TALLER 3

Integrantes:

-Héctor Augusto Daza Roa: -Hadazar@unal.edu.co
-Github: Hadazar.
-Julian Andres Ossa Castro: -Jaossacas@unal.edu.co
-Github:Jaossacas.
Repositorio: https://github.com/Hadazar/Programacion-Orientada-a-Objetos-2015-2

Desarrollo Taller:

1. a. El rol de herencia en los programas de java tienen como función pasar entre métodos y objetos el uso y la funcionalidad de sus características a otros métodos o objetos internos cuando estos lo soliciten y permite crear una cadena de interfaces y funciones.

b. La herencia promueve la reutilización del software por medio de sus traspaso de funcionalidades entre sus métodos y descendientes para propagar la funcionalidad como una cadena por medio de una línea completa de códigos las cuales están sujetas a este.

c. Herencia en POO es cuando un objeto o clase se basa en otro objeto o tipo, utilizando la misma aplicación que especifica la aplicación para mantener el mismo comportamiento. Se trata de un mecanismo para la reutilización del código y para permitir extensiones independientes del software original a través de las clases públicas y las interfaces.

d. La composición y la herencia colocan subobjetos dentro de la clase. Ambos usan la lista de inicialización del constructor para construir esos subobjetos. Pero se preguntará cuál es la diferencia entre los dos, y cuando escoger una y no la otra.

La composición generalmente se usa cuando se quieren las características de una clase existente dentro su clase, pero no en su interfaz. Esto es, aloja un objeto para implementar características en su clase, pero el usuario de su clase ve el interfaz que se ha definido, en vez del interfaz de la clase original. Para hacer esto, se sigue el típico patrón de alojar objetos privados de clases existentes en su nueva clase.

En ocasiones, sin embargo, tiene sentido permitir que el usuario de la clase acceda a la composición de su clase, esto es, hacer públicos los miembros objeto. Los miembros objeto usan su control de accesos, entonces es seguro y cuando el usuario conoce que esta formando un conjunto de piezas, hace que la interfaz sea más fácil de entender. Un buen ejemplo es la clase Car.

e. Ventajas

- Ayuda a los programadores a ahorrar código y tiempo, ya que la clase padre ha sido implementada y verificada con anterioridad, restando solo referenciar desde la clase derivada a la clase base (que suele ser extends, inherits, subclass u otras palabras clave similares, dependiendo del lenguaje).

- Los objetos pueden ser construidos a partir de otros similares. Para ello es necesario que exista una clase base (que incluso puede formar parte de una jerarquía de clases más amplia).
- La clase derivada hereda el comportamiento y los atributos de la clase base, y es común que se le añada su propio comportamiento o que modifique lo heredado.
- Toda clase pueden servir como clase base para crear otras.

Desventajas

Si la jerarquía de clases es demasiado compleja, el programador puede tener problemas para comprender el funcionamiento de un programa. Además puede volverse más complejo detectar y resolver errores de programación, por ejemplo al modificar una clase padre que afecta el funcionamiento de las subclases.

Otro problema es que las subclases se deben definir en código, por lo que los usuarios del programa no puede definir subclases nuevas. Otros patrones de diseño permiten que los usuarios puedan definir variantes de una entidad en tiempo de ejecución.