

PRIMER TALLER DE PARADIGMA ORIENTADO HACIA OBJETOS

TEMA: Diagrama UML

ESTUDIANTE: Samuel David Colmenarez Quero

CARRERA: Ingeniería de Software

GRUPO: TERRITORIAL 402

PROFESOR: Chritian David Jaimes Acevedo

Introducción

El objetivo de ambos ejercicios, además de fortalecer la sintaxis del estudiante en el lenguaje Java, es desarrollar su capacidad para diseñar diagramas de clases UML y diagramas de casos de uso. Estos diagramas servirán como apoyo para que el estudiante pueda determinar por dónde comenzar a escribir su código.

A continuación, se presentan los diagramas de clases UML y los diagramas de casos de uso correspondientes a ambos ejercicios.

Diagrama UML del primer ejercicio

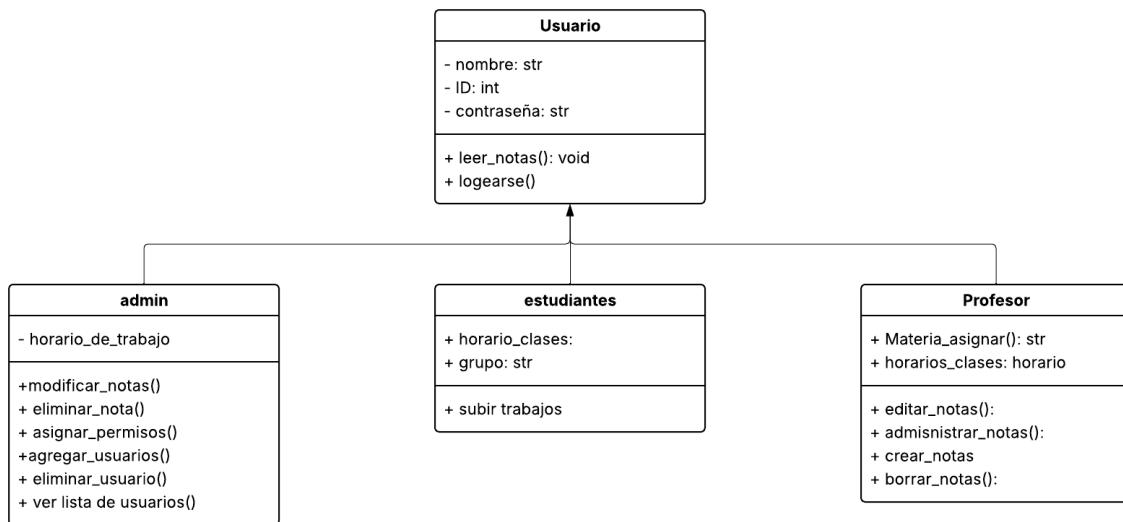


Diagrama de casos de uso del primer ejercicio

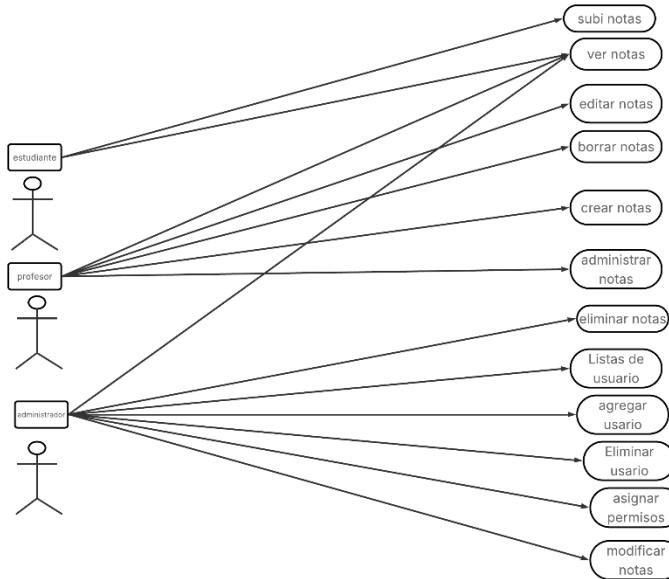


Diagrama UML del Segundo ejercicio

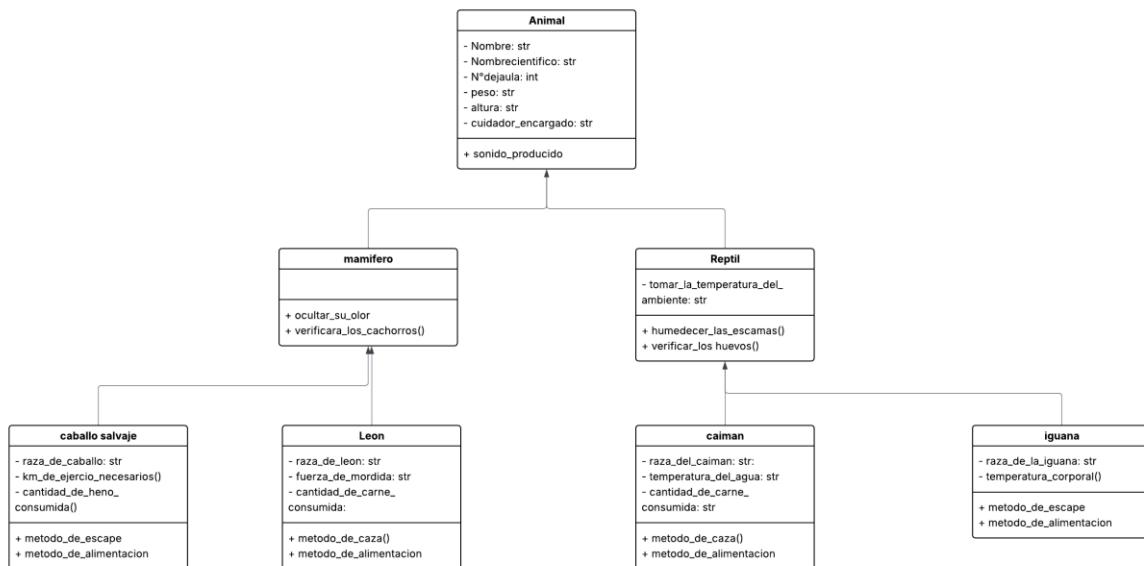
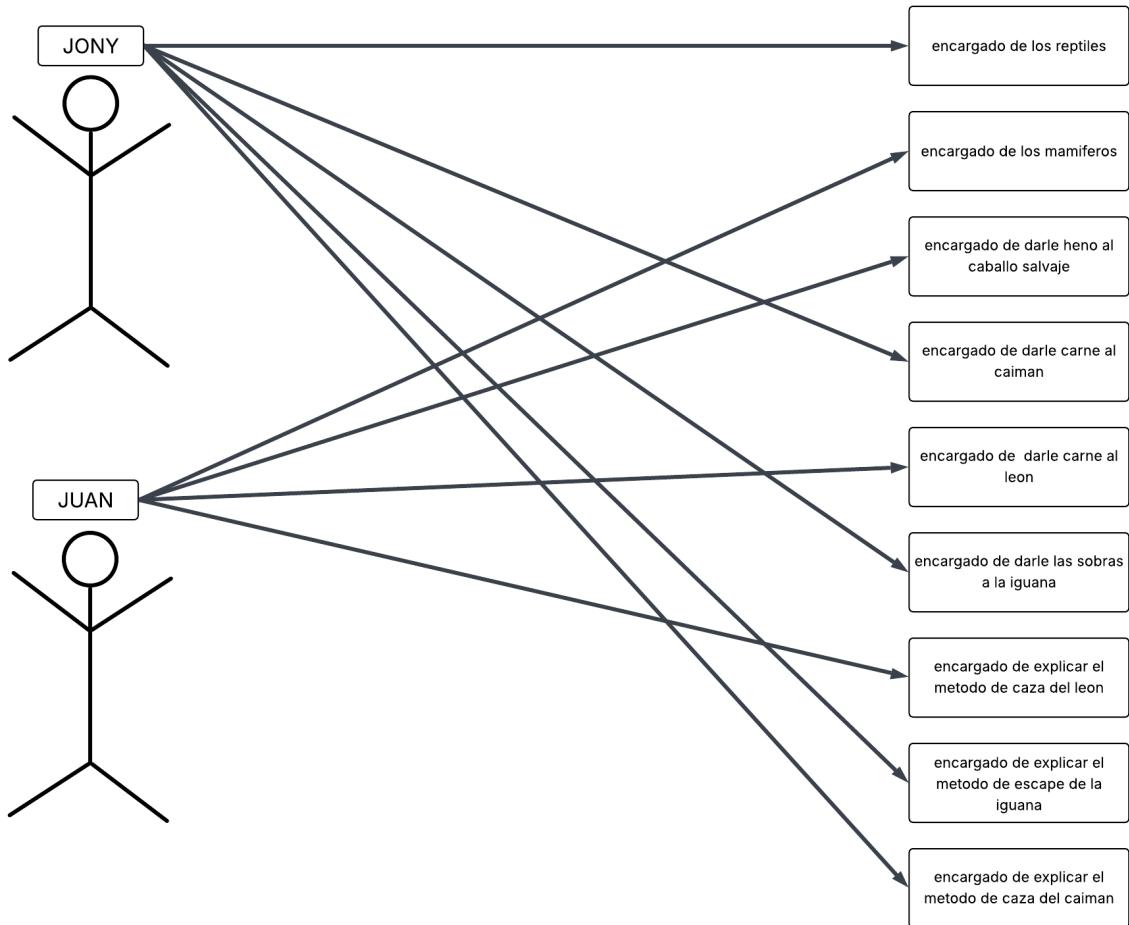
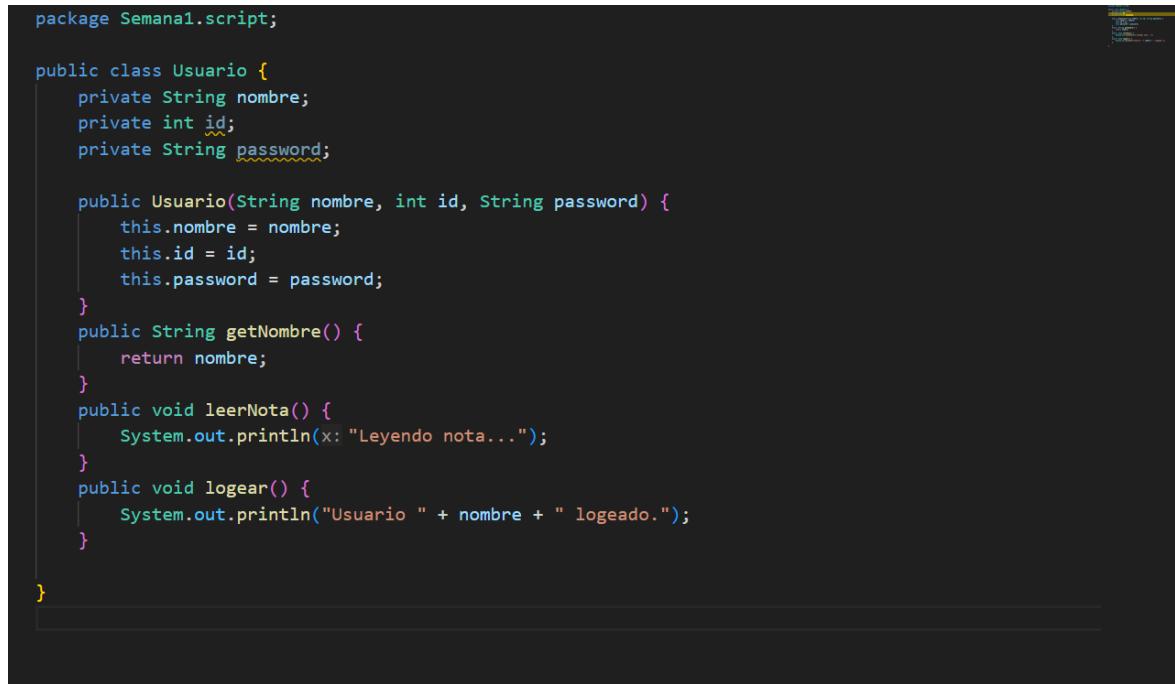


Diagrama de casos de uso de segundo ejercicio



Explicación del código de ambos ejercicios (Ejercicio 1)

Clase de usuario



```
package Semana1.script;

public class Usuario {
    private String nombre;
    private int id;
    private String password;

    public Usuario(String nombre, int id, String password) {
        this.nombre = nombre;
        this.id = id;
        this.password = password;
    }
    public String getNombre() {
        return nombre;
    }
    public void leerNota() {
        System.out.println("Leyendo nota...");
    }
    public void logear() {
        System.out.println("Usuario " + nombre + " logeado.");
    }
}
```

Debido a que se trata de la clase principal, decidí incluir solo algunos atributos privados, como nombre, id y password. Además, implementé el primer constructor y añadí tres métodos que estarán presentes a lo largo de las demás clases.

Clase de administrador

```
1 package Semana1.script;
2
3 public class Administrador extends Usuario {
4
5     private String horarioTrabajo;
6     public Administrador(String nombre, int id, String password) {
7         super(nombre, id, password);
8     }
9
9     public void setHorarioTrabajo(String horarioTrabajo) {
10         this.horarioTrabajo = horarioTrabajo;
11     }
12     public String getHorarioTrabajo() {
13         return horarioTrabajo;
14     }
15
16     public void asignarPermisos() {
17         System.out.println("Asignando permisos de administrador...");
18     }
19     public void editarNotas()
20     {
21         System.out.println("Editando notas...");
22     }
23
24     public void eliminarNota()
25     {
26         System.out.println("Eliminando nota...");
27     }
28     public void crearNota()
29     {
30         System.out.println("Creando nota...");
31     }
32 }
```

Al tratarse de la clase Administrador, solo fue necesario incluir un atributo privado que fue HorarioTrabajo, por lo que pude concentrarme más en la implementación de los métodos, como editarNotas, eliminarNotas y asignarPermisos, junto con sus respectivos mensajes mediante println.

Clase de profesor

```
package Semana1.script;
public class Profesor extends Usuario{
    private String horarios_de_clase;
    private String asignar_materia;

    public Profesor(String nombre, int id, String password, String horarios_de_clase, String asignar_
        super(nombre, id, password);
    }
    public void asignar_materia(String asignar_materia) {
        System.out.println(x: "asignando las notas...");
    }
    public void editarNotas() {
        System.out.println(x: "comenzando a editar notas...");
    }
    public void Administrar_notas() {
        System.out.println(x: "Administrando notas del profesor...");
    }
    public void Crear_notas() {
        System.out.println(x: "Creando notas del profesor...");
    }
    public void Borrar_notas() {
        System.out.println(x: "Borrando notas del profesor...");
    }
}
```

En la clase Profesor tuve que agregar otros atributos privados, como horariosDeClase y asignarMateria, los cuales añadí en el constructor. También implementé los métodos como asignarMateria, editarNotas, administrarNotas, crearNotas y borrarNotas, junto con sus respectivos mensajes mediante println.

Clase de estudiante

```
package Semana1.script;
public class Estudiante extends Usuario {
    private String horariodeclase;
    private String grupo;
    public Estudiante(String nombre, int id, String password, String horariodeclases, String grupo){
        super(nombre, id, password);
    }
    public void sethorariodeclase(String horariodeclase){
        this.horariodeclase = horariodeclase;
    }
    public String gethorariodeclase() {
        return this.horariodeclase;
    }
    public void subirnotas(){
        System.out.println(x: "subiendo notas del estudiante");
    }
}
```

En la clase Estudiante tuve que agregar dos nuevos atributos privados: horarioDeClase y grupo, los cuales incorporé al constructor original. Además, implementé un nuevo método llamado subirNotas, junto con su respectivo mensaje mediante println.

Clase Main

```
package Semana1.script;

public class Main {
    Run | Debug
    public static void main(String[] args) {
        System.out.println("Hello, World!");
        Usuario pepito = new Usuario(nombre: "Juan", id: 123, password: "hola");
        pepito.logear();
        pepito.leerNota();

        Administrador admin = new Administrador(nombre: "Admin", id: 1, password: "adminpass");
        admin.logear();
        admin.crearNota();

        Profesor profesor = new Profesor(nombre: "PROFE JUAN", id: 2, password: "Profe01", horarios_de_
        profesor.logear();
        profesor.Administrar_notas();
        profesor.editarNotas();
        profesor.Crear_notas();
        profesor.Borrar_notas();
        profesor.asignar_materia(asignar_materia: "matematicas");

        Estudiante estudiante =new Estudiante(nombre: "Juan", id: 123, password: "hola", horariodeclas
        estudiante.logear();
        estudiante.subirnotas();
    }
}
```

Esta podría considerarse la clase más sencilla de todas, ya que únicamente se encarga de invocar los elementos previamente establecidos en las demás clases.

Explicación del código de ambos ejercicios (Ejercicio 2)

Clase principal (animales)

```
public class animales {  
    private String nombre;  
    private String nombre_cientifico;  
    private int Numero_de_jaula;  
    private String peso;  
    private String altura;  
    private String cuidador_encargado;  
    public animales () {}  
    public animales (String nombre, String nombre_cientifico, int Numero_de_jaula, String peso, String  
        nombre_cientifico, int Numero_de_jaula, String peso, String  
        this.nombre=nombre;  
        this.nombre_cientifico= nombre_cientifico;  
        this.Numero_de_jaula= Numero_de_jaula;  
        this.peso= peso;  
        this.altura= altura;  
        this.cuidador_encargado= cuidador_encargado;  
    }  
    public String getnombre() {  
        return nombre;  
    }  
    public void presentacion() {  
        System.out.println("Han llegado los nuevos animales del zoológico, dos reptiles y dos mami  
    }
```

Al ser la clase principal y tratar un tema más específico relacionado con los animales, decidí hacerla un poco más extensa en comparación con el primer ejercicio. Incluí atributos como nombre, nombre científico, peso, número de jaula, altura y cuidador encargado, además de agregar métodos adicionales como presentación.

Clase mamífero (heredada de animales)

```
package Semana1.zoologico;  
  
public class mamiferos extends animales {  
    public mamiferos(String nombre, String nombre_cientifico, int numero_de_jaula, String peso, String  
        nombre_cientifico, int numero_de_jaula, String peso, String  
        cuidador_encargado) {  
    }  
  
    public static void ocultar_su_olor() {  
        System.out.println("Los mamíferos son expertos en ocultar su olor");  
    }  
    public void verificar_a_los_cachorros() {  
        System.out.println("Todo mamífero verifica a sus cachorros");  
    }  
}
```

Como indica el título, la clase Mamíferos hereda directamente de la clase Animales, por lo que también hereda su constructor. En este caso, se agregan los atributos privados

correspondientes a la clase Mamíferos; sin embargo, como esta no cuenta con nuevos atributos, se pasa directamente a la implementación de los métodos ocultar_su_olor() y verificar_a_los_cachorros().

Clase reptiles (heredada de animales)

```
package Semana1.zoologico;

public class reptiles extends animales {
    private String Tomarlatemperaturadelambiente;
    public reptiles(String nombre, String nombre_cientifico, int Numero_de_jaula, String peso, String altura, String cuidador_encargado) {
        super(nombre, nombre_cientifico, Numero_de_jaula, peso, altura, cuidador_encargado);
    }
    public void humedecer_las_escamas() {
        System.out.println("los reptiles son expertos en humedecer sus escamas");
    }
    public void verificar_a_los_Huevos() {
        System.out.println("todo reptil verifica a sus huevos");
    }
}
```

Al igual que su clase hermana Mamíferos, la clase Reptiles también hereda directamente de la clase principal, por lo que su constructor es heredado. Sin embargo, a diferencia de Mamíferos, la clase Reptiles sí cuenta con atributos privados, los cuales se añaden al constructor original. Además, se implementan los métodos propios de esta clase, como humedecer_las_escamas() y verificar_a_los_huevos().

Clase león (heredada de mamíferos)

```
3  public class leon extends mamiferos {
4      private String fuerzademordida;
5      private String cantidaddecarneconsumida;
6      public leon(String nombre, String nombre_cientifico, int Numero_de_jaula, String peso, String altura, String cuidador_encargado) {
7          super(nombre, nombre_cientifico, Numero_de_jaula, peso, altura, cuidador_encargado);
8      }
9
10     public void metododecaza() {
11         System.out.println("Los leones cazan en grupo, generalmente liderados por las leonas, que");
12     }
13     public void metododealimenta(){
14         System.out.println("Juan se mete al comedero de null mientras la jaula principal esta cerrada,");
15     }
16 }
17
18 }
```

La clase León hereda directamente de la clase Mamíferos, utilizando el constructor sin modificaciones. Además, se añaden nuevos atributos privados, como la raza del león, la fuerza de mordida y la cantidad de carne consumida. También se incorporan dos nuevos métodos: métododecaza() y métododealimenta().

Clase caballo (heredada de mamíferos)

```
package Semana1.zoologico;

public class caballo extends mamiferos {
    private String razadecaballo;
    private String velocidadlalcanzada;
    private String cantidaddehenoconsumido;

    public caballo(String nombre, String nombre_cientifico, int numero_de_jaula, String peso, String
        super(nombre, nombre_cientifico, numero_de_jaula, peso, altura, cuidador_encargado);

    }
    public void heno(){
        System.out.println("Juan prepara desde la madrugada los 15 kilos de heno para null");
    }
    public void metododeescape() {
        System.out.println("Los caballos salvajes escapan de los depredadores gracias a su gran ve
```

Al igual que la clase León, la clase Caballo también hereda directamente el constructor sin modificaciones de la clase Mamíferos, añadiendo sus propios atributos privados: raza del caballo, velocidad alcanzada y cantidad de heno consumida. Además, se incorporan dos métodos correspondientes: heno() y metododeescape().

Clase caimán (heredada de reptiles)

```
package Semana1.zoologico;

public class caiman extends reptiles {
    private String razadecaiman;
    private String fuerzademordida;
    private String cantidaddecarneconsumida;
    public caiman(String nombre, String nombre_cientifico, int Numero_de_jaula, String peso, String
        super(nombre, nombre_cientifico, Numero_de_jaula, peso, altura, cuidador_encargado, Tomarlat

    }
    public void metododecaza() {
        System.out.println("Los caimanes son cazadores sigilosos que utilizan la técnica del acech
    }
    public void metododealimenta(){
        System.out.println("Jony se mete a la jaula aquatica de null mientras la jaula principal esta
```

La clase Caimán hereda directamente de la clase Reptiles, heredando en el proceso el constructor modificado y agregando nuevos atributos privados: raza del caimán, fuerza de mordida y cantidad de carne consumida. Los métodos añadidos son: metododecaza() y metododealimenta().

Clase Iguana (heredada de reptiles)

```
package Semana1.zoologico;

public class iguana extends reptiles {
    private String razadeiguana;
    private String temperaturacorporal;
    public iguana(String nombre, String nombre_cientifico, int Numero_de_jaula, String peso, String cuidador_encargado, String Tomarlateperaturedelambiente, String razadeiguana, super(nombre, nombre_cientifico, Numero_de_jaula, peso, altura, cuidador_encargado, Tomarlateperaturedelambiente) {
    }
    public void metododeescape() {
        System.out.println(x: "Las iguanas escapan de sus depredadores principalmente corriendo a gran velocidad");
    }
    public void metododealimenta(){
        System.out.println(x: "Jony le da las sobras de carne que quedaron de los demas animales a null");
    }
}
```

Al igual que su clase hermana Caimán, la clase Iguana también hereda el constructor modificado de la clase Reptiles, agregando sus propios atributos privados: raza de iguana y temperatura corporal. Los métodos añadidos fueron: metododeescape() y metododealimenta().

Clase proceso

```
public class proceso {
    Run | Debug
    public static void main(String[] args) {
        System.out.println(x: "hoy tenemos en zologico");
        animales animales= new animales(nombre: "null", nombre_cientifico: null, Numero_de_jaula: 0);
        animales.presentacion();
        animales.encargados();
        System.out.println(x: "primero vayamos con los mamiferos");
        mamiferos mamiferos= new mamiferos(nombre: "null", nombre_cientifico: null, numero_de_jaula: 0);
        mamiferos.ocultar_su_color();
        mamiferos.verificar_a_los_cachorros();
        leon Leo= new leon(nombre: "null", nombre_cientifico: "Panthera Leo", Numero_de_jaula: 1, peso: 200);
        Leo.presentacion2();
        Leo.metododealimenta();
        Leo.metododecaza();
        caballo caballo=new caballo(nombre: "null", nombre_cientifico: "Equus ferus przewalskii", Numero_de_jaula: 0);
        caballo.presentacion2();
        caballo.heno();
        caballo.metododeescape();

        System.out.println(x: "ahora pasemos con los reptiles");

        reptiles reptiles=new reptiles(nombre: null, nombre_cientifico: null, Numero_de_jaula: 0, peso: 0);
        reptiles.humedecer_las_escamas();
        reptiles.verificar_a_los_Huevos();

        caiman caiman= new caiman(nombre: "null", nombre_cientifico: "Caiman yacare", Numero_de_jaula: 0);
        caiman.presentacion2();
        caiman.metododealimenta();
```

Al igual que la clase Main del primer ejercicio, la clase Proceso se encarga de recopilar la información proveniente de todas las clases que heredan de la clase principal (Animales).

Conclusiones

En mi opinión, este ha sido uno de los mejores trabajos que he realizado a lo largo de estos dos últimos semestres, ya que, a diferencia de otros profesores, el profesor Cristian supo explicarse muy bien respecto a la temática, y su ritmo de explicación fue adecuado para todos los estudiantes. Por ello, no me resultó muy difícil completar la actividad.