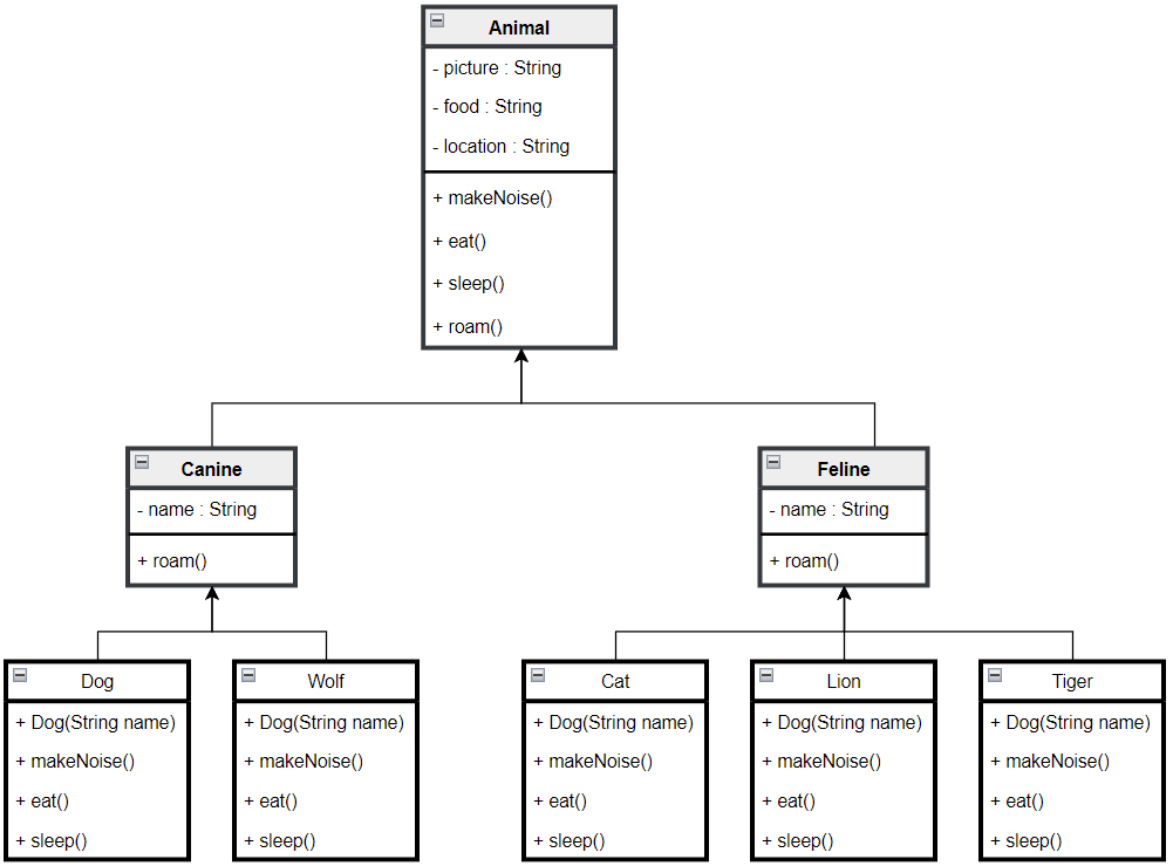
Leonardo Alfredo Velazquez Luna

INTERFACES, CLASES ABSTRACTAS Y HERENCIA

En este ejemplo se crea una clase abstracta **Animal** que es la plantilla para crear todos los atributos o características de un animal.

Se hereda a otras dos clases abstractas (**Canine**, **Feline**) a la cual se refiere a su especie y en esta se define su roam() (andar) dependiendo de la especie.

Finalmente se crean clases concretas, para **Canine** (Dog, Wolf) y **Feline** (Cat, Lion, Tiger). En la siguiente imagen se muestra el diagrama UML:



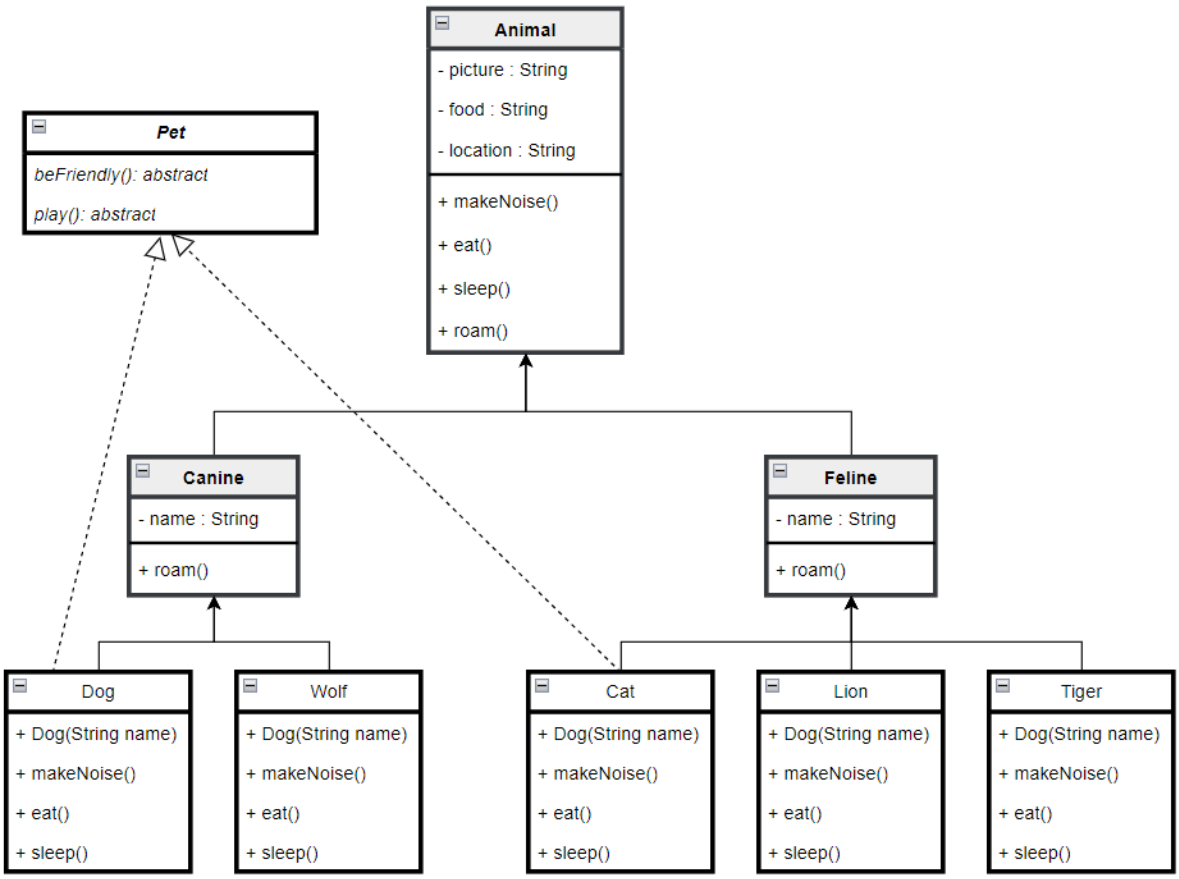
En cada una de las clases concretas se personalizan los métodos de cada uno de ellos, al igual que en su constructor se inicializa el objeto con un nombre.

Todos estos objetos son probados en la clase Zoo.java, se realiza un arreglo de *Animal* donde se crean uno de cada tipo para que en un método estático se imprima el tipo y sus métodos, en el cual varía dependiendo del animal.

Implementando interface

En este caso, instancias de las clases Dog y Cat pueden ser mascotas, por lo que lo mas plausible es implementarle una interfaz *Pet*, donde se creen métodos abstractos y estos se definan y personalicen donde en estas dos clases.

A continuación, se presenta el diagrama UML actualizado:



En las clases Cat y Dog, se implementan y personalizan los métodos *beFriendly()* y *play()*.

En la clase Zoo.java ahora se imprimen todos los animales con los métodos de cada uno, y además se le incluyen los de mascota mediante un if, donde verifica si son instancias de *Pet.*