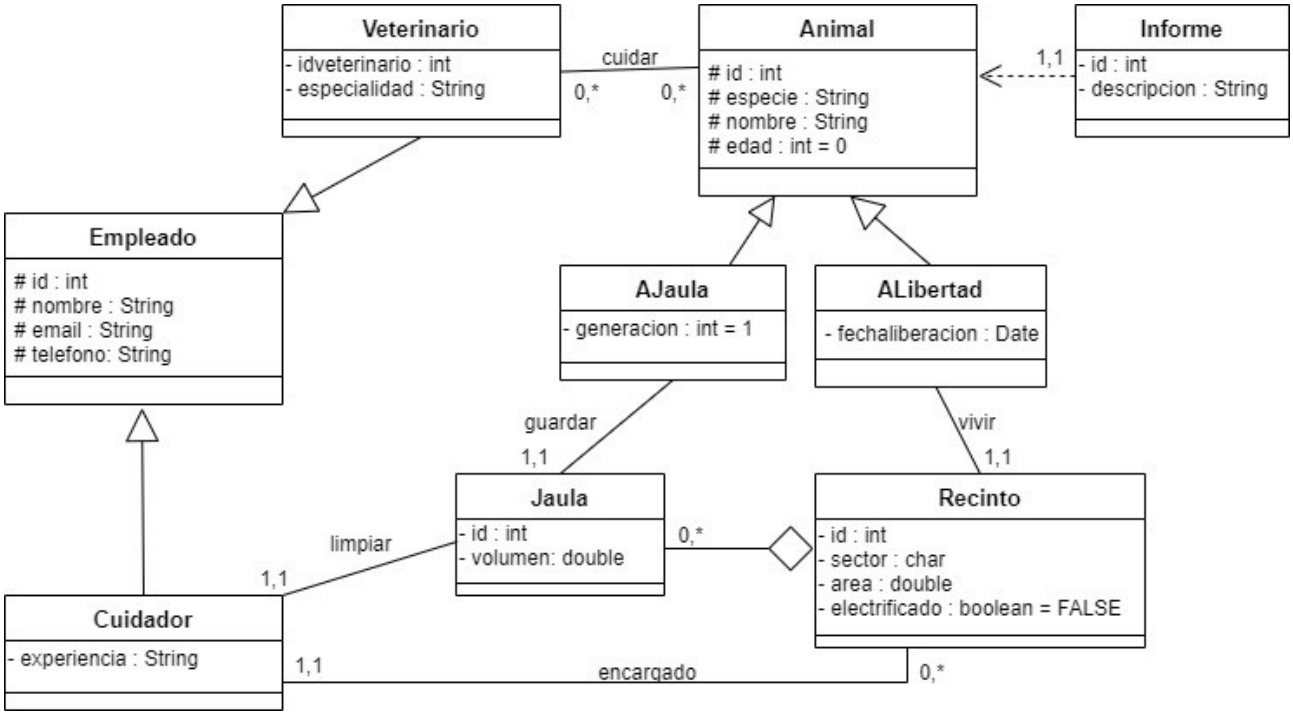
TAÚLFO ARGENTA

PROGRAMACION – 1º DAM-DAW **SOLUCIÓN a la PRUEBA 2 de la 2ª EVALUACIÓN curso 2020/2021**	
Nombre y Apellidos:	**Puntuación:**

Junto a estas especificaciones se incluye un proyecto Java completo de NetBeans que codifica el siguiente diagrama de clases para la **gestión de un Zoológico** (las especificaciones del sistema se encuentran al final de este documento):



En este examen deberán codificarse todas las siguientes funciones, y deberán ser usadas en un programa principal (`Examen2.java`) para verificar su correcto funcionamiento. Queda a elección de cada estudiante el dónde y de qué forma implementarlas, lo cual será valorado en la nota final.

1) (Máx 2ptos.) Realizar la validación de los campos básicos de la clase `Recinto` y crear un nuevo método `nuevoRecinto()` que solicite por la entrada estándar los valores para un nuevo recinto, es decir, `identificador`, `sector`, `area` y `electrificado` (sin el encargado y sin las jaulas) y que incluya esa validación.

```
public class RecintoException extends Exception {

    public RecintoException(String msj) {
        super(msj);
```

```
    }

    public static boolean validarId(int id) {
        return id > 0;
    }

    public static boolean validarSector(char sector) {
        return sector == 'A' || sector == 'E' || sector == 'I';
    }

    public static boolean validarArea(double area) {
        return area > 0.0;
    }
}

public class Recinto {
(…)

public static Recinto nuevoRecinto() {
        Scanner in = new Scanner(System.in);
        Recinto ret = new Recinto();
        boolean valido = false;
        //se lee el valor para el identificador y se valida
        int id = 0;
        do {
            in = new Scanner(System.in);
            id = in.nextInt();
            valido = RecintoException.validarId(id);
        } while (!valido);
        ret.setId(id);
        //se lee el valor del sector y se valida
        valido = false;
        char sector = ' ';
        do {
            in = new Scanner(System.in);
            sector = in.nextLine().charAt(0);
            valido = RecintoException.validarSector(sector);
        } while (!valido);
        ret.setSector(sector);
        //se lee el valor del area y se valida
        valido = false;
        double area = 0.0;
        do {
            in = new Scanner(System.in);
            area = Utilidades.leerDouble();
            valido = RecintoException.validarArea(area);
        } while (!valido);
        ret.setArea(area);
        //se lee el valor de electrificado
        boolean elect = Utilidades.leerBoolean();
        ret.setElectrificado(elect);
```

```
            //se establece a null el encargado del nuevo recinto
            ret.setEncargado(null);
            //se deja vacía la lista de jaulas del nuevo recinto
            ret.setJaulas(new ArrayList<Jaula>());
            return ret;
        }

    }
```

2) (Máx 2ptos.) Crear una enumeración `Vacunas` para las posibles vacunas de los animales del zoo. Los valores de esa enumeración son: "Rabia", "Tiña", "Hepatitis" y "Cólera". Cada vacuna tiene un identificador de tipo entero distinto y una abreviatura con las 3 primeras letras en mayúsculas del nombre de la vacuna ("RAB" para la rabia, "TIÑ" para la tiña, "CÓL" para cólera…). Se pide disponer de métodos para obtener tanto el `id`, como el `nombre` y la `abreviatura` de las vacunas de la enumeración, así como un constructor para nuevas vacunas con 2 argumentos (el `id` y el `nombre` de la vacuna). Este constructor calculará automáticamente la `abreviatura` de la nueva vacuna en función del valor dado para el nombre.

```
public enum Vacunas {
    RABIA(1, "Rabia"),
    TIÑA(2, "Tiña"),
    HEPATITIS(3, "Hepatitis"),
    CÓLERA(4, "Cólera");

    private int id;
    private String nombre;
    private String abreviatura;

    Vacunas(int id, String nombre) {
        this.id = id;
        this.nombre = nombre;
        this.abreviatura = nombre.substring(1, 3).toUpperCase();
    }

    public int getId() {
        return this.id;
    }

    public String getNombre() {
        return this.nombre;
    }

    public String getAbreviatura() {
        return abreviatura;
    }
}
```

3) (Máx 2ptos.) Escribir una función que exporte a un fichero de texto de nombre `limpiezas.txt` la información sobre la limpieza de todas las `Jaulas` del zoo, es decir, por cada jaula se escribirá en una línea distinta la cadena:

Jaula <`idJaula`> limpiada por <`nombre_limpiador`>.

```
private static void exportarDatosLipiezaJaulas(ArrayList<Jaula> jaulas) {
        String path = "limpiezas.txt";
        File fichero = new File(path);
        FileWriter escritor = null;
        PrintWriter buffer = null;
        try {
            try {
                escritor = new FileWriter(fichero, true);
                buffer = new PrintWriter(escritor);
                for (Jaula j : jaulas) {
                    buffer.print("Jaula " + j.getId() + " limpiada por " +
j.getLimpiador().getNombre() + ".\n");
                }
            } finally {
                if (buffer != null) {
                    buffer.close();
                }
                if (escritor != null) {
                    escritor.close();
                }
            }
        } catch (FileNotFoundException e) {
            System.out.println("Se ha producido una FileNotFoundException" +
e.getMessage());
        } catch (IOException e) {
            System.out.println("Se ha producido una IOException" +
e.getMessage());
        } catch (Exception e) {
            System.out.println("Se ha producido una Exception" +
e.getMessage());
        }
    }
```

4) (Máx 2ptos.) En el proyecto se incluyen 2 ficheros de nombres `veterinarios.txt` y `cuidadores.dat` que contienen los datos básicos de varios nuevos empleados del zoo. El primero de ellos es un fichero de caracteres, mientras que el segundo es un fichero de bytes que contiene empleados cuidadores. Las clases `Cuidador` y `Veterinario` ya disponen de un método `data()` que indica el orden en que se encuentra cada campo de cada clase en el fichero correspondiente. Se pide implementar 2 funciones distintas, una para cada tipo de empleado, a las que se le pasa como argumento el nombre del fichero y que devuelven una colección de nuevos objetos importados desde ese fichero.

```
public static ArrayList<Cuidador> importarCuidadores (String path) {
        ArrayList<Cuidador> ret = new ArrayList<>();
        FileInputStream lector = null;
        ObjectInputStream lectorObjeto = null;
```

```
        try {
            try {
                lector = new FileInputStream(path);
                lectorObjeto = new ObjectInputStream(lector);
                Cuidador c;
                while ((c = (Cuidador) lectorObjeto.readObject()) != null) {
                    ret.add(c);
                }
            } finally {
                if (lectorObjeto != null) {
                    lectorObjeto.close();
                }
                if (lector != null) {
                    lector.close();
                }
            }
        } catch (FileNotFoundException e) {
            System.out.println("Se ha producido una FileNotFoundException" +
e.getMessage());
        } catch (EOFException e) {
            System.out.println("Final de fichero");
        } catch (IOException e) {
            System.out.println("Se ha producido una IOException: " +
e.getMessage());
        } catch (ClassNotFoundException e) {
            System.out.println("Se ha producido una ClassNotFoundException" +
e.getMessage());
        } catch (Exception e) {
            System.out.println("Se ha producido una Exception" +
e.getMessage());
        }
        return ret;
    }

public static ArrayList<Veterinario> importarVeterinarios (String path) {
        ArrayList<Veterinario> ret = new ArrayList<>();
        File fichero = new File(path);
        FileReader lector = null;
        BufferedReader buffer = null;
        try {
            try {
                lector = new FileReader(fichero);
                buffer = new BufferedReader(lector);
                String linea;
                while ((linea = buffer.readLine()) != null) {
                    //Se separa cada linea del fichero de acuerdo al orden
marcado en el método Veterinario.data()
                    String[] campos = linea.split("\\|");
                    int id = Integer.parseInt(campos[0]);
                    String nombre = campos[1];
                    String email = campos[2];
```

```
                    String telefono = campos[3];
                    int idvet = Integer.parseInt(campos[4]);
                    String especialidad = campos[5];
                    Veterinario v = new Veterinario(id, nombre, email,
telefono, idvet, especialidad);
                    ret.add(v);
                }
            } finally {
                if (buffer != null) {
                    buffer.close();
                }
                if (lector != null) {
                    lector.close();
                }
            }
        } catch (FileNotFoundException e) {
            System.out.println("Se ha producido una FileNotFoundException" +
e.getMessage());
        } catch (IOException e) {
            System.out.println("Se ha producido una IOException" +
e.getMessage());
        } catch (Exception e) {
            System.out.println("Se ha producido una Exception" +
e.getMessage());
        }
        return ret;
    }
```

5) (Máx 2ptos.) Función que se le pasa un `Informe` de un animal y busca en su campo `descripcion` si ha sido vacunado de alguna o de varias vacunas de las enunciadas en el ejercicio 2 de esta prueba. Esta función devolverá un `ArrayList` con los elementos de las vacunas que sí disponga el animal.

```
private static ArrayList<Vacunas> vacunasDe(Informe informe) {
        ArrayList<Vacunas> ret = new ArrayList<Vacunas>();
        for (Vacunas v : Vacunas.values()) {
            if (informe.getDescripcion().contains(v.getAbreviatura()) ||
informe.getDescripcion().contains(v.getNombre())) {
                ret.add(v);
            }
        }
        return ret;
    }
```

Especificaciones para la gestión del Zoológico:

El zoológico se encuentra ubicado en una llanura y se compone de 3 sectores (‘A’, ‘E’, ‘I’). El sector A es el más grande y consta de los recintos 1 (de aves) y 2. No tiene jaulas ya que en él viven animales en libertad, pero está electrificado en todos sus límites. El sector I consta únicamente del recinto 3, en el que están las jaulas de los insectos, anfibios y otros reptiles. La parte exterior del zoo la compone el sector E. Es un único recinto en el que se ubica la jaula para los animales de granja y las correspondientes a los 2 acuarios. Desde este recinto se accede o se sale del zoo, pasando justo al lado de la taquilla. También es aquí donde se sitúa la tienda. De los recintos se almacena en el sistema su id, la lista de jaulas que contiene, su encargado/a y su área (en metros cuadrados). Para cada jaula se guarda su id, su volumen y su limpiador/a. En el zoo se distinguen 2 tipos de animales, todos ellos con su propio identificador, nombre, especie a la que pertenece y la edad, así como un informe individual asociado a cada ejemplar:

- Animales en libertad, que viven en el recinto 1 (las aves) o en el recinto 2. Para este tipo de animales se guarda su fecha de liberación en el zoo. Cada recinto tiene un único empleado que es encargado del mismo.
- Animales en Jaula, de los que se guarda propiamente la generación a la que pertenecen dentro del zoo y la jaula en la que viven. Cada jaula tiene su propio empleado limpiador.

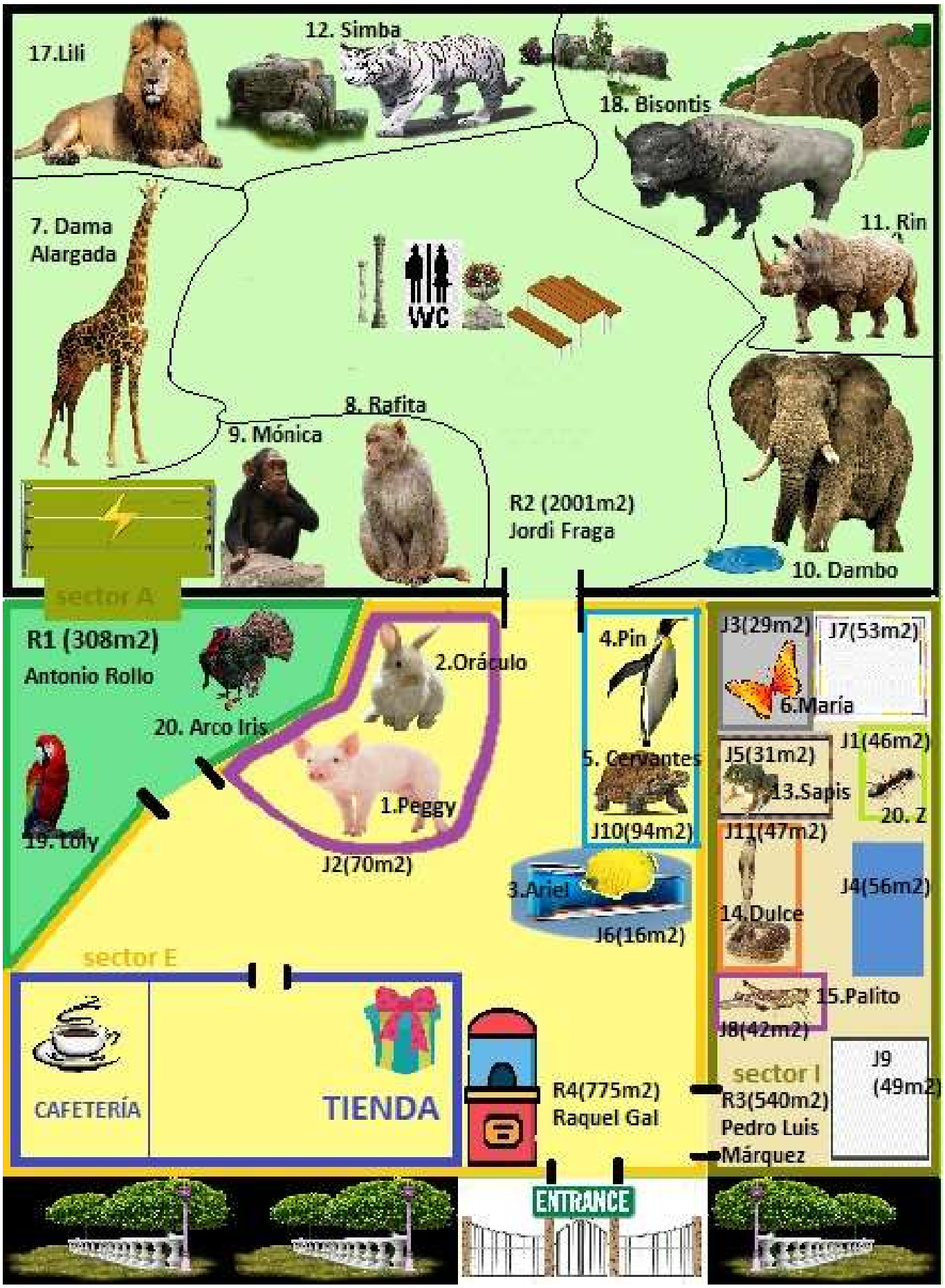
Además de los animales, se tienen los empleados del zoo, de todos ellos se guarda su identificador propio, su nombre, email y teléfono. Los empleados pueden ser de 2 tipos:

- Veterinarios, con su identificador de veterinario individual, se almacena en el sistema su especialidad y la lista de los animales que cuida. Hay animales que son cuidados por varios veterinarios y otros que por ninguno. Así mismo, un mismo veterinario puede cuidar a varios animales o a ninguno.
- Cuidadores, de los que se guarda su experiencia. Este tipo de empleados pueden ser encargado/a de algún recinto y/o limpiador/a de alguna jaula.

Los datos sobre cuidadores (veterinarios-animales) y limpieza de Jaulas son los siguientes:

J1 limpiada por Pedro Luis Márquez Prieto
J2 limpiada por Marta Tosco Valdés
J3 limpiada por Laura Díez Silva
J4 limpiada por Pedro Luis Márquez Prieto
J5 limpiada por Laura Díez Silva
J6 limpiada por Raquel Gal Ordas
J7 limpiada por Pedro Luis Márquez Prieto
J8 limpiada por Laura Díez Silva
J9 limpiada por Pedro Luis Márquez Prieto
J10 limpiada por Raquel Gal Ordas
J11 limpiada por Laura Díez Silva

Vet1 cuida de los animales 8, 9
Vet2 cuida de los animales 1, 2, 3, 4, 5, 6, 12, 13, 17
Vet3 cuida de los animales 3, 4, 5, 8, 9, 12, 17, 18
Vet4 cuida de los animales 1, 2, 6, 13, 14, 15, 16
Vet5 cuida de los animales 19, 20
Los animales 7, 10, 11 no son cuidados por ningún veterinario.


17.Lili
12. Simba
18. Bisontis
7. Dama Alargada
11. Rin
WC
8. Rafita
9. Mónica
R2 (2001m2)
Jordi Fraga
10. Dambo
sector A
R1 (308m2)
Antonio Rollo
20. Arco Iris
2.Oráculo
1.Peggy
J2(70m2)
19. Loly
4.Pin
5. Cervantes
J10(94m2)
3.Ariel
J6(16m2)
J3(29m2)
J7(53m2)
6.María
J5(31m2)
13.Sapis
J1(46m2)
20. Z
J11(47m2)
14.Dulce
J4(56m2)
15.Palito
J8(42m2)
J9 (49m2)
sector I
R3(540m2)
Pedro Luis Márquez
sector E
CAFETERÍA
TIENDA
R4(775m2)
Raquel Gal
ENTRANCE