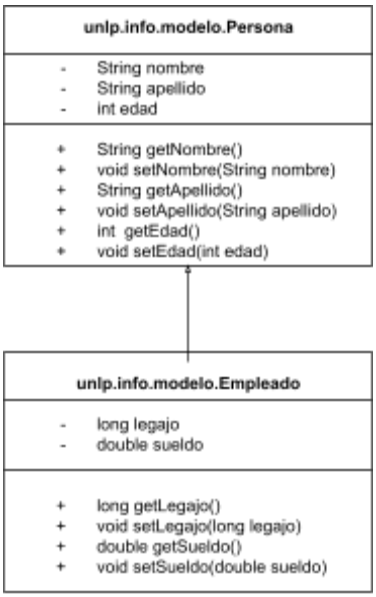
Taller de Lenguajes II
Práctica nº 6

Temas : Interfaces: usos, comparación de objetos, iteradores

1. Considere el siguiente diagrama UML:



   a. Dado un arreglo de Personas, se quiere obtener un arreglo ordenado en función de su apellido y nombre. Para esto usaremos la clase Arrays y su método sort https://docs.oracle.com/javase/8/docs/api/java/util/Arrays.html#sort-java.lang.Object:A-

      Su tarea es escribir el código necesario para que el arreglo de Personas pueda ordenarse. Pruebe que su solución funciona.

   b. Si se quiere hacer lo mismo que en el punto a. pero con un arreglo de Empleados y ordenado por apellido, nombre y sueldo. ¿Cómo lo implementaría?
   c. Y si además quisiera contar con otra opción de ordenación, que sería en base al sueldo, ¿cómo lo implementaría?

2. **Escriba** en eclipse el siguiente código, respetando los paquetes indicados para cada clase o interfaz.

```
package unlp.info.animalespeligrosos;
interface AnimalPeligroso {
	void muerde();
}

package unlp.info.domestico;
interface PerroDomestico {
	void ladra();
	void mueveCola();
```

```
}

package unlp.info.animal;
public class PerroPeligroso implements AnimalPeligroso, PerroDomestico {
	public void muerde() {
		System.out.println("GRRRR!");
	}

	public void ladra() {
		System.out.println("GUAU GUAU!");
	}

	public void mueveCola() {
		System.out.println("Mueve cola");
	}
}
```

- **Verifique que las 2 interfaces y la clase compilan correctamente**

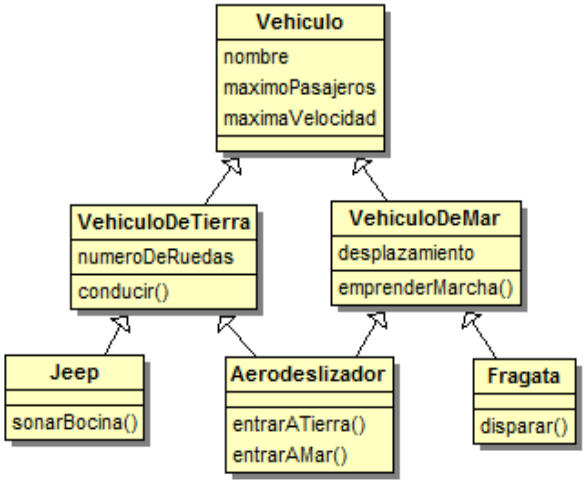
Podemos decir que todos los PerroPeligroso son AnimalPeligroso, y también que todo los PerroPeligroso son PerroDomestico. Gracias al polimorfismo, podemos declarar -por ejemplo- a "gordo" y "rita" de tipo PerroDomestico o AnimalPeligroso con una instanciación diferente. Escriba el siguiente código en una clase ubicada en el paquete unlp.info.test

```
PerroDomestico gordo = new PerroPeligroso();
gordo.ladra();
gordo.mueveCola();
gordo.muerde();

AnimalPeligroso rita = new PerroPeligroso();
rita.muerde();
rita.ladra();
rita.mueveCola();
```

a. Indique cual es el problema que ocurre en compilación relacionados a los mensajes que entienden "rita" y "gordo".
b. Indique una solución de modo que compile y se ejecute sin errores.

3. **Considere** el siguiente diagrama UML referido a Vehiculos. El problema que presenta al intentar implementarlo en Java es la imposibilidad de estructurarlo con *Herencia Múltiple*. Su objetivo es proponer una solución alternativa que si sea viable de implementar en Java.



4. **Catalogador de Sitios Web.**

Considere el siguiente problema: se quiere disponer de un **Catalogador** de sitios web que permite recuperar **Catálogos**. Un **Catálogo** no es más que una colección de información, en este caso, de **sitios web**.
Cada **sitio web** cuenta con **una dirección web, un contador de cantidad de visitas, fecha de creación y un conjunto de "etiquetas"** que permiten catalogar el sitio web, por ej: "Cultural", "Deportivo", "Social", "Tecnología".

El **Catalogador** debe permitir la siguiente funcionalidad:

a. Obtener un Catálogo de sitios web a partir de la aplicación de algún "**filtro**" sobre los sitios web por algún **criterio de filtrado** (a modo de ejemplo: "los sitios web que incluyan 'Cultural' " o "los sitios web con al menos '3 etiquetas'" o "los sitios web con visitas mayor a 1000"), además, el resultado de aplicar el filtro será lueg **ordenado** también por algún criterio: "por cantidad de visitas" o "por mayor cantidad de 'etiquetas' ", etc. En este punto, debe tener en cuenta que el sistema debe ser lo suficientemente flexible de modo que posteriormente se puedan realizar **otros tipos de filtrados y ordenaciones**.

   En conclusión, el **Catalogador** debería tener disponible el siguiente método:
   Catalogo obtenerCatalogo(cjtoSitiosWeb, filtro, orden);

Una vez obtenido el Catálogo se debe poder iterar sobre su contenido. Para lograr esto, se pretende que el Catálogo **pueda ser usado en una cláusula del tipo foreach**.

Su tarea consiste en escribir las clases mencionadas en el ejercicio y el código Java correspondiente. Luego probar su funcionamiento.