Introducción a la Herencia

En la Programación Orientada a Objetos, la **Herencia** es un mecanismo que permite crear **nuevas clases a partir de otras** ya existentes, que se suponen comprobadas y correctas. Se evita así el problema de tener que andar rediseñando, reescribiendo y modificando una clase que ya existente (y funciona) o duplicando código innecesariamente.

Dado que conceptualmente es **difícil** de entender la **Herencia** y todo lo que conlleva, vamos a realizar un ejercicio práctico sencillo para comprender las bases y después, empezar la parte teórica.

Ejercicio sin herencia – Tienda de Mascotas

Enunciado: Crea un programa para una tienda de mascotas. Los tipos de mascotas que vende esta tienda son: gatos, perros y tortugas. De cada una de ellas hay que saber lo siguiente:

- **Gatos**: la id del collar, el nombre, la raza, el color

- **Perros**: la id del collar, el nombre, la raza, si está vacunado o no

- **Tortugas**: la id del collar, la especie, si es de agua dulce o no

El programa tiene que permitir:

- Mostrar todas las mascotas de la Base de Datos.

- Mostrar todas las mascotas del tipo que nos pidan.

Puedes utilizar la estructura de clases que te damos como base para el ejercicio. Crea los POJOs, indica bien los tipos de datos de las variables, genera las tablas en base de datos y cárgalas con datos de prueba.

Ejercicio con herencia – Tienda de Mascotas

Consulta ahora los apuntes relativos a la **Herencia en Java**. Por ahora, sólo necesitas entender el apartado **Herencia sencilla – Extends**. Vamos a modificar el ejercicio anterior. Si te fijas, en los POJOs de Gato, Perro y Tortuga tienen un montón de atributos comunes. Organiza una estructura de clases en las que los POJO heredan los unos de los otros.

Para hacerlo, sigue los siguientes pasos:

- Perro, Gato y Tortuga tienen un atributo en común: la **id.** Por tanto, lo lógico es que exista una clase con esa id y que la hereden todos los demás. A esa clase decido llamarla **Animal**.
- Gato y Perro tienen los atributos **nombre** y **raza** en común. Decidimos crear una clase que contenga estos dos atributos y la llamamos **Mamífero**.
- No hay más atributos en común, así que los atributos que 'sobran' se los ponemos a Perro, Gato y Tortuga según convenga.
- Generamos las clases en su paquete correspondiente e indicamos el **extends**:
    - Perro y Gato extienden de Mamífero.
    - Mamífero extiende de Animal
    - Tortuga extiende de Animal

Si has completado el proceso correctamente, tu programa deberá de seguir funcionando sin problemas, y sin realizar ningún otro cambio.

¿Por qué Gato, que sólo tiene definido el atributo color en su código, también tiene id, nombre y raza? Porque **han sido heredados** de sus clases padre.

Ejercicio con Abstractas e Interfaces – Tienda de Mascotas

Consulta los apuntes sobre **Herencia compleja – Abstracción**. A continuación, realiza los siguientes cambios:

- En nuestra Tienda no tiene sentido poder instanciar las clases **Animal** y **Mamífero**; principalmente porque no existen en base de Datos. Hazlas **abstractas**.
- Crea una Interfaz llamada **GestorInterfaz**. Esta interfaz debe de implementar los siguientes métodos:
    - ObtenerTodos () – Retorna todos animales de un tipo.
    - ObtenerPorId () – Retorno todos los animales de un tipo por id.

NOTA. Necesitarás revisar qué son las Clases Genérica (Tipo T).

Ejercicio completo – Tienda de Mascotas

Completa el ejercicio para que podamos realizar las siguientes operaciones:

- Añade un **Cocodrilo**. El cocodrilo tiene la id del collar, la especie, el número de dientes y si es de agua dulce o no. (Nueva clase abstracta: **Reptil**).
- Añade las operaciones obtenerTodos () y obtenerPorId () para **Cocodrilo**.
- Nuevas operaciones:
    - Añadir un Perro, Gato, Tortuga o Cocodrilo.
    - Borrar un Perro, Gato, Tortuga o Cocodrilo.
    - Modificar todos los datos de un Perro, Gato, Tortuga o Cocodrilo excepto la id.