ANDRÉS DAVID PÉREZ GÓMEZ
JUAN DAVID NOGUERA NARVÁEZ
JOAN SEBASTIAN CORDOBA SALAZAR

EJERCICIO HERENCIA PT1 PT2 Y PT3 (Polimorfismo).

En este ejercicio le dimos desarrollo al primer, segundo y tercer punto de la guía planteada.

Se creó una superclase Vehiculo con atributos comunes (marca, modelo) y un método encender(). Luego, se definieron tres subclases: Auto, Motocicleta y Camioneta, cada una con atributos adicionales y su propia versión del método encender() (sobreescritura).

Además, se implementó un método iniciarVehiculo() que demuestra polimorfismo, permitiendo encender cualquier tipo de vehículo usando la misma interfaz.

RESULTADOS:

Se imprimió información específica de cada vehículo.
Se probó la capacidad de carga de la camioneta
Se mostró cómo el método encender () se comporta distinto según el tipo de vehículo.

EJERCICIO DE JERARQUIA:

Jerarquía de animales

En este ejercicio nuestro grupo implementó una jerarquía de clases para representar animales usando herencia:

Aquí la clase base Animal define atributos comunes y métodos generales como hacerRuido(), comer(), dormir() y rugir().

Se crearon dos subclases felino y canino, que sobrescriben el método rugir() y sirven como una base para clases más específicas.

Las clases Leon, Tigre, Gato, Lobo y Perro heredan de Felino o Canino, sobrescriben métodos como comer() y hacerRuido(), y agregan otros como vacunar() o pasear().

RESULTADOS:

Se imprimen comportamientos específicos de cada animal.
Se demuestra herencia y sobreescritura de métodos.
Se reutiliza código de la clase base y se extiende con nuevas funcionalidades.

EJERCICIO 5 JERARQUIA COMPLETA:

Sistema de empleados

En este ejercicio se diseñó un sistema de gestión de empleados utilizando herencia, clases abstractas y polimorfismo:

La clase abstracta Empleado define atributos comunes (nombre, id, salarioBase) y un método abstracto calcularSalario(), que es implementado por cada subclase.

Se crearon tres subclases: Gerente, Desarrollador y Administrativo, cada una con lógica propia para calcular el salario según los bonos, proyectos o horas extra.

Aquí el método mostrarInformacion() utiliza polimorfismo para mostrar los datos de cada empleado, llamando a su versión específica de calcularSalario().

RESULTADOS:

Se genera un informe con los datos de cada empleado.
Se demuestra herencia, sobrescritura de métodos y uso de clases abstractas.
Se reutiliza código base y se especializa el comportamiento según el rol del empleado.

PUNTO 6. PREGUNTAS

En este punto damos respuesta a las siguientes preguntas:

6 Preguntas
¿Cuál es la diferencia entre una superclase y una subclase en Java?
¿Cómo se implementa la herencia en Java utilizando la palabra clave *extends*?
¿Qué es la sobreescritura de métodos y cómo se utiliza en herencia?
¿En qué casos es recomendable utilizar herencia sobre composición?
¿Cómo se manifiesta el polimorfismo en un programa con herencia en Java?
¿Cuáles son los problemas comunes al usar herencia y cómo pueden evitarse?

Solución

1. La superclase viene siendo la clase general que contiene lo básico, y la subclase lo que hace es heredar eso y agregar cosas más específicas.
2. Se implementa escribiendo: class Subclase extends Superclase. Eso hace que la subclase herede lo que tenga la otra
3. Es cuando una subclase cambia el comportamiento de un método que ya esta en la superclase para que haga algo diferente o distinto
4. Cuando hay una relación de (ES UN), como por ejemplo un perro ES UN animal, o sino pues es mejor usar composición.
5. Se manifiesta cuando una misma acción por ejemplo hacer ruido se comporta diferente dependiendo del objeto, por ejemplo, un perro ladra, y un gato maúlla.
6. Si usamos mucho la herencia el código se va a volver complicado y difícil de cambiar, eso lo podemos evitar aplicando la herencia solo cuando sea necesario y en otros casos usar composición.

¡FIN!