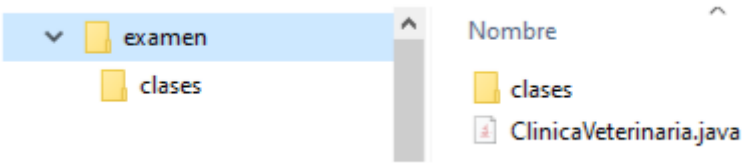
EXAMEN UNIDADES 9 y 16

PROGRAMACIÓN ORIENTADOS A OBJETOS

Se quiere crear una aplicación para gestionar las mascotas de la «**Clínica Veterinaria Anivet**». Sin embargo, el equipo de desarrollo ha implementado la aplicación principal, pero faltan las clases sobre las que se apoya. Tu tarea en este ejercicio será la de completar la aplicación desarrollando todo lo necesario para que funcione correctamente.

TAREA 0 (preparación del examen)

Crea en un ordenador una carpeta llamada **`examen`** y dentro de ella descomprime el material que se te proporciona a través de Moodle. La estructura de carpetas y del código proporcionado debería ser como se muestra en la siguiente imagen:



Seguidamente, abre la carpeta **`examen`** con Visual Studio Code. A partir de ahora trabajarás exclusivamente en la carpeta **clases** y **no eberás tocar**, bajo ningún concepto, el material que se te ha proporcionado.

TAREA 1

a. Crea una interfaz llamada **`MascotaInterface`** y define en su interior un método llamado **`pasear`**, al que le pasamos un parámetro de tipo cadena de texto.

b. Crea una clase que no pueda ser instanciadallamada **`Mascota`**.

c. En la clase **`Mascota`**, define los atributos **`chip`** (cadena de texto), **`nombre`** (cadena de texto), **`edad`** (número entero) y **`vacunada`** (booleano). Este último atributo deberá inicializarse en el constructor como **`false`** como valor por defecto.
d. La clase **`Mascota`** debéra además implementar la interfaz **`MascotaInterface`**.
e. Implementa un método llamado **`vacunar`** que se encargará únicamente de cambiar el estado de la vacunación de la mascota a **`true`**.

TAREA 2

f. Crea una clase llamada **`Perro`** que herede de **`Mascota`** e implemente dos constructores. El primero de ellos recibirá como parámetro únicamente el **`chip`** del animal; el segundo constructor recibirá, además del **`chip`**, el **`nombre`** del perro como parámetro.
g. Define en la clase `Perro` un atributo llamado `raza` (cadena de texto).
h. Define los siguientes comportamientos del perro:
    i. **`ladrar`**: cuando el perro ladra, se muestra por pantalla el mensaje **"¡Guau! ¡Guau!"**.
    ii. **`grunir`**: cuando el perro gruñe, se muestra por pantalla el mensaje **"Grrrrr…."**.
    iii. **`pasear`**: si el perro no ha sido vacunado, deberá mostrarse en pantalla el mensaje **"NombreDelPerro no puede pasear hasta no estar vacunado/a"**, donde **`NombreDelPerro`** es el nombre del perro. En caso de estar vacunado, el perro ladrará cuando sale a pasear.
    iv. **`vacunar`**: si el perro ya ha sido vacunado, se mostrará por pantalla el mensaje **"NombreDelPerro se encuentra vacunado/a"**, donde **`NombreDelPerro`** es el

nombre del perro. Si, por el contrario, no ha sido vacunado, se deberá cambiar el estado de vacunación del animal y, además, éste emitirá un gruñido.

TAREA 3

a. Crea una clase llamada **`Gato`** que herede de **`Mascota`** e implemente dos constructores. El primero de ellos recibirá como parámetro únicamente el **`chip`** del animal; el segundo constructor recibirá, además del **`chip`**, el **`nombre`** del gato como parámetro.
b. Define en la clase **`Gato`** un atributo llamado **`pelaje`** (cadena de texto).
c. Define los siguientes comportamientos del perro:
    i. **`maullar`**: cuando el gato maulla y ronronea, se muestra por pantalla el mensaje **"¡Miaaaaauuuuuu! Brrrrr…."**.
    ii. **`bufar`**: cuando el gato bufa, se muestra por pantalla el mensaje **"Fuuuuuu…."**.
    iii. **`pasear`**: si el gato no ha sido vacunado, deberá mostrarse en pantalla el mensaje **"NombreDelGato no puede pasear hasta no estar vacunado/a"**, donde **`NombreDelGato`** es el nombre del gato. En caso de estar vacunado, el gato maullará y ronroneará cuando sale a pasear.
    iv. **`vacunar`**: si el gato ya ha sido vacunado, se mostrará por pantalla el mensaje **"NombreDelGato se encuentra vacunado/a"**, donde **`NombreDelGato`** es el nombre del gato. Si, por el contrario, no ha sido vacunado, se deberá cambiar el estado de vacunación del animal y, además, éste emitirá un bufido.

TAREA 4

a. Las clases **`Perro`** y **`Gato`** deberán definir los atributos **`totalPerros`** y **`totalGatos`** respectivamente, para que la clínica pueda llevar el cómputo de cuántos animales de cada tipo se han dado de alta.
b. De igual manera, la clase **`Mascota`** deberá definir el atributo **`totalMascotas`** para que la clínica pueda llevar la cuenta de cuántas mascotas se han dado de alta.

TAREA 5

a. Crea la clase **`Clinica`** que deberá implementar la funcionalidad base de la aplicaci´n. Define, por tanto, un atributo llamado **`listado`** que permita guardar hasta un total de **100 mascotas**.
b. Define los siguientes comportamientos para la clase:
    i. **`esVacia`**: comprueba si el listado de mascotas está o no vacío devolviendo **`true`** o **`false`** según corresponda.
    ii. **`esCompleta`**: de forma contraria al punto anterior, comprobará si el listado de mascotas está completo o no. En el primer caso devolverá **`true`** y en el segundo caso, **`false`**.
    iii. **`totalPerros`**: devuelve el número total de perros dados de alta en la aplicación.
    iv. **`totalGatos`**: devuelve el número total de gatos dados de alta en la aplicación.
    v. **`totalMascotas`**: devuelve el número total de perros y gatos que se han dado de alta en la aplicación.
    vi. **`buscarChip`**: busca una mascota por el valor del atributo **`chip`** y la devuelve si se encuentra dentro del listado; en caso contrario, devuelve **`null`**.

vii. **`agregar`**: agrega una mascota, bien sea perro o gato, al listado de la clínica.

viii. **`listar`**: muestra un listado de mascotas con el siguiente formato:

```
[chip]: nombre, edad años
```

Por ejemplo:

```
[1124]: Hachiko, 11 años
[3929]: Mango, 3 años
```

Se deberá implementar **como mínimo** los atributos y métodos indicados. No obstante, si necesitas añadir comportamientos adicionales podrás hacerlo, **siempre que se utilicen**, penalizando la inclusión de los que no se usen. Recuerda comentar convenientemente el código y las cabeceras de los métodos para un mejor entendimiento y corrección.

Añade, al principio de cada clase que hayas creado, el autor con tu nombre y apellidos. Finalizado el examen, revisa tu código, comprime la carpeta **`examen`** en **formato ZIP** y fíjale el nombre **`examen_XXXX.zip`**, donde **`XXXX`** son las iniciales de tu nombre y apellidos. Sube a la plataforma Moodle el **archivo ZIP**.