**Trabajo Practico Numero 2**

**Programacion 2**

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

La herencia en java es un concepto que permite a una clase (hijo) heredar atributos y métodos de otra clase (padre).

Utilizar herencia en java tiene beneficios como: reutilización de código, facilitar la organización del código, polimorfismo, extensibilidad, abstracción.

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

Esto se indica mediante una línea con una flecha solida que apunta desde la subclase hacia la superclase.

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

Su uso es recomendable cuando se quiere establecer una relación entre dos clases, permitiendo que la subclase herede atributos y métodos. Es útil para reutilizar código y organizar nuestro código y establecer una jerarquía.

* 1. **¿Qué es una interfaz en java?**

Es una colección de métodos y constantes que define un conjunto de comportamientos que una clase puede implementar.

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

En UML, una interfaz es representada usando la misma caja que la de una clase, solo que en esta se pone el nombre de la interfaz, precedido por una línea punteada, además en la interfaz, se muestran los métodos sin detalles de implementación (solo nombre y parámetros).

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

Estas son útiles para generar abstracción, soportar la herencia múltiple, facilitar el testing y la modularidad del código. Permiten crear código mas genérico, reutilizable y mantenible.

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

Las interfaces se utilizan para definir contratos entre clases y permitir la implementación de múltiples interfaces, mientras que las clases abstractas se utilizan para proporcionar una implementación parcial y compartir funcionalidades comunes entre clases relacionadas.

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

Existe la asociación, la agregación, la composición, la herencia y dependencia.

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

Asociación: se representa con una línea solida que conecta las dos partes relacionadas. Puede agregarse una flecha para indicar la navegabilidad.

Agregación: se representa de manera similar a la asociación, pero con una línea que tiene un rombo en el extremo de la clase origen.

Composición: se representa de manera similar a la agregación, pero con un rombo relleno en el extremo de la clase de origen.

Herencia: se representa con una línea solida con una flecha solida que apunta desde la subclase hacia la superclase.

Dependencia: se representa con una línea punteada que va desde la clase que depende hacia la clase de la que depende.

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

Asociación: Se recomienda cuando dos clases están relacionadas de manera general, pero no hay una dependencia fuerte entre ellas. Puede ser bidireccional o unidireccional según la naturaleza de la relación.

Agregación: Se recomienda cuando una clase "todo" contiene una o más instancias de otra clase "parte", pero las partes pueden existir de manera independiente. Es adecuada cuando las partes pueden ser compartidas entre múltiples objetos "todo".

Composición: Se recomienda cuando una clase "todo" contiene una o más instancias de otra clase "parte", y las partes tienen una relación de dependencia fuerte con el "todo".

Es adecuada cuando las partes son exclusivas del objeto "todo" y no tienen sentido sin él.

Herencia: Se recomienda cuando una clase (subclase) comparte características y comportamientos comunes con otra clase (superclase). Es adecuada para modelar relaciones "es un", donde la subclase es un tipo más específico de la superclase.

Dependencia: Se recomienda cuando una clase utiliza o depende de otra clase para realizar una tarea específica, pero no hay una relación estructural entre ellas.

Puede ser temporal y puede cambiar con el tiempo.