Nombre: Jose Javier Bonilla Salazar
Carnet: 202200035
Clase: Introducción a la programación
Profesor: Brayan Prado Marroquín

Investigación Tarea 2

Contenido

Introducción 3

1. Marco teórico 4

1.1. Recursividad 4

1.2. Programación orientada a objetos 4

1.3. Pilares de la programación orientada a objetos 5

1.3.1. Encapsulación 5

1.3.2. Abstracción 5

1.3.3. Herencia. 6

1.3.4. Polimorfismo 6

1.4. Diagrama de clases 6

1.4.1. Sección superior 7

1.4.2. Sección central 7

1.4.3. Sección inferior 7

1.5. Relaciones entre clases 7

1.5.1. Asociación 7

1.5.2. Composición 8

1.5.3. Herencia 8

1.6. Clase padre e hijo 9

1.6.1. Superclase o clase padre 9

1.6.2. Clase derivada o clase hija 9

Conclusiones 10

Bibliografía 11

Introducción

Se adentrará en el concepto de la programación orientada a objetos, indagando en sus conceptos más básicos hasta los más complejos. Todo esto con el fin de dar a entender que es ese famoso paradigma de la programación tan usado.

1. Marco teórico

1.1. Recursividad

La recursividad consiste en funciones que se llaman a si mismas, evitando el uso de bucles y otros iteradores.

Otra cosa importante a tener en cuenta es que, cada vez que se hace una llamada a una función desde otra función (aunque sea a sí misma), se crea una nueva entrada en la pila de llamadas del intérprete. Ésta tiene un espacio limitado por lo que puede llegar un punto en el que si se hacen demasiadas se sature y se produzca un error. A este error se le denomina "Desbordamiento de pila" o "Stack Overflow". Ahora ya sabes de donde viene el nombre del famoso sitio para dudas de programadores sin el que la programación moderna no sería posible ;-)

Además, hay que tener mucho cuidado con la condición de parada. Ésta se refiere a la condición que se debe comprobar para determinar que ya no se harán más llamadas a la función. Es en ese momento en el que empiezan a devolverse los valores hacia "arriba", retornando a la llamada original.

Los operadores relacionales comparan datos numéricos, de serie de caracteres o lógicos. El resultado de la comparación ya sea Verdadero (1) o falso (0), puede utilizarse para tomar una decisión referente al flujo del programa.

(Alarcom, 2016)

1.2. Programación orientada a objetos

La programación orientada a objetos (Object Oriented Programming, OOP) es un modelo de programación informática que organiza el diseño de software en torno a datos u objetos, en lugar de funciones y lógica. Un objeto se puede definir como un campo de datos que tiene atributos y comportamiento únicos.

La programación orientada a objetos se centra en los objetos que los desarrolladores quieren manipular en lugar de enfocarse en la lógica necesaria para manipularlos. Este enfoque de programación es adecuado para programas que son grandes, complejos y se actualizan o mantienen activamente.

La organización de un programa orientado a objetos también hace que el método sea beneficioso para el desarrollo colaborativo, donde los proyectos se dividen en grupos.

Los beneficios adicionales de la programación orientada a objetos incluyen la reutilización, la escalabilidad y la eficiencia del código. Incluso cuando se utilizan microservicios, los desarrolladores deben seguir aplicando los principios de la programación orientada a objetos.

El primer paso en OOP es recopilar todos los objetos que un programador desea manipular e identificar cómo se relacionan entre sí, un ejercicio que a menudo se conoce como modelado de datos.

Los ejemplos de un objeto pueden variar desde entidades físicas, como un ser humano que se describe por propiedades como nombre y dirección, hasta pequeños programas informáticos, como widgets.

Una vez que se conoce un objeto, se etiqueta con una clase de objetos que define el tipo de datos que contiene y cualquier secuencia lógica que pueda manipularlo. Cada secuencia lógica distinta se conoce como método. Los objetos pueden comunicarse con interfaces bien definidas llamadas mensajes.

(TechTarget, 2021)

1.3. Pilares de la programación orientada a objetos

1.3.1. Encapsulación

La implementación y el estado de cada objeto se mantienen de forma privada dentro de un límite definido o clase. Otros objetos no tienen acceso a esta clase o la autoridad para realizar cambios, pero pueden llamar a una lista de funciones o métodos públicos. Esta característica de ocultación de datos proporciona una mayor seguridad al programa y evita la corrupción de datos no intencionada.

1.3.2. Abstracción

Los objetos solo revelan mecanismos internos que son relevantes para el uso de otros objetos, ocultando cualquier código de implementación innecesario. Este concepto ayuda a los desarrolladores a realizar cambios y adiciones más fácilmente a lo largo del tiempo.

1.3.3. Herencia.

Se pueden asignar relaciones y subclases entre objetos, lo que permite a los desarrolladores reutilizar una lógica común sin dejar de mantener una jerarquía única. Esta propiedad de OOP obliga a un análisis de datos más completo, reduce el tiempo de desarrollo y asegura un mayor nivel de precisión.

1.3.4. Polimorfismo

Los objetos pueden adoptar más de una forma según el contexto. El programa determinará qué significado o uso es necesario para cada ejecución de ese objeto, reduciendo la necesidad de duplicar código.

(TechTarget, 2021)

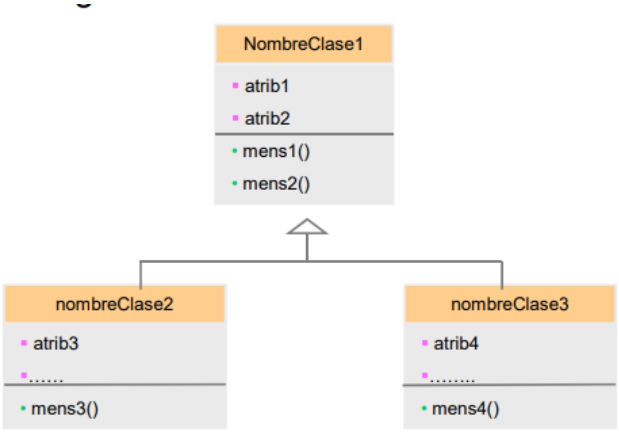
1.4. Diagrama de clases

Un diagrama de clases es un diagrama de estructura estática que describe la estructura de un sistema. Permite modelar sus clases, atributos, operaciones y relaciones entre objetos.

Un diagrama de clases describe:

- Los tipos de objetos en el sistema.
- Las relaciones estáticas que existen entre ellos.
- Los atributos y operaciones de las clases.
- Las restricciones a las clases y a sus asociaciones.

Figura 1. **Diagrama de clases**



Fuente: http://www.frlp.utn.edu.ar/materias/paradigmas/clase4-POO-UML.pdf

Componentes básicos de un diagrama de clases

En UML las clases se representan mediante un rectángulo que puede estar dividido en tres partes.

1.4.1. Sección superior

Contiene el nombre de la clase.

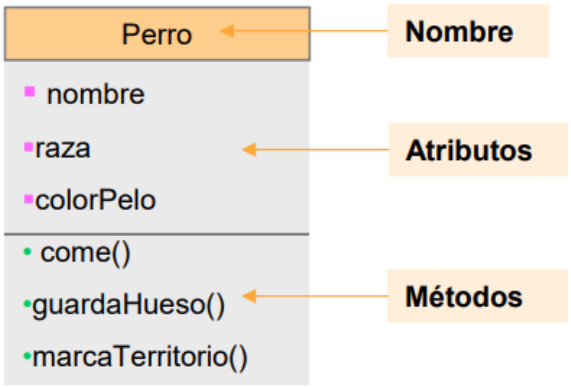
1.4.2. Sección central

Contiene los atributos de la clase.

1.4.3. Sección inferior

Incluye operaciones de clases (métodos), organizadas en un formato de lista. Cada operación requiere su propia línea.

Figura 2. **Diagrama de clase**



Fuente: http://www.frlp.utn.edu.ar/materias/paradigmas/clase4-POO-UML.pdf

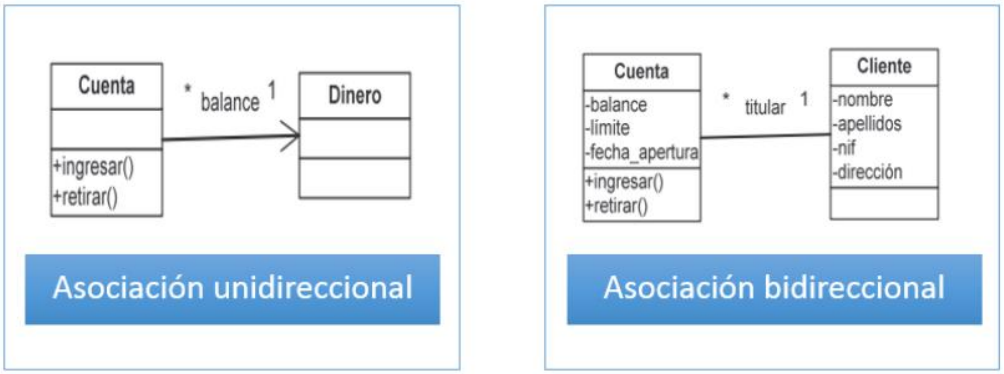
1.5. Relaciones entre clases

Las relaciones existentes entre las distintas clases indican como se están comunicando las clases entre sí. Una asociación es una conexión estructural simple entre clases. Las instancias de las clases implicadas en una asociación estarán probablemente comunicándose en el momento de ejecución.

1.5.1. Asociación

Es una relación estructural que describe una conexión entre objetos. Suelen ser bidireccionales, pero es importante a veces hacerlas unidireccionales para restringir su navegación en un solo sentido.

Figura 2. **Diagrama de Asociación**

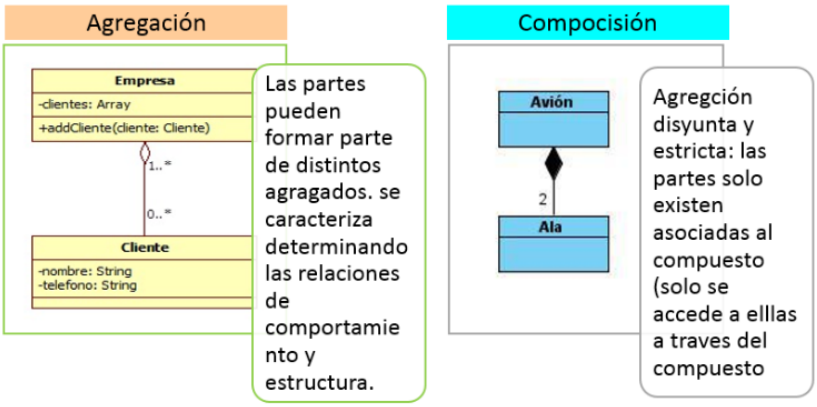


Fuente: https://ingenieriaensoftwarenathalyalava.wordpress.com/2015/07/01/148/#:~:text=Las%20relaciones%20existentes%20entre%20las,en%20el%20momento%20de%20ejecución.

1.5.2. Composición

Son casos particulares de asociación cuya relación esta entre un todo y sus partes, gráficamente es como una asociación con un rombo en los extremos.

Figura 2. **Diagrama de composición**



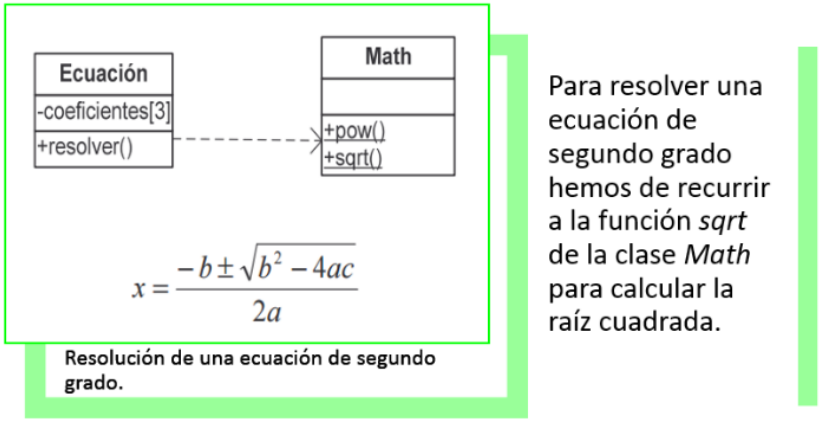
Fuente: https://ingenieriaensoftwarenathalyalava.wordpress.com/2015/07/01/148/#:~:text=Las%20relaciones%20existentes%20entre%20las,en%20el%20momento%20de%20ejecución.

1.5.3. Herencia

Relación más débil que una asociación que muestra la relación entre un cliente y el proveedor de un servicio usado por el cliente. Cliente es el objeto que solicita

un servicio y servidor es el objeto que provee un servicio. Gráficamente la dependencia se muestra como una línea discontinua con una punta de flecha que apunta del cliente al proveedor.
(Alava, 2015)

Figura 2. **Diagrama de herencia**



Fuente: https://ingenieriaensoftwarenathalyalava.wordpress.com/2015/07/01/148/#:~:text=Las%20relaciones%20existentes%20entre%20las,en%20el%20momento%20de%20ejecución.

1.6. Clase padre e hijo

1.6.1. Superclase o clase padre

Es la clase de la cual se hereda. Por ejemplo, la clase Object es la clase padre de la que heredan el resto de clases en java. Incluso aquellas que nosotros creamos de forma personalizada.

1.6.2. Clase derivada o clase hija

Es aquella que hereda de la clase padre. Además, también puede ser clase padre de otras. Esta clase, no solo heredará las características y propiedades de la clase padre, sino también sus métodos constructores.

Conclusiones

La programación orientada a objetos nos brinda una mejor forma de estructurar el código, evitando las repeticiones innecesarias en este. Además de que genera un código más limpio y fácil de leer.

Bibliografía

Alarcom, J. (2016, 27 enero). *¿Qué es la recursividad o recursión? - Un ejemplo con JavaScript*. campusMVP.es. Recuperado 18 de septiembre de 2022, de https://www.campusmvp.es/recursos/post/Que-es-la-recursividad-o-recursion-Un-ejemplo-con-JavaScript.aspx

en-c/funciones-utiles-de-cadenas-l17932

TechTarget, C. de. (2021, 17 mayo). *Programación orientada a objetos, OOP*. ComputerWeekly.es. Recuperado 18 de septiembre de 2022, de https://www.computerweekly.com/es/definicion/Programacion-orientada-a-objetos-OOP

Alava, N. (2015, 25 julio). *Relaciones entre clases*. Ingenieria en Software. Recuperado 18 de septiembre de 2022, de https://ingenieriaensoftwarenathalyalava.wordpress.com/2015/07/01/148/#:%7E:text=Las%20relaciones%20existentes%20entre%20las,en%20el%20momento%20de%20ejecuci%C3%B3n.