MEMORIA TRABAJO LABORATORIO

G3

Realizado por:

LAURA FERNANDEZ DEL MORAL GARCÍA CONSUEGRA

MIGUEL DE LAS HERAS FUENTES

ÍNDICE

1. INTRODUCCIÓN

Para resolver la practica que consiste en hacer un sistema de gestión de la información que maneja una protectora de animales crearemos varias clases que nos permitirán llevar a cabo todas las consultas que se nos piden.

Estas clases serán Ayuntamiento, Protectora, Clínica veterinaria, Animal, Perro, Gato, Solicitud, la interfaz constantes y la excepción Intervalo.

2. CLASES

2.1. Ayuntamiento

Esta clase se encarga de conceder una subvención de 1000 € mas otra cantidad que depende del numero de animales de la protectora.

De esta clase se conocen los atributos de **teléfono** de contacto que es de la clase int y **subvención** que es la cantidad que concede a la protectora por animal y es de la clase double. Por ello, tiene una relación de dependencia con la clase Protectora.

Esta clase tendrá los siguientes métodos.

- Un método constructor con los atributos String teléfono y double subvención.
- Getters de los atributos teléfono y subvención.
- toString.

```
3  public class Ayuntamiento {
4
5      private String telefono;
6      private double subvencion; //lo que dona el ayuntamiento por animal
7
8      public Ayuntamiento(String telefono, double subvencion) {
9          this.telefono=telefono;
0          this.subvencion=subvencion;
1      }
2
3      public String getTelefono() {
4          return telefono;
5      }
6
7      public double getSubvencion() {
8          return subvencion;
9      }
0
1      public String toString() {
2          return "Ayuntamiento [telefono=" + telefono + ", subvencion=" + subvencion + "]";
3      }
4
5  }
6
```

2.2. Clínica veterinaria

Esta clase se encarga de ofrecer un precio especial a la clase Protectora para poder hacer una campaña de esterilización para gatas no esterilizadas. Por ello, de esta clase depende la protectora, por lo que tienen relación de dependencia.

De esta clase se conocen los atributos de **nombre** y **teléfono** que son de la clase String y **pEsterilizacion** que corresponde al precio que le ofrece la clínica a la protectora por gata esterilizada y es de tipo double.

Los métodos que encontraremos en esta clase son:

- Un método constructor en el que se le pasan los atributos nombre, teléfono y pEsterilizacion.
- Getters de los atributos
- toString

```
public class Clinica_veterinaria {
    private String nombre,telefono;
    private double pEsterilizacion;

    public Clinica_veterinaria(String nombre, String telefono, double pEsterilizacion) {
        this.nombre=nombre;
        this.telefono=telefono;
        this.pEsterilizacion=pEsterilizacion;
    }
    public String getNombre() {
        return nombre;
    }

    public String getTelefono() {
        return telefono;
    }

    public double getP_esterilizacion() {
        return pEsterilizacion;
    }

    public String toString() {
        return "Clinica_veterinaria [nombre=" + nombre + ", telefono=" + telefono + ", p_esterilizacion=" + pEsteri
            + "]";
    }

}
```

2.3. Solicitud

Esta clase sirve para asignarle una solicitud nueva a un animal. Por ello tiene una relación de asociación con Animal ya que a cada animal se le va a asignar un array de solicitudes.

En esta clase encontraremos los atributos **tipo** que corresponde al tipo de solicitud (0 para acogida y 1 para adopción) y es de tipo int, y **teléfono** que corresponde al teléfono de contacto de la persona que quiere apadrinar o acoger al animal y es de tipo String.

En esta clase tendremos los siguientes métodos:

- Un método constructor al que se le pasan los atributos tipo y teléfono.
- Getters de los atributos nombrados anteriormente
- toString en el que se muestra toda la información de la solicitud; en caso de que el tipo sea 0, se imprimirá por pantalla acogida, y en caso de que sea 1 se imprimirá adopción.

```java
public class Solicitud {
    private int tipo;
    private String telefono;

    public Solicitud(int tipo, String telefono) {
        this.tipo = tipo;
        this.telefono = telefono;
    }

    public int getTipo() {
        return tipo;
    }

    public String getTelefono() {
        return telefono;
    }

    public String toString() {
        String tipoa;
        if (tipo==1) {
            tipoa="adopcion";
        }else {
            tipoa="acogida";
        }
        return "solicitud [tipo=" + tipoa+ ", telefono=" + telefono + "]";
    }
```

2.4. Animal

Esta clase será abstracta ya que tiene relación de herencia con las clases perro y gato. También tendrá relación de asociación con la clase Protectora ya que se le va a pasar un array de animales a su vez, como hemos dicho anteriormente, tendrá una relación de asociación con la clase Solicitud ya que recibirá un array de solicitudes. Al ser una clase abstracta, tendrá un método abstracto que será el de calcular gastos.

Los atributos que encontraremos en esta clase son **edad** que corresponde a la edad del animal y **nSolicitudes** que corresponde al número de solicitudes que se han hecho por el animal y son de la clase int, **sexo** que es de la clase char, **nombre** que es la clase String, **sociable** y **apadrinado** de la clase boolean y por último el atributo **solicitudes** que corresponde a un array de solicitudes que tendrá el animal.

Los métodos que tendrá esta clase son los siguientes:

- Un método constructor en el que se le pasarán todos los atributos (edad, sexo, nombre, apadrinado, sociable y Solicitud).
- Getters de los atributos nombrados anteriormente
- Addsolicitud al que se le pasarán como parámetros los atributos int tipo que corresponde al tipo de solicitud y String teléfono que es el teléfono de contacto.
  Este método sirve para añadir solicitudes al array de solicitudes (atributo solicitudes) de cada animal. Para ello primeramente crearemos un objeto de tipo Solicitud con los atributos nombrados al llamar al método y le daremos el nombre de sol.
  A continuación, miraremos si caben mas solicitudes para ese animal. En caso de que no sea así, se lanzará la excepción “ExceptionIntervalo”. Para que se puede lanzar esta excepción deberemos permitir que este método pueda manejarla mediante el comando throws ExceptionIntervalo.
  En el caso de que si haya espacio, se añadirá la solicitud al array de solicitudes.
  Este método es de tipo void.
- mostrarSolicitudes que nos permitirá imprimir por pantalla todas las solicitudes de tipo adopción que se han hecho por ese animal. Para ello primeramente crearemos una variable de tipo String que utilizaremos para ir añadiendo la información de las solicitudes. Para ello utilizaremos un bucle for que recorra el array de solicitudes del animal y en caso de que la solicitud sea de tipo 1, la cadena se actualizará con el toString de la solicitud correspondiente a esa posición. Esta cadena será la que devuelve el método.
- calcGastos que es un método abstracto que desarrollaremos en las clases Perro y Gato.
- toString

```java
public abstract class Animal implements Constantes{

    protected int edad;
    protected char sexo;
    protected String nombre;
    protected boolean sociable,apadrinado;
    protected Solicitud[]solicitudes;
    protected int n_solicitudes;

    public Animal(String nombre, char sexo,int edad, boolean sociable, boolean apadrinado) {
        solicitudes = new Solicitud[solicitudes_maximas];
        this.edad = edad;
        this.sexo = sexo;
        this.nombre = nombre;
        this.sociable = sociable;
        this.apadrinado = apadrinado;
        n_solicitudes=sol_iniciales;
    }

    public int getEdad() {
        return edad;
    }

    public char getSexo() {
        return sexo;
    }
```

```java
    public String getNombre() {
        return nombre;
    }
    public boolean isSociable() {
        return sociable;
    }
    public boolean isApadrinado() {
        return apadrinado;
    }
    public int getN_solicitudes() {
        return n_solicitudes;
    }
    public void addsolicitud(int tipo, String telefono) throws ExceptionIntervalo{
        Solicitud sol= new Solicitud(tipo,telefono);
        if(n_solicitudes>=solicitudes_maximas) {
            throw new ExceptionIntervalo("No hay mas espacio");
        }else{solicitudes[n_solicitudes]= sol;
            n_solicitudes++;
        }
    }
    public String mostrarSolicitudes() {
        String cadena="";
        for (int i = 0; i < n_solicitudes; i++) {
            if (solicitudes[i].getTipo()==1) {
                cadena+= solicitudes[i].toString();
            }

        }

        return cadena;
    }
    public abstract double calcGastos();
    public String toString() {
        return "Animal [edad=" + edad + ", sexo=" + sexo + ", nombre=" + nombre + ", sociable=" + sociable
                + ", apadrinado=" + apadrinado +  "]";
    }
}
```

2.5. Perro

Esta clase es una de las clases hijas de la clase Animal por lo que tiene relación de herencia con la misma. Además, tiene relación de dependencia con la clase protectora.

En esta clase encontraremos los atributos heredados de la clase padre y algunos nuevos como **raza** que corresponde a la raza del perro y es de tipo String, **tamano** que corresponde al tamaño del perro (peso en Kg) y los atributos **ppp**(potencialmente peligroso) y **leishmania** que son de tipo boolean.

En esta clase encontramos métodos como los siguientes.

- Un método constructor en el que se le pasar los atributos heredados y los nombrados anteriormente.
- Getters de los atributos
- calcGastos que es el método heredado por la clase padre y sirve para calcular los gastos que tiene el perro par la protectora.
  Para ello primeramente crearemos una variable de tipo double llamada gastos, que es una variable acumulativa.
  Como los gastos dependen de los atributos de los animales, utilizaremos condicionales "if" e iremos actualizando la variable gastos que será la que devuelve el método.
  En caso de que el atributo apadrinado sea true, los gastos serán 0.
- Calcpienso: Este método nos servirá para calcular la cantidad de pienso que come el animal. Para ello utilizaremos la formula anterior. Primeramente, crearemos una variable de tipo double llamada pienso. Como la cantidad de pienso depende del tamaño de los perros, utilizaremos condicionales if para calcular la cantidad. Este método devuelve la variable pienso.
- toString

```
3
4  public class Perro extends Animal implements Constantes{
5      private String raza;
6      private int tamano;
7      private boolean ppp,leishmania;
8
9
10     public Perro(String nombre,char sexo, int edad, boolean sociable, boolean apadrinado, String raza,int tamano,
11             boolean ppp, boolean leishmania) {
12         super(nombre, sexo, edad, sociable,apadrinado);
13         this.raza=raza;
14         this.tamano=tamano;
15         this.ppp=ppp;
16         this.leishmania=leishmania;
17         }
18
19     public String getRaza() {
20         return raza;
21     }
22
23     public int getTamano() {
24         return tamano;
25     }
26
27     public boolean isPpp() {
28         return ppp;
29     }
30
31     public boolean isLeishmania() {
32         return leishmania;
33     }
```

```
38     public double calcGastos() {
39         double gastos=0;
40         if (apadrinado) {
41             return gastos=0;
42         }else {
43         if (leishmania)
44             gastos+=gleishmania;
45
46         if (ppp&&!sociable)
47             gastos+=pppysociable;
48
49         gastos+=rabia;
50         return gastos;
51         }
52     }
53     public double calcpienso() {
54         double pienso=0;
55
56         if (tamano<=tamanoa) pienso=piensoa;
57         if (tamano>tamanoa&&tamano<tamanob) pienso=piensob;
58         if (tamano>tamanob) pienso= piensoc*(tamano*kg);
59
60         return (pienso*diasemana)/kg;
61
62     }
63
64     public String toString() {
65         return  "Perro[nombre=" + nombre + ", raza=" + raza + ", tamano=" + tamano + ", ppp=" + ppp + ", leishma
66                 + edad + ", sexo=" + sexo +", sociable=" + sociable + ", apadrinado="
67                 + apadrinado  + "]";
68     }
69 }
70
```

2.6. Gato

Esta clase es una de las clases hijas de la clase Animal por lo que tiene relación de herencia con la misma. Además, tiene relación de dependencia con la clase protectora.

En esta clase encontraremos los atributos heredados de la clase padre y algunos nuevos como **esterilizado** que es de tipo boolean.

Los métodos que encontraremos son:

- Un método constructor al que se le pasaran los atributos heredados y el atributo esterilizado.
- Getter del atributo esterilizado.
- calcGastos: Es el método heredado de la clase Animal y sirve para calcular los gastos que tiene el gato par la protectora.
  Para ello primeramente crearemos una variable de tipo double llamada gastos, que es una variable acumulativa.
  Como los gastos dependen de los atributos de los animales, utilizaremos condicionales "if" e iremos actualizando la variable gastos que será la que devuelve el método.
  En caso de que el atributo apadrinado sea true, los gastos serán 0.
- toString

```
3
4  public class Gato extends Animal implements Constantes{
5
6      private boolean esterilizado;
7
8      public Gato(String nombre, char sexo, int edad, boolean sociable, boolean apadrinado, boolean esterilizado) {
9          super(nombre,sexo,edad,sociable, apadrinado);
10         this.esterilizado=esterilizado;
11     }
12
13     public boolean isEsterilizado() {
14         return esterilizado;
15     }
16
17     public double calcGastos() {
18         int gastos=0;
19         if (apadrinado) {
20             return gastos=0;
21         }else {
22         if (!esterilizado&&sexo=='h')
23             gastos+=esterilizacion;
24             return gastos;
25         }
26     }
27
28     public String toString() {
29         return "Gato[nombre=" + nombre +", sexo="+sexo+ ", esterilizado=" + esterilizado + ", edad=" + edad +
30                 ", sociable=" + sociable + ", apadrinado=" + apadrinado +  "]";
31     }
32 }
```

2.7. Protectora

Esta clase tendrá una relación de asociación con la clase Animal ya que uno de los atributos de la clase será un array de animales. También tendrá relación de dependencia con las clases Perro y Gato ya que, para hacer algunos métodos de esta clase, se necesita llamar a métodos de las clases Perro y Gato.

Los atributos de esta clase son **nAnimales** que representa el numero de animales que tiene la protectora y es de tipo int, y el anteriormente mencionado array de animales llamado **animales** que es de la clase Animal.

En esta clase también se implementará la interfaz "Constantes" para realizar algunos cálculos.

Los métodos que encontraremos en esta clase son:

- Un método constructor al que se le pasan los atributos citados anteriormente.
- Getters y Setters de los atributos de la clase (nAnimales y animales)
- calcSubvencion en el que se le pasa la variable ayto de la clase ayuntamiento. Este método sirve para calcular cual va a ser la subvención que le dará el ayuntamiento a la protectora. Retorna un double.
- campEsterilizacion en el que se le pasa la variable clínica de la clase Clinica_veterinaria que sirve para saber cual es el precio de esterilización por gata. Este método sirve para calcular el precio total de la campaña de esterilización.
  Para llevar a cabo este calcula, primero deberemos crear una variable double llama dinero. A continuación, tendremos que recorrer todo el array de animales para lo que utilizaremos un bucle "for". En cada posición deberemos comprobar mediante una sentencia if que el animal es de la clase Gato (utilizaremos instanceof Gato) y si esta esterilizado y es hembra para los que utilizaremos los métodos isesterilizado () y getSexo () respectivamente. En caso de que se cumpla todo lo anterior, la variable dinero se actualizara.
  Este método devuelve el valor de la variable dinero.
- Existencia: Este método sirve para comprobar si el animal que hemos seleccionado en el main pertenece a la protectora. Para ello se le pasa como argumento una variable de tipo String llamada animal la cual indica el animal que hemos elegido en el main.
  Para poder comprobar la existencia, primeramente crearemos una variable boolean llamada existe que se inicializa a false. A continuación, recorreremos todo el array de animales, y en caso de que el nombre de algún animal coincida con la variable animal pasada como parámetro, se actualizara a true.
  Este método devolverá un boolean que será el valor de la variable existe.
- Addsol: Este método permitirá añadir solicitudes a un animal. Para ello se le pasarán como parámetros las variables animal de tipo String la cual se refiere al animal al cual le queremos añadir la solicitud, otra variable de tipo int llamada tipo que indica el tipo de solicitud y otra variable de tipo String llamada teléfono.
  Para añadir la solicitud tendremos que recorrer todo el array de animales. En caso de que en esa posición coincidan el nombre del animal con la variable animal pasada como parámetro, le añadiremos la solicitud mediante el método Addsolicitud de la clase animal. Este método manejará la excepción ExceptionIntervalo.
  El método devolverá una variable de tipo String.
- Addanimales: Este método recibirá como parámetro una variable de tipo Animal llamada animal. El método sirve para añadir animales al array de animales que tiene la protectora. Para ello utilizamos un condicional "if" que comprueba si hay espacio para mas animales en el array. Este método es de tipo void.
- mostrarAnimales: Este método recorrerá el listado de animales e ira concatenando toda la información del array animales. Este método devolverá un String que será la cadena que hemos dicho anteriormente.
- mostrarsol: Este método recibirá como parámetro una variable de tipo string llamada animal. Este recorrerá al igual que los anteriores el número de animales, pero comprobará si el nombre del animal es equivalente al introducido por teclado. Una vez hecha esta comprobación se mostrará el nombre del animal y su número de solicitudes. Este método devuelve una variable String.
- calcGastos: Este método sirve para reunir los gastos de todo el listado de animales y es de tipo double.

- calcPienso: Este método hace un recorrido de todo el listado de animales y va comprobando si el animal es un perro y su edad es mayor a la mínima, en caso de no ser así la cantidad es cero debido a que tiene menos de 18 meses. Este método es de tipo double.
- toString: este método retornará un mensaje con toda la información de la protectora.

```java
public class Protectora implements Constantes{

private int nAnimales;
private Animal[]animales;


public Protectora( ) {

    nAnimales=animalesin;
    animales=new Animal[animalesmax];
}

public int getN_animales() {
    return nAnimales;
}

public void setN_animales(int n_animales) {
    this.nAnimales = n_animales;
}

public double calcSubvencion(Ayuntamiento ayto) {
    return (nAnimales*ayto.getSubvencion())+subvencion;
}

public double campEsterilizacion(Clinica_veterinaria clinica) {
    double dinero=0;
    for (int i = 0; i < nAnimales; i++) {
        if (animales[i] instanceof Gato && animales[i].getSexo()=='h'&&!((Gato)animales[i]).isEsterilizado()) {
            dinero+=clinica.getP_esterilizacion();
        }
    }
    return dinero;
}
```

```java
public boolean existencia(String animal) {

    boolean existe=false;
    for (int i = 0; i < nAnimales; i++) {
        if (animales[i].getNombre().equalsIgnoreCase(animal)) {
            existe=true;
        }
    }
    return existe;
}

public String addSol(String animal, int tipo, String telefono) throws ExceptionIntervalo{
    String cadena="";
    for (int i = 0; i < nAnimales; i++) {
        if (animales[i].getNombre().equalsIgnoreCase(animal)) {
            animales[i].addsolicitud(tipo,telefono);
            cadena="Solicitud añadida";
        }
    }
    return cadena;


}

public void addAnimales(Animal animal) {
    if (nAnimales<animalesmax) {
        animales[nAnimales]=animal;
        nAnimales++;
    }
}

public String mostrarAnimales() {
    String cadena="";
    for (int i = 0; i < nAnimales; i++)
        cadena=cadena+(i+1)+". " + animales[i]+"\n";
    return cadena;
}
```

```
public String mostrarsol(String animal) {
    String cadena= "";
    for (int i = 0; i < nAnimales; i++) {
        if (animales[i].getNombre().equalsIgnoreCase(animal)) {
            cadena= cadena + animales[i].getNombre() +": "+ animales[i].mostrarSolicitudes() + "\n";
        }
    }
    return cadena;
}

public double calcGastos() {
    double gastos=0;
    for (int i = 0; i < nAnimales; i++) {
        gastos+=animales[i].calcGastos();
    }
    return gastos;
}

public double calcPienso() {
    double cantidad=0;
    for (int i = 0; i < nAnimales; i++) {
        if (animales[i] instanceof Perro&&animales[i].getEdad()>edadminima) {
            cantidad+=((Perro)animales[i]).calcpienso();
        }else {
            cantidad+=0; //si la cantidad es 0 es porque tiene menos de 18 meses
        }
    }
    return cantidad;
}

public String toString() {
    return "Protectora [nAnimales=" + nAnimales + ", animales="
            + Arrays.toString(animales) + "]";
}

}
```

2.8. Principal

- Main:

```
public class Principal {
    final static Scanner teclado= new Scanner(System.in);

    public static void main(String[]args) {
        System.out.println("--BIENVENIDO AL MENÚ DE CONSULTAS DE LA PROTECTORA--\n");
        Protectora protectora= new Protectora();
        Ayuntamiento ayto= new Ayuntamiento("926254678",25);
        Clinica_veterinaria clinica= new Clinica_veterinaria("Kivet","926215247",25);
```

Antes de comenzar con el programa se crean los distintos objetos (clases) que son protectora, Clínica veterinaria, ayuntamiento, con sus respectivos datos (constructores).

```
    try {
        leerfichero(protectora);

        int opcion=0;
        do {
            try {
                opcion=menu();
                if (opcion<1||opcion>8) throw new ExceptionIntervalo("Numero fuera de rango[1-8]");
                consultas(protectora,ayto,clinica,opcion);

            }
            catch (ExceptionIntervalo ex) { //excepcion fuera del intervalo

                System.out.println(ex.getMessage());

            }
            catch (InputMismatchException ex) { //excepcion caracter no numerico

                System.out.println("Incorrecto.Seleccione un caracter numerico");
                teclado.next();

            }

        }while(opcion!=8);

    } catch (FileNotFoundException e) {
        System.out.println("Fichero no encontrado");
    }
}
```

El sistema arrancará leyendo el fichero de datos de la protectora de animales, es decir, Animales.txt, y posteriormente lanzará el menú de opciones con las distintas consultas. En caso de que la opción elegida sea menor que 1 o mayor que 8 lanzará una excepción (ExceptionIntervalo), de no ser así, se podrá ejecutar la consulta elegida, todo ello metido dentro de un try. Capturará dos excepciones, una de ellas será la comentada anteriormente que lanzará el mensaje que hemos puesto (“Numero fuera de rango”), la otra excepción saltará en caso de ser

un carácter no numérico. Todo este apartado se repetirá hasta que el usuario introduzca la opción dentro del intervalo y no sea un String. Por último, capturará la principal excepción en caso de que no encuentre el fichero de texto.

- Método menú:

```
public static int menu() {

    System.out.println("Seleccione que desea hacer\n 1. Mostrar toda la informacion de los animales de la protectora\n"
            + " 2. Realizar solicitud de adopcion o acogida\n 3. Consultar el listado de solicitudes de adopcion de un animal\n"
            + " 4. Mostrar gastos veterinarios anuales de la protectora\n"
            + " 5. Calcular coste de campaña de esterilizacion de gatas\n"
            + " 6. calcular cantidad de pienso de un perro adulto para un semana( en Kg)\n"
            + " 7. calcular subvencion que concederia el Ayuntamiento\n 8. Salir");

    int opcion=teclado.nextInt();
    return opcion;
}
```

Este método imprimirá por pantalla el listado de opciones que nos ofrece la protectora de animales introduciendo por teclado un número y devolviendo un String. este método es de tipo int.

- Método consultas

```
public static void consultas(Protectora protectora, Ayuntamiento Ayto, Clinica_veterinaria clinica,int opcion) throws ExceptionIntervalo {
    switch (opcion) {
    case 1:
        consulta1(protectora);
        break;
    case 2:
        consulta2(protectora);
        break;
    case 3:
        consulta3(protectora);
        break;
    case 4:
        consulta4(protectora);
        break;
    case 5:
        consulta5(protectora,clinica);
        break;
    case 6:
        consulta6(protectora);
        break;
    case 7:
        consulta7(protectora,Ayto);
        break;
    case 8:
        System.out.println("Fin del programa");
        break;

    }
}
```

Este método recibe como parámetros una protectora, un ayuntamiento y una clínica veterinaria.

Dentro de este método se utilizará una condición switch comprobando la opción elegida y ejecutando cada una de las consultas en caso de ser uno de los números elegidos

- Consulta 1:

```
public static void consulta1(Protectora protectora) {
    System.out.println(protectora.mostrarAnimales());
}
```

Este método imprimirá por pantalla todo el listado de animales recibiendo como parámetro una variable protectora llamada protectora.

- Consulta 2:

```
public static void consulta2(Protectora protectora) throws ExceptionIntervalo{
    System.out.println("Introduzca el nombre del animal que quiere apadrinar o acoger");
    String nombre=teclado.next();
    boolean existe=protectora.existencia(nombre);
    if (!existe) {
        System.out.println("El animal no existe");
    }else {

    System.out.println("Introduzca el tipo de solicitud(acogida-->0 adopcion-->1");
    boolean correcto=false;
    int tipo=0;
    do {
    try {
        tipo=teclado.nextInt();
        if(tipo!=1&&tipo!=0)throw new ExceptionIntervalo("Error. Seleccione 0 para acogida o 1 para adopcion");
        correcto=true;
        System.out.println("Introduzca un numero de contacto");
        String telefono=teclado.next();
        System.out.println(protectora.addSol(nombre, tipo, telefono));

    }catch (ExceptionIntervalo e) {
        System.out.println(e.getMessage());
    }catch (InputMismatchException e) {
        System.out.println("Introduzca un caracter numerico");
        teclado.next();
    }
    }while(!correcto);
    }
}
```

En esta consulta recibirá como parámetro una variable protectora. Este servirá para poder apadrinar o acoger un animal. Primero pedirá el nombre a buscar en el listado de animales y se ira al método protectora, existencia que comprobará si existe ese nombre en dicho listado, de no ser así saltará un mensaje diciendo que el animal no existe. Si existe, el sistema podrá entonces guardar la solicitud que le diremos. Este método al igual que el try catch del principio, tendrá la misma excepción, es decir, la excepción, se ira al método protectora.addSol que posteriormente añadirá la solicitud introducida al animal elegido. Es de tipo void esta consulta.

- Consulta 3:

```
public static void consulta3(Protectora protectora) {
    System.out.println("Elija un animal para ver sus solicitudes");
    String animal=teclado.next();
    boolean existe=protectora.existencia(animal);
    if (!existe) {
        System.out.println("El animal no existe");
    }else {
    System.out.println(protectora.mostrarsol(animal) +"\n");
    }
}
```

Recibirá como parámetro una variable protectora, y dentro de este método el usuario introducirá el nombre del animal para después dirigirse al método de existencia que comprobará si el animal introducido existe en el listado de animales de la protectora de animales, de ser así el sistema imprimirá las solicitudes de dicho animal

- Consultas 4,5,6 y 7:

```java
//consulta 4

public static void consulta4(Protectora protectora) {
    System.out.println("Los gastos anuales de la protectora son: "+ protectora.calcGastos() + " euros");
}

//consulta 5

public static void consulta5(Protectora protectora, Clinica_veterinaria clinica) {
    System.out.println("El coste de la campaña de esterilizacion es: " + protectora.campEsterilizacion(clinica) + " euros\n");
}

//consulta 6

public static void consulta6(Protectora protectora) {
    double cantidad=protectora.calcPienso();

        System.out.printf("%.2f Kg por semana\n",cantidad);

}
//consulta 7
public static void consulta7(Protectora protectora, Ayuntamiento Ayto) {
    System.out.println("La subvencion que recibira la protectora por parte del ayuntamiento es de: "
            + protectora.calcSubvencion(Ayto) + " euros\n");
}
```

La consulta 4 imprimirá por pantalla la suma de todos los gastos anuales veterinarios de cada uno de los animales dependiendo de si es perro o gato. La consulta 5 imprimirá por pantalla el coste de la campaña de la esterilización de gatas, es decir, irá al método de la protectora y posteriormente mostrará el gasto que calcula la clase protectora. La consulta 6, imprimirá por pantalla la suma de la cantidad del pienso de los perros mayores de 18 meses yendo al método de la clase protectora. Y por último la consulta 7, imprimirá por pantalla el cálculo de la subvención del ayuntamiento, es decir, ira al método de la clase protectora y hará dicho calculo. Todos estos métodos son de tipo void, debido a que son las consultas.

- Método leer fichero

```java
public static void leerfichero(Protectora protectora) throws FileNotFoundException {

        Scanner escaner =new Scanner(new File("Animales.txt"));
        Animal newAnimal = null;
        char tipoAnimal,sexo;
        String nombre,raza;
        int edad,tamano;
        boolean sociable,apadrinado,ppp,leishmania,esterilizado;
        while(escaner.hasNext()) {
            tipoAnimal=(escaner.next()).charAt(0);
            nombre=escaner.next();
            sexo=(escaner.next()).charAt(0);
            edad=escaner.nextInt();
            sociable=escaner.nextBoolean();
            apadrinado=escaner.nextBoolean();
            if (tipoAnimal=='p') {

                raza=escaner.next();
                tamano=escaner.nextInt();
                ppp=escaner.nextBoolean();
                leishmania=escaner.nextBoolean();


                newAnimal= new Perro(nombre,sexo,edad,sociable,apadrinado,raza,tamano,ppp,leishmania);


            }else {
                esterilizado=escaner.nextBoolean();

                newAnimal= new Gato(nombre,sexo,edad,sociable,apadrinado,esterilizado);
            }
            protectora.addAnimales(newAnimal);
        }
        escaner.close();
    }
}
```

Este método irá leyendo cada una de las líneas del fichero Animales.txt e irá comprobando cada una de las líneas fijando así cada una de las variables a cada atributo de los animales. Al final de dicho método, se utilizará el método protectora.addAnimales y este añadirá cada uno de los animales a nuestro array.

2.9. Excepción ExceptionIntervalo

Esta clase sirve para poder lanzar y manejar la excepción que nos permite controlar que no se introduzcan datos fuera del intervalo permitido. Esta formada por un constructor en el que se le pasa por parámetro una variable de tipo String que será el mensaje que se imprimirá en caso de que se lance la excepción.

3. MANUAL DE USUARIO

- Al iniciar el programa nos saldrá el menú de consultas.

```
--BIENVENIDO AL MENÚ DE CONSULTAS DE LA PROTECTORA--

Seleccione que desea hacer
 1. Mostrar toda la informacion de los animales de la protectora
 2. Realizar solicitud de adopcion o acogida
 3. Consultar el listado de solicitudes de adopcion de un animal
 4. Mostrar gastos veterinarios anuales de la protectora
 5. Calcular coste de campaña de esterilizacion de gatas
 6. calcular cantidad de pienso para un semana( en Kg)
 7. calcular subvencion que concederia el Ayuntamiento
 8. Salir
```

- Si introducimos un 1 nos saldrá la lista de animales que tiene la protectora.

```
1
1. Perro[nombre=Sombra, raza=Collie, tamano=16, ppp=false, leishmania=true, edad=5, sexo=h, sociable=true, apadrinado=true]
2. Gato[nombre=Copito, sexo=m, esterilizado=false, edad=3, sociable=false, apadrinado=true]
3. Perro[nombre=Scooby, raza=Mestizo, tamano=13, ppp=false, leishmania=true, edad=3, sexo=m, sociable=true, apadrinado=false]
4. Gato[nombre=Blanquita, sexo=h, esterilizado=false, edad=3, sociable=true, apadrinado=false]
5. Perro[nombre=Ulises, raza=Rottweiler, tamano=51, ppp=true, leishmania=false, edad=7, sexo=m, sociable=false, apadrinado=false
6. Perro[nombre=Samba, raza=Podenco, tamano=6, ppp=false, leishmania=false, edad=1, sexo=h, sociable=true, apadrinado=false]
7. Perro[nombre=Nala, raza=Mestizo, tamano=10, ppp=false, leishmania=true, edad=8, sexo=h, sociable=true, apadrinado=false]
8. Gato[nombre=Estrella, sexo=h, esterilizado=true, edad=4, sociable=true, apadrinado=true]
9. Perro[nombre=Pancho, raza=Galgo, tamano=27, ppp=false, leishmania=true, edad=3, sexo=m, sociable=true, apadrinado=false]
10. Perro[nombre=Coco, raza=Pitbull, tamano=22, ppp=true, leishmania=false, edad=6, sexo=m, sociable=true, apadrinado=true]
11. Gato[nombre=Isidoro, sexo=m, esterilizado=true, edad=6, sociable=false, apadrinado=false]
12. Perro[nombre=Zara, raza=Pitbull, tamano=18, ppp=true, leishmania=false, edad=9, sexo=h, sociable=false, apadrinado=true]
13. Gato[nombre=Minino, sexo=m, esterilizado=false, edad=2, sociable=true, apadrinado=false]
14. Gato[nombre=Luna, sexo=h, esterilizado=false, edad=1, sociable=false, apadrinado=true]
15. Perro[nombre=Curro, raza=Mestizo, tamano=7, ppp=false, leishmania=false, edad=1, sexo=m, sociable=true, apadrinado=true]
```

- Si introducimos un 2 podremos hacer una solicitud para un animal. Introduciendo un nombre valido podremos poner tipo y teléfono de la solicitud. En caso contrario nos saldrá un mensaje de que no existe el animal.

```
2
Introduzca el nombre del animal que quiere apadrinar o acoger
sombra
Introduzca el tipo de solicitud(acogida-->0 adopcion-->1
1
Introduzca un numero de contacto
12345
Solicitud añadida
```

```
2
Introduzca el nombre del animal que quiere apadrinar o acoger
eher
El animal no existe
```

- Si introducimos un 3 podremos ver las solicitudes de tipo adopción que tiene un animal.

```
3
Elija un animal para ver sus solicitudes
sombra
Sombra: El animal tiene 1 solicitudes
solicitud [tipo=adopcion, telefono=12345]
```

- Introduciendo un 4 nos saldrá en la pantalla los gastos anuales de la protectora.

```
4
Los gastos anuales de la protectora son: 1178.0 euros
```

- Introduciendo un 5 nos saldrá en la pantalla el coste de la campaña de esterilización de gatas.

```
5
El coste de la campaña de esterilizacion es: 50.0 euros
```

- Introduciendo un 6 nos saldrá en la pantalla la cantidad de pienso que comen los perros adultos durante la semana

```
6
17,29 Kg por semana
```

- Introduciendo un 7 nos imprimirá la subvención que le da el ayuntamiento a la protectora

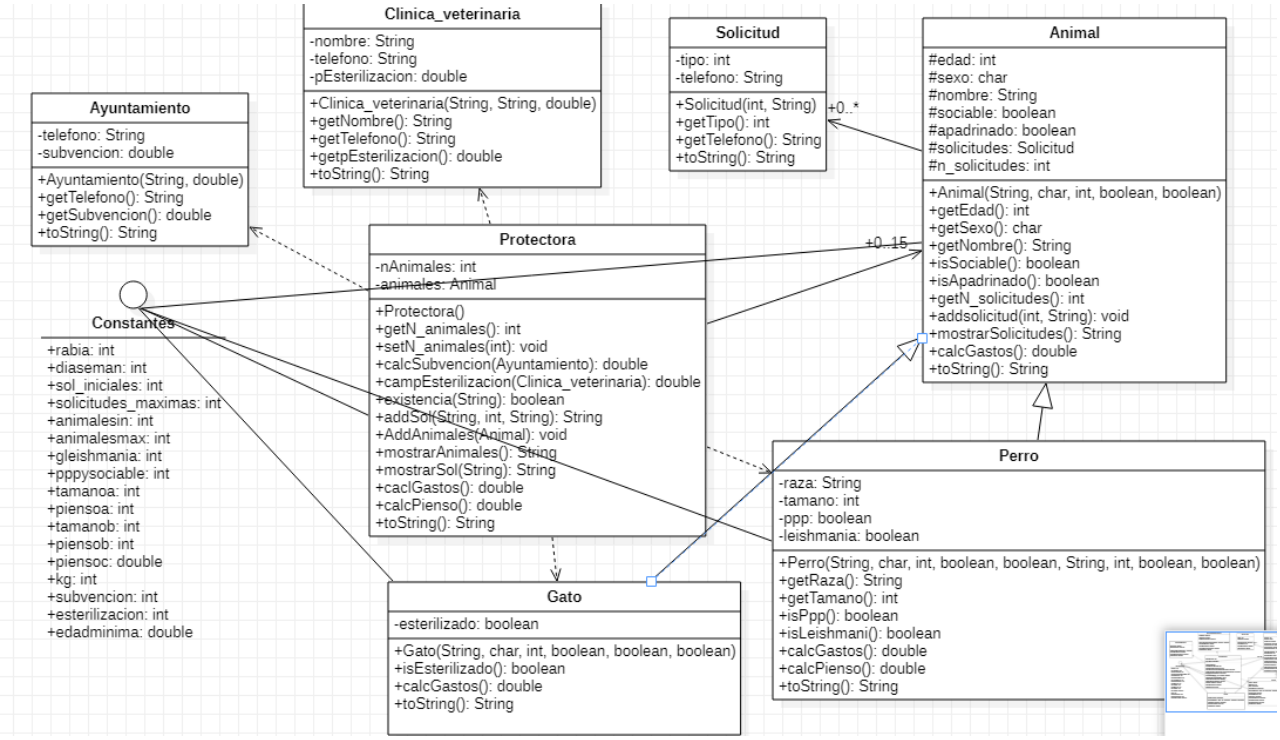
```
7
La subvencion que recibira la protectora por parte del ayuntamiento es de: 1375.0 euros
```

- Introduciendo un 8 finaliza el programa.

```
8
Fin del programa
```

4. DIAGRAMA UML

A continuación, está una captura del diagrama UML del programa. En caso de que no se aprecio bien, esta adjuntado el archivo del programa.



5. PORCENTAJE DE PARTICIPACIÓN EN LA ELABORACIÓN DEL TRABAJO

NOMBRE	PORCENTAJE(%)
Laura Fernandez del Moral García Consuegra	50
Miguel de las Heras Fuentes	50