# PAC 2. UF4.

# *Desarrollo de programas organizados en clases.*

Aníbal Santos Gómez

La clínica veterinaria VetIlerna nos ha pedido realizar una aplicación de escritorio con  Java para la gestión de sus clientes. Durante todo el módulo, vamos a realizar poco a poco, el desarrollo del proyecto.

En esta primera PAC, vamos a crear las clases necesarias para después crear instancias de ellas. Vamos a crear dos clases principales:

- Cliente o Atributos: nombre, dni. o Constructor vacío o Constructor con parámetros o Métodos get y set de ambos atributos o Método toString.

  - Ejemplo: María con DNI 000000000A.

- Mascota o Atributos: nombre, código, género. o Constructor vacío o Constructor con parámetros o Métodos get y set de todos los atributos. o Método toString

  - Ejemplo: Luna con el código 001 es hembra. No hay que entregar ningún programa main, no es ejecutable. Pero se recomienda realizarlo para el estudio de los objetos. Crear y cambiar atributos de los tipos de objetos de la aplicación.

En primer lugar para desarrollar este proyecto crearemos dos clases, dentro de un paquete que llamaremos VetIlerna.

Una primera clase la denominaremos “Cliente”, dentro de ella definiremos los atributos nombre y dni como un tipo de datos private string.

Después llevaremos a cabo la realización de un constructor vacío, el cual tendrá ausencia de los parámetros nombre y din asignados dentro de los atributos de la clase.

Posteriormente llevaremos a cabo un constructor con parámetros donde se le pasaran los atributos que serán de tipo string.

Seguidamente llevaremos a cabo la tarea de crear los métodos get y set y finalmente el método tostring.

```
package VetIlerna; //paquete

public class Cliente { //clase

    private String nombre, dni; //atributos o propiedades de la clase

    public Cliente() {} //constructor vacio que no recibe parametros

    public Cliente (String nombre,String dni) // constructor con parametros de tipo string
    {
        this.nombre=nombre; //enlace de atributos de la clase con los parámetros del constructor
        this.dni=dni;
    }

    public String getDni()  // metodo get
    {
        return dni;
    }

    public String getNombre()
    {
        return nombre;
    }

    public void setNombre(String nombre) //metodo set
    {
        this.nombre=nombre;
    }

    public void setDni(String dni)
    {
        this.dni=dni;
    }

    public String toString() //metodo tostring
    {
        return nombre + " con DNI: " + dni;
    }
}
```

Continuaremos con la siguiente clase del ejercicio llamada “Mascota”, donde seguiremos los pasos anteriormente dicho, con la única excepción de tener en cuenta que el atributo código será de tipo int y no string. Pero la creación de la misma en este caso es idéntica a la anterior.

```
package VetIlerna; //paquete

public class Mascota {  //clase

    private String nombre, genero; //atributos o propiedades de la clase
    private int codigo;

    public Mascota() {} //constructor vacio que no recibe parametros

    public Mascota(String nombre, String genero,int codigo)  // constructor con parametros de tipo string e int
    {
        this.nombre=nombre; //enlace de atributos de la clase con los parámetros del constructor
        this.genero=genero;
        this.codigo=codigo;
    }

    public String getNombre()  // metodo get
    {
        return nombre;
    }

    public String getGenero()
    {
        return genero;
    }
```

```
    public int getCodigo ()
    {
        return codigo;
    }

    public void setNombre(String nombre) //metodo set
    {
        this.nombre=nombre;
    }

    public void setGenero(String genero)
    {
        this.genero=genero;
    }

    public void setCodigo(int codigo)
    {
        this.codigo=codigo;
    }

    public String toString() //metodo tostring
    {
        return nombre + " con el código " + codigo + " es " + genero;
    }
}
```

Finalmente aunque no era obligatorio he procedido a desarrollar un programa main para hacer funcionar las dos clases, donde primeramente creo un tipo de dato Cliente llamado cliente1, en el que utilizo el método de constructor vacío y le paso los parámetros a parte y luego un cliente2 donde el constructor recibe los parámetros, y finalmente utilizando System.out.println(), imprimimos por pantalla lo realizado.

En cuanto a la clase Mascota, llevamos a cabo el mismo procedimiento pero directamente pasando los parámetros e imprimiendo por pantalla. Ambas impresiones se han llevado a cabo con el método toString para ejemplificar lo aprendido.

```
package VetIlerna;

public class VetIlernaMain {

    public static void main(String[] args)
    {
        Cliente cliente1 = new Cliente (); // metodo constructor vacio sin parametros

        cliente1.setNombre("Maria"); // parametros a parte
        cliente1.setDni("70871351E");

        System.out.println(cliente1.toString());

        Cliente cliente2 = new Cliente("Jose","70984532"); // metodo constructor con parametros

        System.out.println(cliente2.toString());

        Mascota mascota1= new Mascota();

        mascota1.setNombre("Kenia");
        mascota1.setGenero("hembra");
        mascota1.setCodigo(7083191);

        Mascota mascota2= new Mascota();

        mascota2.setNombre("Congo");
        mascota2.setGenero("macho");
        mascota2.setCodigo(7145678);

        System.out.println(mascota1.toString());
        System.out.println(mascota2.toString());

    }

}
```

```
Problems  Javadoc  Declaration  Console
<terminated> VetIlernaMain [Java Application] \Library/Java/JavaVirtualMachines/jdk-9.jdk/Contents/Home/bin/java (21 oct. 20 7 20:47:56)
Maria con DNI: 70871351E
Jose con DNI: 70984532
Kenia con el código 7083191 es hembra
Congo con el código 7145678 es macho
```