

UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE CIENCIAS PURAS Y NATURALES



“SISTEMA DE GESTIÓN DE VETERINARIA”

Grupo 11

Integrantes:

1. Aruquipa Ticona Samir Elias
2. Chambi Loza Angela Miriam
3. Gutierrez Bautista Patricia Maylen
4. Moya Choque Natalia Miroslava
5. Paye Burgoa Andrés Reynaldo
- 6 .Quispe Mamani Cristian
7. Yupanqui Huanca David Alejandro

Link Repositorio GitHub: <https://github.com/CraftRey01/Veterinaria.git>

Materia: PROGRAMACIÓN II – INF 121

Carrera: INFORMÁTICA

Fecha de Entrega: 28/01/2025

Sistema de Gestión de Veterinaria

Introducción

Este sistema de gestión veterinaria se presenta como una solución completa y adaptable, diseñada para optimizar las operaciones diarias y asegurar una administración eficaz de los recursos. La plataforma no solo mejora la eficiencia, sino que también proporciona una experiencia organizada y fluida tanto para los clientes como para el personal de la clínica. Su principal objetivo es satisfacer las necesidades específicas de una veterinaria moderna, ofreciendo funcionalidades que resuelven los problemas comunes de gestión.

El sistema de gestión veterinaria está diseñado para abordar las necesidades esenciales de una clínica. Para lograr este objetivo, se han implementado varias clases clave que representan las entidades del sistema, como:

Veterinaria: Representa la clínica veterinaria, abarcando la información esencial de la clínica y su lista de clientes.

Cliente: Describe a los clientes y almacena la información básica junto con la información de sus mascotas.

Animal: Representa a las mascotas del cliente, incluyendo su información básica, tratamiento activo e historial de consultas.

Gato y Perro: Subclases especializadas de Animal que incluyen atributos específicos para gatos y perros respectivamente.

Consulta: Define una consulta médica veterinaria, incluyendo todos los detalles sobre el motivo de la visita, el costo y una descripción de los síntomas presentados.

DocVeterinario: Representa a los veterinarios empleados en la clínica, almacenando su información personal y especialidades médicas.

Examen: Documenta los exámenes realizados a los animales, proporcionando una identificación, descripción y resultados de cada examen.

Tratamiento: Define los tratamientos administrados a los animales, incluyendo el período de tratamiento y los servicios y medicaciones aplicadas.

ArchAnimal: Gestiona el archivo de registro de animales, proporcionando métodos para registrar, eliminar, modificar y listar los animales en la base de datos.

ComparadorGenerico: Proporciona comparaciones genéricas para atributos específicos de las clases derivadas de Animal, como la longitud.

Este sistema no solo está diseñado para automatizar los procesos administrativos de la clínica veterinaria, sino también para mejorar la comunicación y coordinación con los propietarios de las mascotas. Además, permite un seguimiento integral de la salud de los animales a través de consultas y tratamientos organizados de manera eficiente.

Implementar este sistema representa un avance clave hacia una gestión más profesional y efectiva en las clínicas veterinarias, beneficiando tanto a los animales como a sus dueños.

1. Definición del Proyecto

1.1 Descripción General:

El sistema de Gestión de Veterinaria es un diseño para administrar eficientemente , el registro de información manual puede llegar a ser morosa y hasta llegar a ver algunos errores y este sistema nos proporciona el poder registrar de una manera mas sencilla los procesos de atención medica y cuidado de mascotas en la clínica Veterinaria. Este sistema esta diseñado para múltiples funcionalidades para gestionar información sobre pacientes (mascotas), los propietarios , los tratamientos , los exámenes y demás servicios que la veterinaria brindara.

En este sistema se permite el registro de pacientes (mascotas), juntamente con los dueños de cada uno en este caso los clientes. Además se gestionan las clases consulta y examen, asegurando una adecuada atención.

Utilizando la Programación Orientada A Objetos(POO), el sistema incluye las funciones de persistencia (gestión de archivos) para el almacenamiento de registros , herencia y composición para la estructura correcta entre las relaciones de cada clase.

Nuestra clase base es la clase Veterinaria contando con la agregación a cliente , de cliente con la agregación a animal , de animal con herencia a mascotas y a tratamiento y con composición a consulta .

Este sistema nos permite agregar tratamientos, registrar a los animales y proporcionarles tratamientos a cada uno , asi es un sistema con buen desarrollo para el buen manejo de la veterinaria.

1.2 Objetivos:

1.2.1.Objetivo General:

Desarrollar un sistema de Gestión de Veterinaria que permita administrar de manera eficiente la información de las mascotas, de los propietarios, de las consultas medicas , de los tratamientos y exámenes, donde estemos garantizando un mejor control de los servicios de la veterinaria, mediante la Programación Orientada Objetos(POO) , utilizando persistencia de datos.

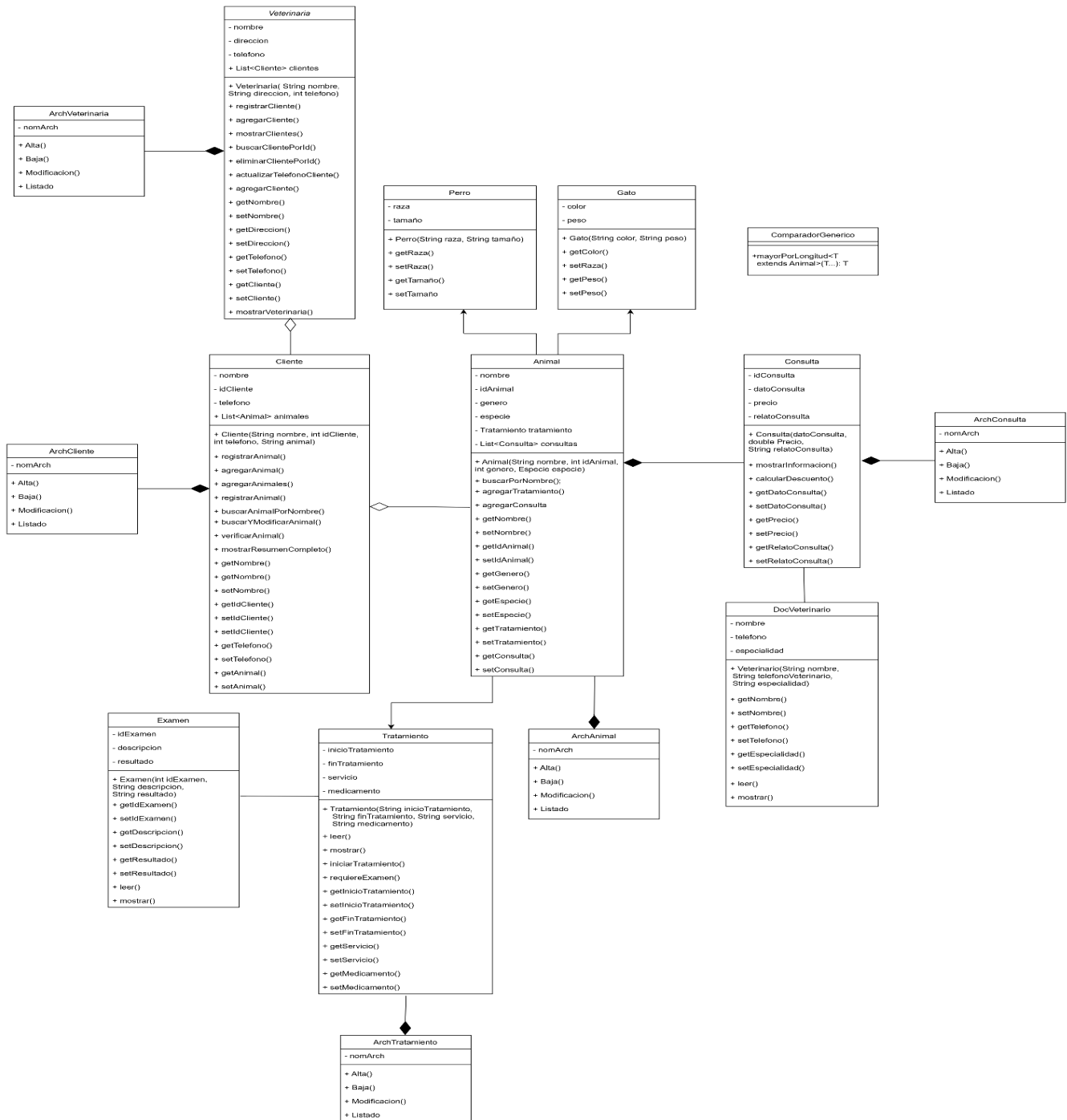
1.2.2 Objetivos Específicos:

- Administrar u Registrar las mascotas y mantener la información de las mascotas que fueron atendidas en la veterinaria.

- Registrar a los dueños y propietarios de cada mascota.
- Gestionar y registrar los diagnósticos , las consultas y los tratamientos de cada servicio.
- Controlar las citas y seguimientos de los pacientes (mascotas).
- tener una buena calidad de atención al cliente.

2. Análisis y Diseño

2.1 Diagrama UML



2.2 Principios de Diseño

Herencia: las clases gato y perro heredan de animal.

Composición: La clase consulta incluye un objeto de tipo Doc Veterinario

Agregación: Una Animal puede estar asociado a varias consultas, un Cliente puede estar asociado a varios animales y una Veterinaria puede estar asociada a varios clientes.

Genericidad: Uso de genericidad para gestionar listas de Consulta, Animal y Cliente.

Interfaces: La implementación de una interfaz gráfica a la Gestión de Veterinaria para operaciones básicas CRUD.

Persistencia: Se implementó en la clase ArchAnimal pero al usar la base de datos de sql ya no es necesario siendo así que es más fácil de trabajar con sql.

Patrones de diseño: Implementamos el Principio de Abierto/Cerrado, Principio de Responsabilidad Única, Principio de Sustitución de Liskov, Principio de Segregación de Interfaces.

3. Implementación en Java

3.1 Estructura del Proyecto

Paquete Modelo.-

El paquete modelo contiene todas las clases que representan el dominio del sistema, son las entidades principales dentro la gestión de una veterinaria. Estas encapsulan las propiedades y comportamientos básicos de los elementos del sistema, como animales, clientes y la veterinaria misma. Incluye:

- a) *Clase Animal*: Representa a los animales registrados en la veterinaria. Cada instancia contiene atributos como el ID, nombre, género, especie, tratamiento y listas de consultas. También es la clase de la que heredan las clases Perro y Gato, animales más específicos que atiende la veterinaria además que proporciona métodos para modificar atributos, agregar consultas, buscar animales por nombre y mostrar información detallada.
- b) *Clase Cliente*: Modela a los clientes de la veterinaria, asociando cada cliente con una lista de animales además de contener el ID, nombre y teléfono del mismo. Incluye métodos para registrar, buscar y modificar información de los animales del cliente, así como para mostrar un resumen completo de sus datos.
- c) *Clase Veterinaria*: Representa a la veterinaria como un todo. Incluye información básica como el nombre, dirección y teléfono, además de una lista de clientes. Proporciona métodos para registrar, buscar, modificar y eliminar clientes, permitiendo así su gestión.

Paquete Controlador.-

El paquete controlador es responsable de implementar la lógica de negocio del sistema, es el intermediario entre las clases y la interfaz de usuario, coordinando las operaciones necesarias para cumplir con los ejercicios propuestos del sistema:

- a) Encontrar al animal con el nombre más largo: Utiliza un comparador genérico para determinar el animal con el nombre más largo dentro de un arreglo.
- b) Buscar y modificar un animal por su nombre: Permite localizar un animal específico y actualizar su información en caso necesario.
- c) Eliminar un cliente por ID: Elimina un cliente por su id, mostrando después la lista actualizada.
- d) Buscar un cliente por id: Mostrando su información posteriormente.
- e) Modificar el teléfono de un cliente: Accediendo a la información del cliente.
- f) Registrar nuevos tratamientos: Se registran tratamientos para los animales, incluyendo fechas, tipos y medicamentos asociados.
- g) Verificar si un tratamiento requiere un examen: Esta funcionalidad valida si un tratamiento específico exige la realización de pruebas adicionales.
- h) Leer por teclado y mostrar nueva información de un veterinario
- i) Aplicar descuentos: Calcula nuevos precios de servicios, descuentos en porcentajes específicos.
- j) Registrar múltiples animales por cliente: Capacidad de añadir varios animales a un cliente.
- k) Mostrar un resumen completo del cliente: Presenta todos los detalles del cliente, incluyendo sus animales y tratamientos asociados.

Paquete Vista.-

App.java es la clase entrada de la interfaz gráfica utilizando la biblioteca Swing de Java, esta inicializa el formulario de selección (SelectionForm), que funciona como la ventana principal de interacción con el usuario, se presenta:

- Cliente: Formulario para agregar clientes y una opción para listar los clientes de la veterinaria, también una opción de eliminar clientes con un ID específico.
- Animal: Formulario para registrar, buscar animales por su nombre y una opción para mostrar el animal con el nombre más largo.
- Consulta: Formulario para registrar consultas.
- Tratamiento: Formulario para registrar tratamientos.

3.2 Código Fuente

Clase Animal

```
package com.mycompany.veterinaria;

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class Animal {

    private int idAnimal;

    private String nombre;

    private int genero;

    private String especie;

    private Tratamiento tratamiento;

    private List<Consulta> consultas;


    public Animal(int idAnimal, String nombre, int genero, String especie) {

        this.idAnimal = idAnimal;

        this.nombre = nombre;

        this.genero = genero;

        this.especie = especie;

        this.consultas = new ArrayList<>();

    }

    public String getNombre() {

        return nombre;

    }

    public void setNombre(String nombre) {

        this.nombre = nombre;

    }

    public int getIdAnimal() {
```

```
        return idAnimal;
    }

    public void setIdAnimal(int idAnimal) {
        this.idAnimal = idAnimal;
    }

    public int getGenero() {
        return genero;
    }

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

    public String getEspecie() {
        return especie;
    }

    public void setEspecie(String especie) {
        this.especie = especie;
    }

    public Tratamiento getTratamiento() {
        return tratamiento;
    }

    public void setTratamiento(Tratamiento tratamiento) {
        this.tratamiento = tratamiento;
    }

    public void agregarConsulta(Consulta consulta) {
        consultas.add(consulta);

        System.out.println("Consulta agregada para el animal " + nombre);
    }
}
```



```
}
```

```
public List<Consulta> getConsultas() {
```

```
    return consultas;
```

```
}
```

```
public void mostrarInformacion() {
```

```
    System.out.println("Informacion del animal:");
```

```
    System.out.println("1. ID Animal: " + idAnimal);
```

```
    System.out.println("2. Nombre: " + nombre);
```

```
    System.out.println("3. Genero: " + genero);
```

```
    System.out.println("4. Especie: " + especie);
```

```
}
```

```
// Parte 1
```

```
// 2) Buscar a el animal con el nombre 'x' y preguntar si requiere alguna modificación
```

```
public void modificarAtributo() {
```

```
    Scanner scanner = new Scanner(System.in);
```

```
    mostrarInformacion();
```

```
    System.out.println("\nInserte atributo a modificar (Ingrese el numero correspondiente)");
```

```
    System.out.println("1. ID Animal");
```

```
    System.out.println("2. Nombre");
```

```
    System.out.println("3. Genero");
```

```
    System.out.println("4. Especie");
```

```
    int opcion = scanner.nextInt();
```

```
    scanner.nextLine();
```

```
    switch (opcion) {
```

```
        case 1:
```

```

        System.out.print("Nuevo ID Animal: ");

        setIdAnimal(scanner.nextInt());

        break;

    case 2:

        System.out.print("Nuevo nombre: ");

        setNombre(scanner.nextLine());

        break;

    case 3:

        System.out.print("Nuevo Genero (0 = Hembra, 1 = Macho): ");

        setGenero(scanner.nextInt());

        break;

    case 4:

        System.out.print("Nueva Especie: ");

        setEspecie(scanner.nextLine());

        break;

    default:

        System.out.println("Opcion no valida.");

}

System.out.println("\n¡Atributo modificado con exito!");

mostrarInformacion();

}

public static Animal buscarPorNombre(List<Animal> animales, String nombreBuscado) {

    for (Animal animal : animales) {

        if (animal.getNombre().equalsIgnoreCase(nombreBuscado)) {

            return animal;

        }

    }

}

```

```
        }  
    }  
    return null;  
}  
}
```

Clase Cliente

```
package com.mycompany.veterinaria;
```

```
import java.util.ArrayList;
```

```
import java.util.List;
```

```
import java.util.Scanner;
```

```
public class Cliente {
```

```
    private int idCliente;
```

```
    private String nombre;
```

```
    private String telefono;
```

```
    private List<Animal> animales;
```

```
    public Cliente(int idCliente, String nombre, String telefono) {
```

```
        this.idCliente = idCliente;
```

```
        this.nombre = nombre;
```

```
        this.telefono = telefono;
```

```
        this.animales = new ArrayList<>();
```

```
    }
```

```
public String getNombre() {  
    return nombre;  
}
```

```
public void setNombre(String nombre) {  
    this.nombre = nombre;  
}
```

```
public int getIdCliente() {  
    return idCliente;  
}
```

```
public void setIdCliente(int idCliente) {  
    this.idCliente = idCliente;  
}
```

```
public String getTelefono() {  
    return telefono;  
}
```

```
public void setTelefono(String telefono) {  
    this.telefono = telefono;  
}
```

```
public void registrarAnimal(Animal animal) {  
    animales.add(animal);  
    System.out.println("Animal registrado: " + animal.getNombre());  
}
```

```

public List<Animal> getAnimales() {

    return animales;

}

// Parte 1

// 2) Buscar a el animal con el nombre 'x' y preguntar si requiere alguna modificación

public Animal buscarAnimalPorNombre(String nombre) {

    for (int i = 0; i < animales.size(); i++) {

        Animal animal = animales.get(i);

        if (animal.getNombre().equalsIgnoreCase(nombre)) {

            return animal;

        }

    }

    return null;

}

public void buscarYModificarAnimal(String nombreBuscado) {

    Animal x = buscarAnimalPorNombre(nombreBuscado);

    if (x != null) {

        System.out.println("\nAnimal encontrado:");

        x.mostrarInformacion();

        Scanner scanner = new Scanner(System.in);

        System.out.print("\nDesea modificar algun atributo: (s/n): ");

        String respuesta = scanner.nextLine();

        if (respuesta.equalsIgnoreCase("s")) {

            x.modificarAtributo();

        }

    } else {

```

```

        System.out.println("\nNo se encontro un animal con el nombre: " + nombreBuscado);
    }
}

// Parte 5

// 10) para agregar solo un animal

public void agregarAnimal(Animal animal) {
    System.out.println("Animal agregado al cliente " + nombre);
    this.registrarAnimal(animal);
}

// 10) para agregar mas de un animal

public void agregarAnimales(List<Animal> nuevosAnimales) {
    System.out.println("Animales agregados al cliente " + nombre);
    for (Animal animal : nuevosAnimales) {
        this.registrarAnimal(animal);
    }
}

// 11) Verificar si un animal esta registrado

public boolean verificarAnimal(String nombre) {
    for (Animal animal : animales) {
        if (animal.getNombre().equalsIgnoreCase(nombre)) {
            return true;
        }
    }
    return false;
}

// 12) mostrar resumen del cliente

```

```

public void mostrarResumenCompleto() {

    if (animales.isEmpty()) {

        System.out.println("El cliente " + nombre + " no tiene animales registrados.");

    } else {

        System.out.println("Resumen completo para el cliente: " + nombre);

        for (Animal animal : animales) {

            System.out.println("- Animal: " + animal.getNombre());

            System.out.println(" Tipo: " + animal.getEspecie());

            System.out.println(" Consultas:");

            for (Consulta consulta : animal.getConsultas()) {

                System.out.println("   - " + consulta.getDatoConsulta() + " Costo:" +
                consulta.getPrecio() + "bs");

            }

            if (animal.getTratamiento() != null) {

                System.out.println(" Tratamiento:");

                System.out.println("   - Servicio: " + animal.getTratamiento().getServicio());

                //      System.out.println("   - Desde: " +
                animal.getTratamiento().getFechaInicio());

                //      System.out.println("   - Hasta: " +
                animal.getTratamiento().getFechaFin());

            } else {

                System.out.println(" No hay tratamiento registrado.");

            }

        }

    }

}
}

```

Clase DocVeterinario

```
package com.mycompany.veterinaria;
```

```
import java.util.Scanner;
```

```
public class DocVeterinario {
```

```
    private String nombre;
```

```
    private String telefono;
```

```
    private String especialidad;
```

```
    public DocVeterinario(String nombre, String telefono, String especialidad) {
```

```
        this.nombre = nombre;
```

```
        this.telefono = telefono;
```

```
        this.especialidad = especialidad;
```

```
    }
```

```
    public String getNombre() {
```

```
        return nombre;
```

```
    }
```

```
    public void setNombre(String nombre) {
```

```
        this.nombre = nombre;
```

```
    }
```

```
    public String getTelefono() {
```

```
        return telefono;
```



```
}
```

```
public void setTelefono(String telefono) {  
    this.telefono = telefono;  
}
```

```
public String getEspecialidad() {  
    return especialidad;  
}
```

```
public void setEspecialidad(String especialidad) {  
    this.especialidad = especialidad;  
}
```

```
// Parte 4
```

```
// 8) Lee por teclado nueva información y muestra la nueva información del veterinario
```

```
public void leer() {  
    Scanner scanner = new Scanner(System.in);  
    System.out.print("Ingrese el nombre del veterinario: ");  
    nombre = scanner.nextLine();  
    System.out.print("Ingrese el teléfono del veterinario: ");  
    telefono = scanner.nextLine();  
    System.out.print("Ingrese la especialidad del veterinario: ");  
    especialidad = scanner.nextLine();  
}
```

```
public void mostrar() {
```

```
        System.out.println("Nombre: " + nombre);

        System.out.println("Especialidad: " + especialidad);

        System.out.println("Teléfono: " + telefono);

    }

}
```

Clase Examen

```
package com.mycompany.veterinaria;
```

```
import java.util.Scanner;
```

```
public class Examen {
```

```
    private int idExamen;
```

```
    String descripcion;
```

```
    String resultado;
```

```
    public Examen(int idExamen, String descripcion, String resultado) {
```

```
        this.idExamen = idExamen;
```

```
        this.descripcion = descripcion;
```

```
        this.resultado = resultado;
```

```
    }
```

```
    public int getIdExamen() {
```

```
        return idExamen;
```

```
    }
```

```
public void setIdExamen(int idExamen) {  
    this.idExamen = idExamen;  
}
```

```
public String getDescripcion() {  
    return descripcion;  
}
```

```
public void setDescripcion(String descripcion) {  
    this.descripcion = descripcion;  
}
```

```
public String getResultado() {  
    return resultado;  
}
```

```
public void setResultado(String resultado) {  
    this.resultado = resultado;  
}
```

```
public void mostrar() {  
    System.out.println("Id examen:" + idExamen);  
    System.out.println("descripcion:" + descripcion);  
    System.out.println("resultado:" + resultado);  
}
```

```

public void leer() {

    Scanner scanner = new Scanner(System.in);


    System.out.print("Ingrese el ID del examen: ");

    idExamen = scanner.nextInt();

    scanner.nextLine();


    System.out.print("Ingrese la descripcion del examen: ");

    descripcion = scanner.nextLine();


    System.out.print("Ingrese el resultado del examen: ");

    resultado = scanner.nextLine();

}
}

```

Clase ComparadorGenerico

```

package com.mycompany.veterinaria;

```

```

//Parte 1

```

```

// 1) Encontrar a el animal con el nombre más largo

```

```

public class ComparadorGenerico {


    public static <T extends Animal> T mayorPorLongitud(T... animales) {

        if (animales == null || animales.length == 0) {

            throw new IllegalArgumentException("Debe proporcionarse al menos un animal.");

        }

        T mayor = animales[0];
    }
}

```

```

    for (int i = 1; i < animales.length; i++) {
        if (animales[i].getNombre().length() > mayor.getNombre().length()) {
            mayor = animales[i];
        }
    }
    return mayor;
}
}

```

Clase Consulta

```

package com.mycompany.veterinaria;

public class Consulta {
    private int idConsulta;
    private String datoConsulta;
    private double precio;
    private String relatoConsulta;

    public Consulta(int idConsulta, String datoConsulta, double precio, String relatoConsulta)
    {
        this.idConsulta = idConsulta;
        this.datoConsulta = datoConsulta;
        this.precio = precio;
        this.relatoConsulta = relatoConsulta;
    }

    public int getIdConsulta() {
        return idConsulta;
    }

    public void setIdConsulta(int idConsulta) {
        this.idConsulta = idConsulta;
    }
}

```

```
}

public String getDatoConsulta() {

    return datoConsulta;

}

public void setDatoConsulta(String datoConsulta) {

    this.datoConsulta = datoConsulta;

}


public double getPrecio() {

    return precio;

}


public void setPrecio(double precio) {

    this.precio = precio;

}


public String getRelatoConsulta() {

    return relatoConsulta;

}


public void setRelatoConsulta(String relatoConsulta) {

    this.relatoConsulta = relatoConsulta;

}


// Parte 4

/*muestra informacion de la consulta*/
```

```

public void mostrarInformacion() {

    System.out.println("ID Consulta: " + idConsulta);

    System.out.println("Dato de Consulta: " + datoConsulta);

    System.out.println("Precio: " + precio);

    System.out.println("Relato de la Consulta: " + relatoConsulta);

}

```

// 9) da nuevo precio con x % descuento

```

public double calcularDescuento(double PorcentajeDescuento) {

    double descuento = (precio * PorcentajeDescuento) / 100;

    return precio - descuento;

}

}

```

Clase DataBaseConections

```

package com.mycompany.veterinaria;

```

```

import java.sql.Connection;

```

```

import java.sql.DriverManager;

```

```

import java.sql.SQLException;

```

```

public class DatabaseConnection { //jdbc:mysql://host:puerto/DataBaseName

```

```

    private static final String URL = "jdbc:mysql://localhost:3306/veterinaria";

```

```

    private static final String USER = "root";

```

```

    private static final String PASSWORD = "";

```

```
public static Connection getConnection() throws SQLException {  
    return DriverManager.getConnection(URL, USER, PASSWORD);  
}  
}
```

Clase Gato

```
package com.mycompany.veterinaria;  
  
public class Gato extends Animal {  
    private String color;  
    private double peso;  
  
    public Gato(String nombre, int idAnimal, int genero, String especie, String color, double peso) {  
        super(idAnimal, nombre, genero, especie);  
        this.color = color;  
        this.peso = peso;  
    }  
  
    public String getColor() {  
        return color;  
    }  
  
    public void setColor(String color) {  
        this.color = color;  
    }  
  
    public double getPeso() {  
        return peso;  
    }  
  
    public void setPeso(double peso) {
```



```
        this.peso = peso;
    }
}
```

Clase Perro

```
package com.mycompany.veterinaria;
```

```
public class Perro extends Animal {
```

```
    private String raza;
```

```
    private String tamaño;
```

```
    public Perro(String nombre, int idAnimal, int genero, String especie, String raza, String
    tamaño) {
```

```
        super(idAnimal, nombre, genero, especie);
```

```
        this.raza = raza;
```

```
        this.tamaño = tamaño;
```

```
    }
```

```
    public String getRaza() {
```

```
        return raza;
```

```
    }
```

```
    public void setRaza(String raza) {
```

```
        this.raza = raza;
```

```
    }
```

```
    public String getTamaño() {
```

```
        return tamaño;
    }

    public void setTamaño(String tamaño) {
        this.tamaño = tamaño;
    }
}
```

Clase Tratamiento

```
package com.mycompany.veterinaria;

import java.util.Scanner;

public class Tratamiento {

    private int idTratamiento;

    private String inicioTratamiento;

    private String finTratamiento;

    private String servicio;

    private String medicamento;


    public Tratamiento(int idTratamiento, String inicioTratamiento, String finTratamiento, String servicio,
String medicamento) {

        this.idTratamiento = idTratamiento;

        this.inicioTratamiento = inicioTratamiento;

        this.finTratamiento = finTratamiento;

        this.servicio = servicio;

        this.medicamento = medicamento;
    }

    public int getIdTratamiento() {

        return idTratamiento;
    }
}
```

```
public void setIdTratamiento(int idTratamiento) {  
    this.idTratamiento = idTratamiento;  
}  
  
public String getInicioTratamiento() {  
    return inicioTratamiento;  
}  
  
public void setInicioTratamiento(String inicioTratamiento) {  
    this.inicioTratamiento = inicioTratamiento;  
}  
  
public String getFinTratamiento() {  
    return finTratamiento;  
}  
  
public void setFinTratamiento(String finTratamiento) {  
    this.finTratamiento = finTratamiento;  
}  
  
public String getServicio() {  
    return servicio;  
}  
  
public void setServicio(String servicio) {  
    this.servicio = servicio;  
}  
  
public String getMedicamento() {  
    return medicamento;  
}  
  
public void setMedicamento(String medicamento) {  
    this.medicamento = medicamento;  
}
```

```
public void leer() {  
  
    Scanner scanner = new Scanner(System.in);  
  
    System.out.print("Ingrese el inicio del tratamiento: ");  
  
    this.inicioTratamiento = scanner.nextLine();  
  
    System.out.print("Ingrese el fin del tratamiento: ");  
  
    this.finTratamiento = scanner.nextLine();  
  
    System.out.print("Ingrese el servicio: ");  
  
    this.servicio = scanner.nextLine();  
  
    System.out.print("Ingrese el medicamento: ");  
  
    this.medicamento = scanner.nextLine();  
}
```

```
public void mostrar() {  
  
    System.out.println("Inicio del tratamiento: " + inicioTratamiento);  
  
    System.out.println("Fin del tratamiento: " + finTratamiento);  
  
    System.out.println("Servicio: " + servicio);  
  
    System.out.println("Medicamento: " + medicamento);  
}
```

// Parte 3

// 6) realizar un nuevo tratamiento

```
public void iniciarTratamiento(String inicio, String fin, String servicio, String medicamento) {  
  
    setInicioTratamiento(inicio);  
  
    setFinTratamiento(fin);  
}
```

```

        setServicio(servicio);

        setMedicamento(medicamento);


        System.out.println("Nuevo tratamiento configurado.");
        System.out.println("Inicio: " + getInicioTratamiento());
        System.out.println("Fin: " + getFinTratamiento());
        System.out.println("Servicio: " + getServicio());
        System.out.println("Medicamento: " + getMedicamento());
    }


    // 7) Verificar si el tratamiento requiere un examen
    public boolean requiereExamen() {
        return getServicio().equalsIgnoreCase("Consulta Especializada");
    }
}

```

Clase Veterinaria (CLASE BASE)

```
package com.mycompany.veterinaria;
```

```
import java.util.ArrayList;
```

```
import java.util.List;
```

```
public class Veterinaria {
```

```
    private int idVeterinaria;
```

```
    private String nombre;
```

```
    private String direccion;
```

```
    private String telefono;
```

```
private List<Cliente> clientes;
```

```
public Veterinaria(int idVeterinaria, String nombre, String direccion, String telefono) {  
    this.idVeterinaria = idVeterinaria;  
    this.nombre = nombre;  
    this.direccion = direccion;  
    this.telefono = telefono;  
    this.clientes = new ArrayList<>();  
}
```

```
public int getIdVeterinaria() {  
    return idVeterinaria;  
}
```

```
public void setIdVeterinaria(int idVeterinaria) {  
    this.idVeterinaria = idVeterinaria;  
}
```

```
public String getNombre() {  
    return nombre;  
}
```

```
public void setNombre(String nombre) {  
    this.nombre = nombre;  
}
```

```
public String getDireccion() {  
    return direccion;  
}
```

```
public void setDireccion(String direccion) {  
    this.direccion = direccion;  
}
```

```
public String getTelefono() {  
    return telefono;  
}
```

```
public void setTelefono(String telefono) {  
    this.telefono = telefono;  
}
```

```
public List<Cliente> getClientes() {  
    return clientes;  
}
```

```
public void setClientes(List<Cliente> clientes) {  
    this.clientes = clientes;  
}
```

```
@Override
```

```
public String toString() {
```

```
        return "Veterinaria{" + "idVeterinaria=" + idVeterinaria + ", nombre=" + nombre + ",  
direccion=" + direccion + ", telefono=" + telefono + "}";  
    }
```

```
public void mostrarVeterinaria() {  
    System.out.println(toString());  
}
```

//Parte 2

// 3) registrar cliente y mostrar clientes

```
public void registrarCliente(Cliente cliente) {  
    agregarCliente(cliente);  
    System.out.println("Cliente registrado: " + cliente.getNombre());  
}
```

```
public void agregarCliente(Cliente nuevoCliente) {  
    List<Cliente> nuevaListaClientes = new ArrayList<>(clientes);  
    nuevaListaClientes.add(nuevoCliente);  
    clientes = nuevaListaClientes;  
}
```

```
public void mostrarClientes() {  
    System.out.println("Lista de Clientes de la " + nombre + " :");  
    for (Cliente cliente : clientes) {  
        System.out.println("- " + cliente.getNombre() + " (ID: " + cliente.getIdCliente() + ",  
Teléfono: " + cliente.getTelefono() + ")");  
    }  
}
```



```

// 4) buscar cliente por id

public Cliente buscarClientePorId(int id) {

    for (Cliente cliente : clientes) {

        if (cliente.getIdCliente() == id) {

            System.out.println("Cliente encontrado: " + cliente.getNombre() + " ID: " +
cliente.getIdCliente());

            return cliente;

        }

    }

    System.out.println("Cliente con ID " + id + " no encontrado.");

    return null;

}

```

```

// 3) eliminar cliente por id

public boolean eliminarClientePorId(int id) {

    for (Cliente cliente : clientes) {

        if (cliente.getIdCliente() == id) {

            clientes.remove(cliente);

            System.out.println("Cliente eliminado: " + cliente.getNombre());

            return true;

        }

    }

    System.out.println("Cliente con ID " + id + " no encontrado.");

    return false;

}

```

```

// 5) modificar Telefono de cliente

public boolean actualizarTelefonoCliente(int id, String nuevoTelefono) {

    for (Cliente cliente : clientes) {

        if (cliente.getIdCliente() == id) {

            cliente.setTelefono(nuevoTelefono);

            System.out.println("Teléfono actualizado para " + cliente.getNombre() + ": " +
nuevoTelefono);

            return true;

        }

    }

    System.out.println("Cliente con ID " + id + " no encontrado.");

    return false;

}
}

```

Clase Main:

```

package com.mycompany.veterinaria;

import java.util.ArrayList;

import java.util.List;

public class Main {

    public static void main(String[] args) {

        // Crear una veterinaria

        System.out.println("-----VETERINARIA-----");

        Veterinaria veterinaria = new Veterinaria(1, "Veterinaria Central", "Calle Principal 123",
"555-1234");

        veterinaria.mostrarVeterinaria();

        //Crear clientes
    }
}

```

```

Cliente cliente = new Cliente(1, "Juan Perez", "48541521");

Cliente cliente2 = new Cliente(2, "Maria Gomez", "98765432");

Cliente cliente3 = new Cliente(3, "Carlos Lopez", "45612378");

Cliente cliente4 = new Cliente(4, "Luisa Fernandez", "53790524");

veterinaria.registrarCliente(cliente);

veterinaria.registrarCliente(cliente2);

veterinaria.registrarCliente(cliente3);

veterinaria.registrarCliente(cliente4);

System.out.println("\n-----Clientes de la Veterinaria-----");

veterinaria.mostrarClientes();

// Crear animales

System.out.println("\n-----Animales de la Veterinaria-----");

Perro perro = new Perro("Max", 101, 1, "Canino", "Labrador", "Grande");

Gato gato = new Gato("Luna", 102, 2, "Felino", "Blanco", 4.5);

Animal elefante = new Animal(103, "Elefante", 1, "Elefante");

Animal conejo = new Animal(104, "Nina", 2, "Conejo");

cliente.registrarAnimal(perro);

cliente2.registrarAnimal(gato);

cliente3.registrarAnimal(elefante);

cliente4.registrarAnimal(conejo);

// Crear un veterinario

System.out.println("\n-----Veterinarios-----");

DocVeterinario veterinario = new DocVeterinario("Dra. Martinez", "555-9876",
"Cirugía");

DocVeterinario veterinario2 = new DocVeterinario("Dr. Edwin Ramos", "12345678",
"Dermatología");

veterinario.mostrar();

```

```

// Crear una consulta

System.out.println("\n-----Consultas de la Veterinaria-----");

Consulta consulta = new Consulta(1, "Chequeo general", 50.0, "El animal está
saludable.");

perro.agregarConsulta(consulta);

Consulta consulta2 = new Consulta(2, "Consulta General", 75.0, "El paciente presenta
síntomas de resfriado.");

gato.agregarConsulta(consulta2);

Consulta consulta3 = new Consulta(3, "Chequeo anual", 40.0, "Chequeo completo,
todo en orden.");

elefante.agregarConsulta(consulta3);

Consulta consulta4 = new Consulta(4, "Consulta general", 50.0, "El conejo presenta
signos de estrés.");

conejo.agregarConsulta(consulta4);

// Crear un tratamiento

System.out.println("\n-----Tratamientos-----");

Tratamiento tratamiento = new Tratamiento(1, "2025-01-01", "2025-01-10",
"Vacunación", "Vacuna Rabia");

perro.setTratamiento(tratamiento);

System.out.println("Tratamiento asignado al perro: " + tratamiento.getServicio());

tratamiento.mostrar();

Tratamiento tratamiento2 = new Tratamiento(2, "2025-01-15", "2025-01-20",
"tratamiednto", "paracetamol");

gato.setTratamiento(tratamiento2);

System.out.println("Tratamiento asignado al gato: " + tratamiento2.getServicio());

tratamiento2.mostrar();

```

```
Tratamiento tratamiento3 = new Tratamiento(3, "2025-02-01", "2025-02-15", "Consulta General", "Antibioticos");
```

```
elefante.setTratamiento(tratamiento3);
```

```
System.out.println("Tratamiento asignado al elefante: " + tratamiento3.getServicio());
```

```
tratamiento3.mostrar();
```

```
Tratamiento tratamiento4 = new Tratamiento(4, "2025-03-01", "2025-03-10", "Antiestrés", "Sedante natural");
```

```
conejo.setTratamiento(tratamiento4);
```

```
System.out.println("Tratamiento asignado al conejo: " + tratamiento4.getServicio());
```

```
tratamiento4.mostrar();
```

```
System.out.println("Exámenes: ");
```

```
Examen e = new Examen(123, "consulta", "bien");
```

```
e.mostrar();
```

```
System.out.println("\n-----");
```

```
// 1) Encontrar a el animal con el nombre más largo
```

```
System.out.println("\n1) ENCONTRAR AL ANIMAL CON EL NOMBRE MAS LARGO.");
```

```
Animal[] animales = {perro, gato, elefante, conejo};
```

```
Animal aNomMasLargo = ComparadorGenerico.mayorPorLongitud(animales);
```

```
System.out.println("El nombre más largo es: " + aNomMasLargo.getNombre());
```

```
// 2) Buscar a el animal con el nombre 'x' y preguntar si requiere alguna modificación
```

```
System.out.println("\n2) Hallar al animal con nombre 'X' y preguntar si requiere alguna modificacion");
```

```
cliente2.buscarYModificarAnimal("Luna");
```

```
// 3) eliminar cliente por id
```

```
System.out.println("\n3) Eliminar cliente por id ");
```

```
veterinaria.eliminarClientePorId(3);
```

```
veterinaria.mostrarClientes();
```

```
// 4) Buscar un cliente por id

System.out.println("\n4) Buscar un cliente por id");

veterinaria.buscarClientePorId(4);

// 5) Modificar Telefono del Cliente

System.out.println("\n5) Modoficar telefono del Cliente");

veterinaria.actualizarTelefonoCliente(2, "123456");

veterinaria.mostrarClientes();

// 6) Realizar un nuevo tratamiento

System.out.println("\n6) Realizar un nuevo tratamiento");

tratamiento3.iniciarTratamiento("2025-03-01", "2025-03-10", "Consulta Especializada",
"Analgesicos");

// 7) Verificar si el tratamiento requiere un examen

System.out.println("\n7) Verificar si el tratamiento requiere un examen");

if (tratamiento3.requiereExamen()) {

    System.out.println("El tratamiento requiere un examen.");

} else {

    System.out.println("El tratamiento no requiere un examen.");

}

// 8) Lee por teclado nueva información y muestra la nueva información del veterinario

System.out.println("\n8) Lee por teclado nueva información y muestra la nueva
información del veterinario ");

System.out.println("Información inicial del veterinario:");

veterinario2.mostrar();

System.out.println("Ingrese nueva información del veterinario:");

veterinario2.leer();

System.out.println("Información actualizada del veterinario:");

veterinario2.mostrar();
```

```
// 9) Da un nuevo precio con x % descuento
```

```
System.out.println("\n9) Da nuevo precio con x % descuento");
```

```
double nuevoPrecio = consulta.calcularDescuento(20);
```

```
System.out.println("Precio con descuento: " + nuevoPrecio);
```

```
// 10) Registrar mas de un animal en un cliente
```

```
System.out.println("\n10) Registrar mas de un animal en un cliente");
```

```
Perro perro2 = new Perro("Rocky", 103, 1, "Canino", "Pitbull", "Mediano");
```

```
Gato gato2 = new Gato("Mia", 104, 2, "Felino", "Gris", 3.2);
```

```
List<Animal> animalesCliente1 = new ArrayList<>();
```

```
animalesCliente1.add(gato);
```

```
animalesCliente1.add(gato2);
```

```
animalesCliente1.add(perro2);
```

```
cliente.agregarAnimales(animalesCliente1);
```

```
// 11) verificar si un animal esta registrado
```

```
System.out.println("\n11) Verificar si un animal esta registrado ");
```

```
String nombreAnimalABuscar = "Julio";
```

```
boolean existeAnimal1 = cliente.verificarAnimal(nombreAnimalABuscar);
```

```
System.out.println("El cliente " + cliente.getNombre() + " tiene un animal llamado " +  
nombreAnimalABuscar + "? \n" + (existeAnimal1 ? "Si" : "No"));
```

```
// 12) mostrar resumen del cliente
```

```
System.out.println("\n12) Mostrar resumen del cliente");
```

```

        cliente.mostrarResumenCompleto();
    }
}

```

3.3 Diseño de Interfaces.

Tablas de la Base de Datos Veterinaria:

Tabla	Acción	Filas	Tipo	Cotejamiento	Tamaño	Residuo a depurar
<input type="checkbox"/> animal	Examinar Estructura Buscar Insertar Vaciar Eliminar	4	InnoDB	utf8mb4_general_ci	32.0 KB	-
<input type="checkbox"/> cliente	Examinar Estructura Buscar Insertar Vaciar Eliminar	4	InnoDB	utf8mb4_general_ci	32.0 KB	-
<input type="checkbox"/> consulta	Examinar Estructura Buscar Insertar Vaciar Eliminar	4	InnoDB	utf8mb4_general_ci	32.0 KB	-
<input type="checkbox"/> tratamiento	Examinar Estructura Buscar Insertar Vaciar Eliminar	4	InnoDB	utf8mb4_general_ci	32.0 KB	-
4 tablas	Número de filas	16	InnoDB	utf8mb4_general_ci	128.0 KB	0 B

Estructura Cliente:

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
<input type="checkbox"/> 1	idCliente	int(11)			No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/> 2	nombre	varchar(100)	utf8mb4_general_ci		Sí	NULL			Cambiar Eliminar Más
<input type="checkbox"/> 3	telefono	varchar(15)	utf8mb4_general_ci		Sí	NULL			Cambiar Eliminar Más

Estructura Animal:

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
<input type="checkbox"/> 1	idAnimal	int(11)			No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/> 2	nombre	varchar(100)	utf8mb4_general_ci		Sí	NULL			Cambiar Eliminar Más
<input type="checkbox"/> 3	genero	int(11)			Sí	NULL			Cambiar Eliminar Más
<input type="checkbox"/> 4	especie	varchar(50)	utf8mb4_general_ci		Sí	NULL			Cambiar Eliminar Más
<input type="checkbox"/> 5	id_cliente	int(11)			Sí	NULL			Cambiar Eliminar Más

Estructura Consulta:

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
<input type="checkbox"/> 1	idConsulta	int(11)			No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/> 2	dato_consulta	varchar(150)	utf8mb4_general_ci		Sí	NULL			Cambiar Eliminar Más
<input type="checkbox"/> 3	precio	double			Sí	NULL			Cambiar Eliminar Más
<input type="checkbox"/> 4	relato_consulta	text	utf8mb4_general_ci		Sí	NULL			Cambiar Eliminar Más
<input type="checkbox"/> 5	id_animal	int(11)			Sí	NULL			Cambiar Eliminar Más

Estructura Tratamiento:

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
<input type="checkbox"/> 1	idTratamiento	int(11)			No	Ninguna			Cambiar Eliminar Más
<input type="checkbox"/> 2	inicio_tratamiento	date			Sí	NULL			Cambiar Eliminar Más
<input type="checkbox"/> 3	fin_tratamiento	date			Sí	NULL			Cambiar Eliminar Más
<input type="checkbox"/> 4	servicio	varchar(150)	utf8mb4_general_ci		Sí	NULL			Cambiar Eliminar Más
<input type="checkbox"/> 5	medicamento	varchar(150)	utf8mb4_general_ci		Sí	NULL			Cambiar Eliminar Más
<input type="checkbox"/> 6	id_animal	int(11)			Sí	NULL			Cambiar Eliminar Más


Ejecución Interfaz Gráfica

Menú:

Veterinaria	
Agregar Cliente	Listar Clientes
Agregar Animal	Listar Animales
Agregar Consulta	Agregar Tratamiento
1) Animal Con Nombre mas largo	2) Buscar Animal por nombre
3) Eliminar Cliente con ID X	4) Buscar Cliente por ID
5) Modificar Teléfono del Cliente X	6) Realizar Nuevo Tratamiento
7) Verificar si Tratamiento Requiere Examen	8) Actualizar Información del Veterinario
9) Aplicar Descuento a Consulta	10) Verificar si Animal Está Registrado

Agregar Cliente:

Registro Cliente	
ID Cliente:	5
Nombre:	Doofenshmirtz
Teléfono:	53425354
Guardar	Volver

Message	
	Cliente guardado correctamente!
OK	

Listado de Clientes

1		Juan Perez		48541521
2		Maria Gomez		98765432
3		Carlos Lopez		45612378
4		Luisa Fernandez		53790524
5		Doofenshmirtz		53425354


Volver

Agregar Animal:

Registro Animal

ID Animal:	105
Nombre:	Perry
Genero (1=Macho, 2=Hembra):	1
Especie:	Ornitorrinc
Guardar	Volver

Message

 **Animal guardado correctamente!**

OK

Listado de Animales

101		Max		Macho		Canino
102		Luna		Hembra		Felino
103		Elefante		Macho		Elefante
104		Nina		Hembra		Conejo
105		Perry		Macho		Ornitorrinco


Volver

Agregar Consulta:

Registro Consulta

ID Consulta:	5
Dato Consulta:	Consulta General
Precio:	35.0
Relato Consulta:	El paciente esta saludable
ID Animal:	105
Guardar	Volver


Message

 Consulta guardada correctamente!

OK

Agregar Tratamiento:

Registro Tratamiento	
ID Tratamiento:	5
Inicio Tratamiento:	2025-01-28
Fin Tratamiento:	2025-02-02
Servicio:	Consulta General
Medicamento:	Antibioticos
ID Animal:	105
<div>GuardarVolver</div>	

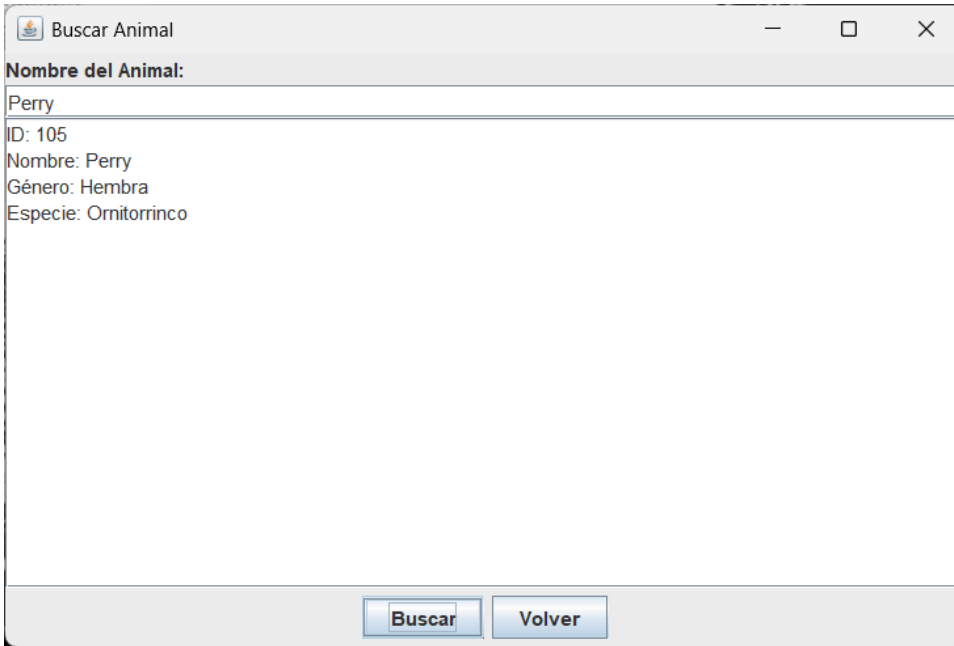
Message	
	Tratamiento guardado correctamente!
<div>OK</div>	

Ejercicios:

1) Encontrar a el animal con el nombre más largo.

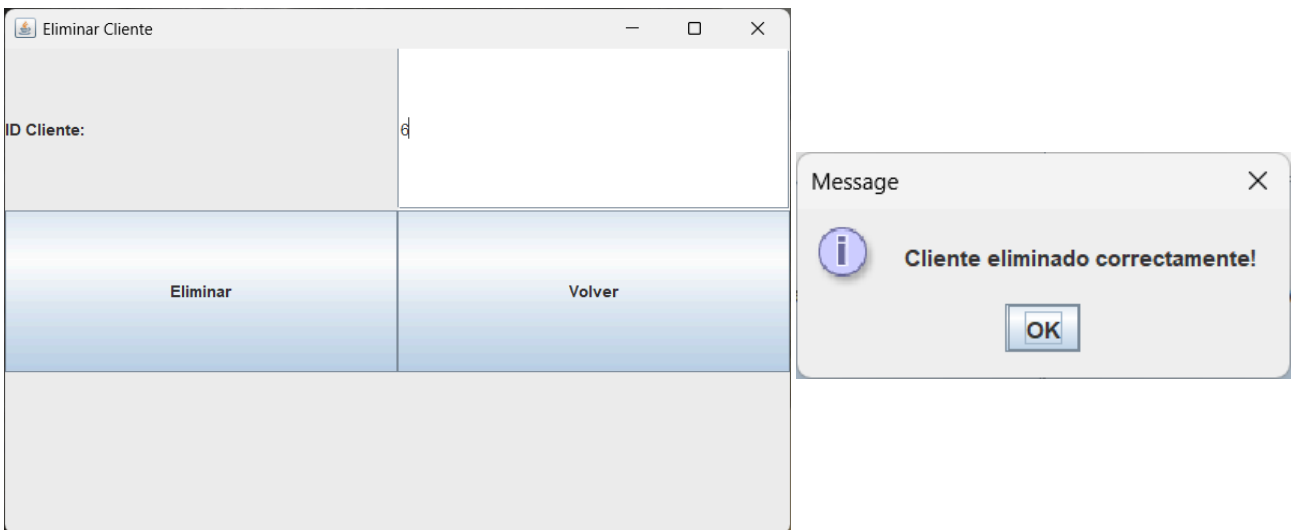
Animal con el Nombre Más Largo	
ID: 103	
Nombre: Elefante	
Género: Hembra	
Especie: Elefante	
<div>BuscarVolver</div>	

2) Buscar a el animal con el nombre 'x'.



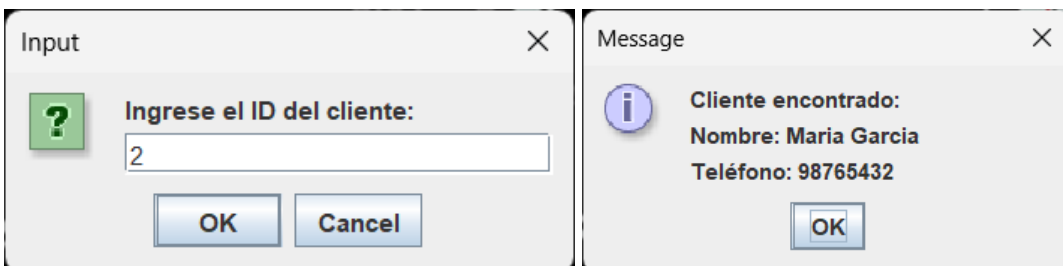
The screenshot shows a window titled "Buscar Animal". It has a label "Nombre del Animal:" followed by a text input field containing "Perry". Below the input field, the following details are displayed: "ID: 105", "Nombre: Perry", "Género: Hembra", and "Especie: Ornitorrinco". At the bottom of the window, there are two buttons: "Buscar" and "Volver".

3) Eliminar cliente por id.



The screenshot shows two windows. The main window is titled "Eliminar Cliente" and contains a label "ID Cliente:" followed by a text input field containing "6". Below the input field, there are two buttons: "Eliminar" and "Volver". To the right of this window is a smaller "Message" window with a blue information icon and the text "Cliente eliminado correctamente!". At the bottom of the message window is an "OK" button.


4) Buscar un cliente por id.



The screenshot shows two windows. The first window is titled "Input" and contains a green question mark icon, the label "Ingrese el ID del cliente:", and a text input field containing "2". Below the input field are "OK" and "Cancel" buttons. The second window is titled "Message" and contains a blue information icon, the text "Cliente encontrado:", and the details "Nombre: Maria Garcia" and "Teléfono: 98765432". At the bottom of the message window is an "OK" button.

5) Modificar Telefono del Cliente


Input

 Ingrese el ID del cliente:

3

OK Cancel


Input

 Ingrese el nuevo teléfono:

77766655

OK Cancel


Message

 **Telefono Actualizado:**
Nombre: Carlos Lopez
Teléfono: 77766655

OK

6) Realizar un nuevo tratamiento.

Registrar Nuevo Tratamiento

 **Inicio del Tratamiento:**
2025-02-02


Fin del Tratamiento:
2025-06-15

Servicio:
Antiestres

Medicamento:
Antibioticos

OK Cancel


Message

 **Tratamiento registrado:**
Inicio: 2025-02-02
Fin: 2025-06-15
Servicio: Antiestres
Medicamento: Antibioticos

OK

7) Verificar si el tratamiento requiere un examen

Message

 **¿Requiere examen? Si**

OK

8) Lee por teclado nueva información y muestra la nueva información del veterinario.

Actualizar Información del Veterinario

Nombre del Veterinario:
Mario

Teléfono:
54326346

Especialidad:
Cirugia

OK Cancel

Message

Veterinario actualizado:
Nombre: Mario
Teléfono: 54326346
Especialidad: Cirugia

OK

9) Da un nuevo precio con x % descuento.

Input

Ingrese el ID de la consulta:
4

OK Cancel

Input

Ingrese el porcentaje de descuento:
15

OK Cancel

Message

Precio original: 50.0
Precio con descuento: 42.5

OK

10) verificar si un animal esta registrado.

Input

Ingrese el nombre del animal:
Max

OK Cancel

Message

¿Está registrado el animal? Si

OK

3.5 Manejo de Archivos

-ANIMAL: Tabla base para nombre, género y especie almacena la información general de los animales.

Nombre: almacena el nombre del animal

Género: almacena el género del animal

Especie: almacena el tipo de especie del animal.

-CLIENTE: tabla base para: nombre y teléfono. Almacenar la información de los clientes.

Nombre del cliente, identificación única (ID), y número de teléfono.

Nombre: almacena el nombre del cliente.

Teléfono: almacena el teléfono del cliente.

-CONSULTA: tabla base para: dato_consulta, precio, relato_consulta y el id animal. Almacena los detalles de las consultas veterinarias realizadas a los animales.

Descripción de la consulta, costo, y relato de los síntomas o motivos de la consulta.

Dato_consulta: almacena la descripción de la consulta del animal.

Precio: almacena el costo de la consulta.

Relato_consulta: almacena el relato de los síntomas o motivos de la consulta.

-TRATAMIENTO: tabla base para: inicio_tratamiento, fin_tratamiento, servicio y medicamento. Almacenar la información específica de los tratamientos que se han aplicado a los animales.

Inicio_tratamiento: almacena la fecha del inicio del tratamiento.

Fin_tratamiento: almacena la fecha de finalización del tratamiento.

Servicio: almacena el tipo de servicio que se aplicó.

Medicamento: almacena los medicamentos administrados.

Cliente_Animal: tabla de relación muchos a muchos entre Cliente y Animal, almacenando las relaciones entre clientes y sus animales.

Tratamiento_Consulta: Tabla de relación muchos a muchos entre Tratamiento y Consulta, almacenando las relaciones entre los tratamientos y las consultas realizadas.

Veterinaria_Veterinario: tabla de relación muchos a muchos entre Veterinaria y DocVeterinario, almacenando las relaciones entre la clínica y sus veterinarios.

4. Pruebas del Sistema


```
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Veterinaria - Apache NetBeans IDE 21
Search (Ctrl+F)

cd C:\Users\Andres\Desktop\UMSA-2024\2doSem\prograInvierno\Veterinaria; "JAVA_HOME=C:\\Program Files\\Java\\jdk-23" cmd /c "%C:\\Program Files\\NetB
Scanning for projects...

-----< com.mycompany:Veterinaria >-----
Building Veterinaria 1.0-SNAPSHOT
from pom.xml
-----[ jar ]-----

--- resources:3.3.1:resources (default-resources) @ Veterinaria ---
skip non existing resourceDirectory C:\Users\Andres\Desktop\UMSA-2024\2doSem\prograInvierno\Veterinaria\src/main/resources

--- compiler:3.11.0:compile (default-compile) @ Veterinaria ---
Nothing to compile - all classes are up to date

--- exec:3.1.0:exec (default-cli) @ Veterinaria ---
-----VETERINARIA-----
Veterinaria(idVeterinaria=1, nombre=Veterinaria Central, direccion=Calle Principal 123, telefono=555-1234)
Cliente registrado: Juan Perez
Cliente registrado: Maria Gomez
Cliente registrado: Carlos Lopez
Cliente registrado: Luisa Fernandez

-----Clientes de la Veterinaria-----
Lista de Clientes de la Veterinaria Central:
- Juan Perez (ID: 1, Telefono: 48541521)
- Maria Gomez (ID: 2, Telefono: 98765432)
- Carlos Lopez (ID: 3, Telefono: 45612378)
- Luisa Fernandez (ID: 4, Telefono: 53790524)

-----Animales de la Veterinaria-----
Animal registrado: Max
Animal registrado: Luna
Animal registrado: Elefante
```

```
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Veterinaria - Apache NetBeans IDE 21
Search (Ctrl+F)

-----Animales de la Veterinaria-----
Animal registrado: Max
Animal registrado: Luna
Animal registrado: Elefante
Animal registrado: Nina

-----Veterinarios-----
Nombre: Dra. Martinez
Especialidad: Cirujano
Telefono: 555-9876

-----Consultas de la Veterinaria-----
Consulta agregada para el animal Max
Consulta agregada para el animal Luna
Consulta agregada para el animal Elefante
Consulta agregada para el animal Nina

-----Tratamientos-----
Tratamiento asignado al perro: Vacunacion
Inicio del tratamiento: 2025-01-01
Fin del tratamiento: 2025-01-10
Servicio: Vacunacion
Medicamento: Vacuna Rabia
Tratamiento asignado al gato: tratamiednto
Inicio del tratamiento: 2025-01-15
Fin del tratamiento: 2025-01-20
Servicio: tratamiednto
Medicamento: paracetamol
Tratamiento asignado al elefante: Consulta General
Inicio del tratamiento: 2025-02-01
Fin del tratamiento: 2025-02-15
Servicio: Consulta General
Medicamento: Antibioticos
```

```
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help Veterinaria - Apache NetBeans IDE 21 Search (Ctrl+F)
<default config> 308,6344,0000

Output - Run (Main)
Id examen:123
descripcion:consulta
resultado:bien

-----

1) ENCONTRAR AL ANIMAL CON EL NOMBRE MAS LARGO.
El nombre mas largo es: Elefante

2) Hallar al animal con nombre 'X' y preguntar si requiere alguna modificacion

Animal encontrado:
Informacion del animal:
1. ID Animal: 102
2. Nombre: Luna
3. Genero: 2
4. Especie: Felino

Desea modificar algun atributo: (s/n): s
Informacion del animal:
1. ID Animal: 102
2. Nombre: Luna
3. Genero: 2
4. Especie: Felino

Inserte atributo a modificar (Ingrese el numero correspondiente)
1. ID Animal
2. Nombre
3. Genero
4. Especie
2
Nuevo nombre: dogo
```

```
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help Veterinaria - Apache NetBeans IDE 21 Search (Ctrl+F)
<default config> 187,6344,0000

Output - Run (Main)
◆ Atributo modificado con exito!
Informacion del animal:
1. ID Animal: 102
2. Nombre: dogo
3. Genero: 2
4. Especie: Felino

3) Eliminar cliente por id
Cliente eliminado: Carlos Lopez
Lista de Clientes de la Veterinaria Central:
- Juan Perez (ID: 1, Telefono: 48541521)
- Maria Gomez (ID: 2, Telefono: 98765432)
- Luisa Fernandez (ID: 4, Telefono: 53790524)

4) Buscar un cliente por id
Cliente encontrado: Luisa Fernandez ID: 4

5) Modificar telefono del Cliente
Telefono actualizado para Maria Gomez: 123456
Lista de Clientes de la Veterinaria Central:
- Juan Perez (ID: 1, Telefono: 48541521)
- Maria Gomez (ID: 2, Telefono: 123456)
- Luisa Fernandez (ID: 4, Telefono: 53790524)

6) Realizar un nuevo tratamiento
Nuevo tratamiento configurado:
Inicio: 2025-03-01
Fin: 2025-03-10
Servicio: Consulta Especializada
Medicamento: Analgesicos

7) Verificar si el tratamiento requiere un examen
El tratamiento requiere un examen.
```

```
7) Verificar si el tratamiento requiere un examen.
El tratamiento requiere un examen.

8) Lee por teclado nueva informaci3n y muestra la nueva informaci3n del veterinario
Informaci3n inicial del veterinario:
Nombre: Dr. Edwin Ramos
Especialidad: Dermatolog3a
Tel3fono: 12345678
Ingrese nueva informaci3n del veterinario:
Ingrese el nombre del veterinario: Mario
Ingrese el tel3fono del veterinario: 4645776
Ingrese la especialidad del veterinario: Dermatalogo
Informaci3n actualizada del veterinario:
Nombre: Mario
Especialidad: Dermatalogo
Tel3fono: 4645776

9) Da nuevo precio con x % descuento
Precio con descuento: 40.0

10) Registrar mas de un animal en un cliente
Animales agregados al cliente Juan Perez
Animal registrado: dogo
Animal registrado: Mia
Animal registrado: Rocky

11) Verificar si un animal esta registrado
El cliente Juan Perez tiene un animal llamado Julio?
No

12) Mostrar resumen del cliente
Resumen completo para el cliente: Juan Perez
- Animal: Max
```

```
11) Verificar si un animal esta registrado
El cliente Juan Perez tiene un animal llamado Julio?
No

12) Mostrar resumen del cliente
Resumen completo para el cliente: Juan Perez
- Animal: Max
  Tipo: Canino
  Consultas:
    - Chequeo general Costo:50.0bs
  Tratamiento:
    - Servicio: Vacunaci3n
- Animal: dogo
  Tipo: Felino
  Consultas:
    - Consulta General Costo:75.0bs
  Tratamiento:
    - Servicio: tratamiednto
- Animal: Mia
  Tipo: Felino
  Consultas:
    No hay tratamiento registrado.
- Animal: Rocky
  Tipo: Canino
  Consultas:
    No hay tratamiento registrado.

-----
BUILD SUCCESS
-----
Total time: 42.862 s
Finished at: 2025-01-28T12:08:23-04:00
-----
```

5. Conclusi3n

El sistema elaborado para la gesti3n de una veterinaria ha resultado ser un proceso enriquecedor que nos permiti3 implementar e integrar principios de la Programaci3n Orientada a Objetos (POO), desde el dise1o de las clases hasta la inclusi3n de funcionalidades espec3ficas, el proyecto est3 orientado a resolver necesidades de una veterinaria.

El proyecto presenta la implementación de diferentes funcionalidades, como la búsqueda y modificación de animales y clientes, la eliminación de registros por ID, la gestión de múltiples animales por cliente y la posibilidad de registrar tratamientos de manera detallada. Estas características no solo aportan practicidad al sistema, sino que también demuestran la capacidad del proyecto para adaptarse a las operaciones diarias de una veterinaria.

El diseño del sistema hace uso de la herencia, asociación, agregación y la composición, logrando una jerarquía de clases clara que facilita la comprensión y la implementación del código. Además, se incluye un método genérico para resolver problemas específicos (como encontrar el animal con el nombre más largo) y el manejo de excepciones para asegurar la estabilidad del sistema ante posibles errores.

Aunque el sistema aún cuenta con áreas por perfeccionar y aspectos que podrían ser optimizados o ampliados en el futuro, destacamos los avances logrados.

Finalmente, el desarrollo de este sistema ha sido una valiosa oportunidad para aplicar conocimientos teóricos en un contexto práctico, resolviendo problemas reales y diseñando soluciones orientadas al usuario.