Alumno: Rodolfo Nicolás Velasco Fessler

Trabajo Práctico 2

Mi Repositorio de GitHub: <https://github.com/RodolfoVelasco1/Programacion2-Com3/>

TP2: <https://github.com/RodolfoVelasco1/Programacion2-Com3/tree/master/src/TP2>

1. **¿Qué es la herencia en java? ¿Cuáles son sus beneficios?**

La herencia define una relación entre clases en la que una de ellas brinda la estructura de comportamiento definida en una o más clases.

La herencia permite a una clase (clase derivada) heredar atributos y métodos de otra clase (clase base). La clase base define la estructura y el comportamiento base, mientras que la clase derivada puede extender o modificar esa base.

1. **¿Cómo se representa la herencia en un diagrama UML?**

Se utiliza una flecha con una punta triangular vacía que va desde la clase derivada hacia la clase base.

1. **¿Cuándo es recomendable usar la herencia?**

- Cuando existe una relación "es-un" entre dos clases: una clase es una versión más específica de otra.

- Para implementar Polimorfismo.

- Cuando se quiere crear jerarquías de clases con especialización.

- Cuando se desea compartir código común entre clases relacionadas.

1. **¿Qué es una interfaz en java?**

Es un conjunto de métodos abstractos (sin implementación) que deben ser implementados por cualquier clase que implemente esa interfaz.

1. **¿Cómo se representa una interfaz en un diagrama UML?**

Se utiliza una línea con punta de flecha triangular vacía que va desde la clase que implementa la interfaz hacia la interfaz.

1. **¿Cuándo es recomendable usar una interfaz?**

- Cuando se quiere definir un conjunto de métodos que deben ser implementados por diferentes clases, pero no necesariamente comparten una relación de herencia.

- Para permitir que diferentes clases implementen el mismo comportamiento de manera diferente.

- Para definir contratos de comportamiento entre clases sin especificar la implementación.

1. **¿Cuál es la diferencia entre una interfaz y una clase abstracta?**

- Las interfaces sólo declaran métodos abstractos sin implementación, mientras que las clases abstractas pueden tener métodos sin implementación o con implementación.

- Una clase puede implementar varias interfaces, pero solo puede heredar de una sola clase abstracta.

1. **¿Qué tipos de relaciones pueden existir entre clases y objetos?**

Las relaciones más comunes son: Asociación, Agregación, Composición y Herencia.

1. **¿Cómo se representa cada relación en un diagrama UML?**

- Asociación: Se representa con una línea simple.

- Agregación: Se representa con una línea con un rombo blanco. El rombo blanco se coloca del lado de la clase TODO (la clase que almacenará objetos de la otra clase) y la línea sale de la clase PARTE (la clase que será almacenada en la otra clase).

- Composición: Se representa con una línea con un rombo negro. El rombo negro se coloca del lado de la clase TODO (la clase que almacenará objetos de la otra clase) y la línea sale de la clase PARTE (la clase que será almacenada en la otra clase).

1. **¿Cuándo es recomendable usar cada tipo de relación?**

- Asociación: Cuando dos clases están relacionadas entre sí pero existen de forma independiente, no hay una dependencia fuerte entre ellas.

- Agregación: Cuando un objeto forma parte o le pertenece a otro objeto, pero pueden existir de forma independiente.

- Composición: Cuando un objeto no puede ser ese objeto sin otros objetos, es decir, que una clase contiene otros objetos y su ciclo de vida está fuertemente ligado al ciclo de vida de la clase contenedora.

A Codificar! TP2: <https://github.com/RodolfoVelasco1/Programacion2-Com3/tree/master/src/TP2>