PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

INGENIERÍA EN SISTEMAS

PROGRAMACIÓN ORIENTADA A OBJETOS

CUARTO NIVEL

HERENCIA EN PROGRAMACIÓN ORIENTADA A OBJETOS

INTEGRANTES:

- **Coronel Paula**
- **Israel Hernández**
- **Daniela Pozo**
- **Gabriel Vásconez**
- **Michael (Melissa) Molina**

Herencia en Programación Orientada a Objetos

Introducción

En la programación orientada a objetos, la herencia ayuda significativamente a lograr un código eficiente y fácil de realizar. Una clase puede copiar características importantes como los métodos y atributos de otra, y esto evita la repetición de un código repetitivamente. Esto les da opción a los desarrolladores basándose en otras ya existentes a crear nuevas a crear unas nuevas, así ahorran tiempo, recursos y esfuerzo, lo que es muy importante para un programador.

1. ¿Qué es la Herencia en P.O.O.?

La herencia es como una forma de dar iguales características que tiene una clase a otra nueva. Pero en lugar de copiar y pegar el código, lo que hace es "heredar" esos atributos y métodos, y luego puedes agregar o modificar lo que se necesite. Este mecanismo es súper útil porque te permite escribir menos código y hacer que tu programa sea más organizado. En lugar de tener que escribir las mismas funcionalidades una y otra vez, simplemente las heredas y las personalizas si es necesario.

2. Conceptos Clave de la Herencia

- **Superclase**: En estas otras clases van a heredar sus atributos y métodos. Se puede pensar en ella como una clase "base" o "principal".
- **Subclase**: Es la clase que hereda de una superclase. Las subclases pueden agregar nuevos atributos y métodos o modificar los que ya han heredado.
- **Reutilización**: La herencia ayuda mucho con la reutilización del código. En lugar de volver a escribir el mismo código una y otra vez para crear clases nuevas, puedes derivarlas de clases que ya están hechas, ahorrando tiempo y esfuerzo.

4. Tipos de Herencia

Existen dos tipos principales de herencia:

- **Herencia Simple**: En este tipo de herencia, una clase solo puede heredar de una clase base. Java solo permite este tipo de herencia. Es la forma más simple y clara de organizar las clases.
- **Herencia Múltiple**: Es cuando una clase puede heredar de más de una clase base. Permite la herencia múltiple en diferentes lenguajes de programación, pero Java no la permite ya que se podría generar problemas si las clases base tienen el mismo método o atributo. Lo que Java si permite es agregar varias interfaces, lo que da algo de flexibilidad sin los problemas de la herencia múltiple.

5. Modificadores de Acceso y Herencia

En Java, los modificadores de acceso definen quién puede ver o modificar los atributos y métodos de una clase. Estos modificadores son importantes cuando hablamos de herencia, ya que controlan lo que las subclases pueden o no acceder de las superclases.

- **private**: Estos solo pueden ser accedidos dentro de la misma clase. Para las subclases estos no son accesibles.
- **default** (sin modificador): Si no se especifica un modificador de acceso, el atributo o método, solo dentro del mismo paquete se puede acceder.
- **protected**: Los elementos protected pueden ser accedidos dentro de la misma clase, el mismo paquete, y también por las subclases, incluso si están en paquetes diferentes.
- **public**: Los elementos public son accesibles desde cualquier clase, sin importar su ubicación.

6. Ventajas de la Herencia

- Al realizar el código Se puede reutilizar el código que ya está escrito, esto ahorra tiempo y también evita que se duplique la lógica del código.
- La extensión de clases facilita cuando se necesite añadir más funcionali9dades se puede hacer de una manera más fácil sin tener que cambiar el código original.
- Por lo general se ordena jerárquicamente, lo que organiza las ideas y es más eficiente para entenderlo.

7. Desventajas de la Herencia

- **Complejidad**: Si las jerarquías de clases son demasiado grandes o complicadas, puede ser difícil entender cómo interactúan entre sí. Esto puede hacer que el código sea más difícil de mantener.
- **Dependencia entre clases**: Si cambias algo en una clase base (superclase), las subclases también se verán afectadas, lo que podría generar errores difíciles de rastrear.

8. Conclusión

La herencia es una de las herramientas más útiles en la programación orientada a objetos. Permite reutilizar código, facilitar la extensión de las clases y organizar mejor el código, todo lo cual ayuda a crear aplicaciones más limpias y fáciles de mantener. Aunque hay algunos inconvenientes, como la complejidad de las jerarquías profundas o la dependencia entre clases, sus beneficios superan estos problemas. Usada correctamente, la herencia hace que el desarrollo de software sea más rápido, eficiente y organizado.

9. Bibliografía:

Logali Group. (2018, 29 mayo). Herencia - Logali Group. *Logali Group*. https://logaligroup.com/herencia/

Platzi. (s. f.). *Herencia en POO*. https://platzi.com/tutoriales/1474-oop/9305-herencia-en-poo/

***III - Herencia*. (s. f.).** http://profesores.fi-b.unam.mx/carlos/java/java_basico3_4.html

***Blog IfGeekThen*. (s. f.). IfgeekthenNTTDATA.** https://ifgeekthen.nttdata.com/s/post/herencia-en-programacion-orientada-objetos-MCPV3PCZDNBFHSROCCU3JMI7UIJQ?language=es

colaboradores de Wikipedia. (2024, 22 noviembre). *Herencia (informática) - Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/wiki/Herencia_(inform%C3%A1tica)