vázquez Angel Geovany. |
| 8 | Nishimura Guerrero Christian Jesús. |
|  |  |

#### Fecha de realización:

11/05/2023

Práctica 9. UML

**PROBLEMAS PARA PRESENTAR:**

Realizar los siguientes diagramas: Diagrama de Clases, Diagrama de Objetos, Diagrama de Casos de Uso, de los siguientes programas:

Texto

Descripción generada automáticamente con confianza media

**Diagrama de clases:**

**Texto

Descripción generada automáticamente**

**Diagrama de objetos:**

**Texto

Descripción generada automáticamente**

**Diagrama de casos de uso:**

**Interfaz de usuario gráfica, Texto

Descripción generada automáticamente**

Texto

Descripción generada automáticamente

**Diagrama de clases:**

**Imagen que contiene Texto

Descripción generada automáticamente**

**Diagrama de objetos:**

**Interfaz de usuario gráfica, Texto

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto

Descripción generada automáticamente**

**Diagrama de casos de uso:**

**Texto

Descripción generada automáticamente**

*Tabla

Descripción generada automáticamente con confianza baja*

**Diagrama de clases:**

**Texto

Descripción generada automáticamente**

**Diagrama de objetos:**

**Texto

Descripción generada automáticamente**

**Diagrama de casos de uso:**

**Texto

Descripción generada automáticamente**

*Tabla

Descripción generada automáticamente*

**Diagrama de clases:**

**Diagrama, Texto

Descripción generada automáticamente**

**Diagrama de objetos:**

**Texto

Descripción generada automáticamente**

**Diagrama de casos de uso:**

**Texto

Descripción generada automáticamente**

**Tabla

Descripción generada automáticamente**

**Diagrama de clases:**

**Texto

Descripción generada automáticamente**

**Diagrama de objetos:**

**Texto

Descripción generada automáticamente**

**Diagrama de casos de uso:**

**Texto

Descripción generada automáticamente**

**Interfaz de usuario gráfica, Texto

Descripción generada automáticamente**

**Diagrama de clases:**

**Imagen que contiene Diagrama

Descripción generada automáticamente**

**Diagrama de objetos:**

**Texto

Descripción generada automáticamente**

**Diagrama de casos de uso:**

**Texto

Descripción generada automáticamente**

**Texto, Tabla

Descripción generada automáticamente**

**Diagrama de clases:**

**Texto

Descripción generada automáticamente**

**Diagrama de objetos:**

**Diagrama

Descripción generada automáticamente**

**Diagrama de casos de uso:**

**Texto

Descripción generada automáticamente**

**CONCLUSIONES INDIVIDUALES:**

Antonio Martínez Rodrigo.

Los diagramas de clases, objetos y casos de uso son herramientas valiosas para la programación ya que nos permiten visualizar y entender mejor la estructura y el comportamiento de nuestro software. Estos diagramas nos ayudan a:

Comprender mejor el problema: los diagramas nos permiten descomponer el problema en partes más pequeñas y comprensibles, lo que nos ayuda a entenderlo mejor y a identificar los requisitos del sistema.

Diseñar soluciones: los diagramas nos permiten diseñar soluciones y modelar las interacciones entre los componentes del sistema.

Comunicar las ideas: los diagramas son una forma efectiva de comunicar nuestras ideas a otros miembros del equipo, ya que proporcionan una representación visual de la estructura y el comportamiento del sistema.

Identificar errores: los diagramas nos permiten identificar errores o inconsistencias en el diseño antes de que se implemente el software, lo que puede ahorrar tiempo y recursos.

Facilitar la implementación: los diagramas pueden ayudar a los desarrolladores a entender mejor cómo deben implementar el software y a identificar los patrones de diseño que pueden ser útiles para resolver el problema.

Briseño Vázquez Angel Geovany.

Los diagramas son una herramienta visual importante en el proceso de desarrollo de software que nos permiten representar la estructura y el comportamiento de nuestro código de una manera más clara y concisa. En los ejercicios anteriores, utilizamos tres tipos de diagramas: diagramas de clases, diagramas de objetos y diagramas de casos de uso.

Los diagramas de clases nos permitieron representar la estructura de nuestro código, mostrando las clases, sus atributos y métodos, y las relaciones entre ellas. Este diagrama nos permitió entender cómo se relacionaban las clases, y cómo se implementaba el polimorfismo y la herencia en nuestro código.

Los diagramas de objetos nos permitieron representar los objetos específicos creados en nuestro código, mostrando sus atributos y valores actuales. Este diagrama nos permitió entender cómo se creaban y almacenaban los objetos en nuestro código, y cómo se relacionaban entre sí.

Por último, los diagramas de casos de uso nos permitieron representar los diferentes casos de uso de nuestro código, mostrando las acciones que los usuarios podían realizar y las interacciones entre ellos y el sistema. Este diagrama nos permitió entender cómo se utilizaba nuestro código desde el punto de vista del usuario, y cómo se relacionaban las diferentes funcionalidades entre sí.

Nishimura Guerrero Christian Jesús.

Los diagramas son herramientas útiles para planificar y diseñar software antes de codificarlo. Al utilizar diagramas, los programadores pueden visualizar la estructura y las relaciones entre diferentes componentes del software, lo que puede ayudar a identificar posibles problemas y errores antes de que se presenten en el código.

Los diagramas de clases, por ejemplo, pueden ayudar a definir la estructura de un programa y sus clases, métodos y atributos, lo que facilita la comprensión del código y su mantenimiento en el futuro. Los diagramas de objetos, por otro lado, son útiles para visualizar cómo los objetos interactúan entre sí en tiempo de ejecución y pueden ayudar a identificar errores lógicos en el diseño del software.

Los diagramas de casos de uso, por su parte, son útiles para comprender las funcionalidades del software desde el punto de vista del usuario y pueden ayudar a asegurarse de que el software cumpla con las necesidades y expectativas de los usuarios.

**CONCLUSIONES POR EQUIPO:**

El UML (Lenguaje Unificado de Modelado) es una herramienta de modelado visual que se utiliza para representar sistemas de software utilizando elementos gráficos y texto. El UML es ampliamente utilizado en la programación orientada a objetos (POO) porque permite a los desarrolladores visualizar y diseñar sus sistemas de software de manera más clara y concisa.

Los diagramas UML son importantes para la programación orientada a objetos por varias razones. Primero, los diagramas UML permiten a los desarrolladores visualizar y comprender la estructura y el comportamiento de un sistema de software. Al proporcionar una vista gráfica de las clases, sus atributos, métodos y relaciones, los desarrolladores pueden entender mejor cómo funciona el sistema de software y cómo interactúan las diferentes partes del sistema.

En segundo lugar, los diagramas UML permiten a los desarrolladores comunicarse con otros miembros del equipo de desarrollo de software, como diseñadores, arquitectos, programadores y probadores. Al utilizar un lenguaje visual común, los miembros del equipo pueden entender y discutir el sistema de software de manera más efectiva y eficiente.

En tercer lugar, los diagramas UML pueden ser utilizados para documentar y mantener sistemas de software. Los diagramas UML pueden ser utilizados para describir la estructura y el comportamiento de un sistema de software, lo que puede ser útil para futuros desarrolladores que necesiten entender el sistema o para los desarrolladores que necesiten actualizar o modificar el sistema.

En conclusión, el uso de diagramas UML es importante en la programación orientada a objetos porque permite a los desarrolladores visualizar, diseñar y comunicar sistemas de software de manera más clara y concisa. Los diagramas UML también pueden ser utilizados para documentar y mantener sistemas de software, lo que es esencial para el desarrollo de software a largo plazo.