UNIVERSIDAD DE LAS FUERZAS ARMADAS-ESPE

SEDE SANTO DOMINGO DE LOS TSÁCHILAS

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN - DCCO-SS

CARRERA DE INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN

PERIODO	:	202450 Mayo– Septiembre 2024
ASIGNATURA	:	Programación Orientada a Objetos
TEMA	:	Proyecto Grupal.
ESTUDIANTES	:	**Santos Fernanda, Manzanillas Klay**
NIVEL-PARALELO - NRC	:	Segundo A - Nrc 15279
DOCENTE	:	Ing. Verónica Martínez C., Mgs.
FECHA DE ENTREGA	:	10 de Junio del 2024.

SANTO DOMINGO – ECUADOR

Índice

1. Introducción

El presente informe documenta el desarrollo de un sistema de gestión para la Granja Grupo B, aplicando los conceptos y temas vistos en la unidad 1 de Programación Orientada a Objetos (POO). Este sistema está diseñado para facilitar la administración de empleados, animales y cultivos en la granja. Se ha desarrollado en Java (Eclipse), haciendo uso de diversas clases y métodos para gestionar eficientemente las actividades diarias.

El sistema permite el ingreso de datos sobre empleados, animales y cultivos, así como la capacidad de guardar esta información en archivos CSV y JSON para su posterior recuperación y análisis. Además, se han implementado funcionalidades adicionales, como la lectura de archivos CSV y JSON para cargar datos previamente guardados.

A lo largo de este informe, se detallarán las funcionalidades implementadas en cada una de las clases del sistema, aplicando los principios de la POO. Se presentarán ejemplos de su uso y los resultados obtenidos, destacando la aplicación práctica de los conceptos aprendidos en la unidad 1.

2. Objetivos

2.1. Objetivo General

- Realizar un programa integrando todos los temas dados en la Unidad 1.

2.2. Objetivos Específicos

- Realizar un programa con todos los temas del silabo 1.1 al 1.15.

3. Marco Teórico/ Desarrollo

Marco Teórico:

NO APLICA

Desarrollo:

Lenguaje Natural

El programa de Granja_GrupoB, contiene 6 tipos de clases diferentes las cuales son Granja_GrupoB que actúa como la clase Padre donde 2 son sub-hijas de la clase padre que son (Cultivos_GrupoB, Animales_GrupoB), las otras 3 clases son independiente (Empleados_GrupoB, MenuArchivo, Principal_GrupoB).

1. **Principal_GrupoB:** La clase Principal_GrupoB es esencial para que ejecute el programa. Contiene el método main, que es un método estático, se inicia la ejecución del programa. Aquí, se crean instancias de las clases Empleados_GrupoB y MenuArchivo, que representan las interacciones con los empleados y las operaciones relacionadas con archivos, respectivamente. Luego, se invocan métodos específicos de estas instancias para realizar acciones como la entrada de datos de los empleados y la gestión de archivos. En resumen, la clase Principal_GrupoB coordina las acciones principales que se llevan a cabo al iniciar el programa de la granja.
2. **Granja_GrupoB**: Nuestra clase padre es una clase abstracta donde importa la clase Scanner del paquete java.util para permitir la entrada de datos desde la consola al usuario

,seguido de eso encontraremos 3 atributos referenciados con metodos de acceso Protected (nombre, tipo, tipo de producción) contiene un constructor para inicializar los atributos, cuenta con 2 métodos el método ingresoDato(), solicita al usuario que ingrese los datos para mayor información. Además también incluye un método abstracto imprimirDat(), diseñado para ser implementado de manera específica por las clases hijas, lo que permite una salida de información, aplicando polimorfismo.

3. **Cultivos_GrupoB:** La clase Cultivo_GrupoB es una subclase de Granja_GrupoB y está diseñada para gestionar los cultivos en la granja. Contiene seis atributos en total: tres atributos heredados (nombre, tipo y tipo de producción) y tres atributos propios (referenciados- rendimiento, primitivos -área de cultivo).

   En cuanto a los métodos, cuenta con cinco propios: datCultivo, realizarAccionesCulti, guardarJSON, leerJSON e imprimirDat. Además, implementa el método abstracto imprimirDat de la clase padre, lo que demuestra el uso del polimorfismo.

   La clase utiliza entrada de datos a través de la clase Scanner para recopilar información sobre los cultivos y su rendimiento. También emplea la escritura y lectura de archivos JSON para guardar y recuperar datos de cultivos.

   Además, la clase contiene métodos para realizar acciones específicas relacionadas con los cultivos, como la impresión de datos, la realización de acciones de cultivo y la gestión de empleados relacionados con los cultivos.

4. **Animales_GrupoB:** La clase Animal_GrupoB utiliza la clase Scanner del paquete java.util para admitir la entrada de datos del usuario. Esta clase cuenta con 7 metodos también, tiene tres atributos referenciados: nombre, tipo y tipo de producción, que almacenan información sobre los animales. El constructor de la clase inicializa estos atributos al crear una nueva instancia. Posee métodos como ingresoDato(), que interactúa con el usuario solicitando datos sobre el animal, e imprimirDat(), para mostrar información específica del animal, aplicando polimorfismo. Además, la clase maneja datos primitivos como double e int para el peso y costo del animal, respectivamente. Utiliza una lista (ArrayList) llamada listaAnimales para almacenar los animales ingresados. La asociación entre la clase Empleados_GrupoB y Animal_GrupoB es de dependencia, ya que la primera instancia la segunda en su método datoEmpleado(). En términos de encapsulamiento, los atributos están protegidos con modificadores de acceso y se acceden a través de métodos públicos. La clase está en el paquete Granja_GrupoB. En cuanto a la abstracción, la clase contiene métodos abstractos, lo que la hace abstracta. Además, hereda atributos y métodos de la clase Granja_GrupoB, mostrando herencia. También cuenta con 2 métodos de lectura y escritura de archivos CSV. El método guardarCSV() se encarga de guardar los datos de los animales en un archivo CSV. Utiliza un BufferedWriter para escribir en el archivo, y guarda el nombre, tipo, tipo de producción, peso y costo de cada animal en una línea separada por comas. Además, utiliza una instancia de la clase DecimalFormat para formatear el peso del animal a dos decimales antes de escribirlo en el archivo.

Por otro lado, el método leerCSV() se utiliza para leer los datos de los animales desde un archivo CSV. Utiliza un BufferedReader para leer el archivo línea por línea. Luego, divide cada línea en sus componentes utilizando el método split() y los almacena en un array de strings. A partir de estos datos, crea nuevos objetos Animal_GrupoB y los agrega a la lista listaAnimales. Este método se encarga de cargar los datos previamente guardados en el archivo CSV y mostrarlos en la consola.

5. **Empleados_GrupoB:** La clase Empleados_GrupoB se encarga de gestionar los empleados de la granja. Contiene tres atributos referenciados: nombre, posición y tipo de servicio. Además, cuenta con un constructor para inicializar estos atributos.

   En cuanto a los métodos, la clase tiene 4: datoEmpleado, guardarCSV ,leerCSV y empleadoDat (). El método datoEmpleado solicita al usuario que ingrese los datos del empleado, incluyendo su nombre, posición y tipo de servicio. Dependiendo de la posición del empleado (cuidador o agrícola), se le da la opción de agregar animales o cultivos, respectivamente. También permite agregar nuevos servicios si la posición no está disponible. Además, este método utiliza el método guardarCSV para almacenar los datos de los empleados en un archivo CSV.

   El método guardarCSV se encarga de guardar los datos del empleado en un archivo CSV llamado "puestosNuevos.csv", mientras que el método leerCSV lee y muestra los datos guardados en este archivo CSV.

   Finalmente, el método empleadoDat imprime los datos del empleado, mostrando su nombre y posición.

6. **MenuArchivo:** La clase MenuArchivo se encarga de proporcionar un menú de opciones para realizar diferentes acciones relacionadas con la lectura de archivos y acciones específicas en la granja.Cuenta con un atributo privado opm para almacenar la opción aplicando encapsulamiento seleccionada por el usuario en el menú.El constructor de la clase inicia el atributo opm en 0.
   Los métodos principales de la clase son:
   - **archivos**: Este método muestra un menú para la lectura de archivos. El usuario puede elegir entre leer datos de animales desde un archivo CSV, leer datos de cultivos desde un archivo JSON o leer datos de puestos desde un archivo CSV. Dependiendo de la opción seleccionada, se crean instancias de las clases Animal_GrupoB, Cultivo_GrupoB o Empleados_GrupoB para llamar a los métodos de lectura correspondientes.
   - **menuAccionesAnimales:** Este método muestra un menú de acciones que se pueden realizar con los animales en la granja, como darles de comer o limpiar su recinto.
   - **accionesCultivos**: Este método muestra un menú de acciones que se pueden realizar con los cultivos en la granja, como regarlos o fertilizarlos. Ambos métodos menuAccionesAnimales y accionesCultivos utilizan un bucle do-while para permitir al usuario seleccionar una opción y continuar realizando acciones hasta que decida salir

del menú.La clase utiliza la clase Scanner para la entrada de datos desde la consola y maneja las excepciones para garantizar que se ingresen opciones válidas.

Clase Principal (Main**)**

```java
package Granja_GrupoB;

public abstract class Principal_GrupoB {
    //******* METODOS STATIC ********
    public static void main(String[] args) {
        Empleados_GrupoB empleados = new Empleados_GrupoB("", "", "");
        empleados.datoEmpleado(empleados);
        MenuArchivo archi = new MenuArchivo (0);
        archi.archivos();

    }
}

```

Clase Padre (Granja_GrupoB**)**

```java
package Granja_GrupoB;
import java.util.Scanner;

public abstract class Granja_GrupoB {
    protected Scanner scanner = new Scanner (System.in);
    //ATRIBUTOS- PREFERENCIADOS
    protected String name, type, productionClass;

    public Granja_GrupoB (String name, String type, String productionClass) {
        //CONSTRUCTOR
        this.name = name;
        this.type = type;
        this.productionClass = productionClass;
    }
    //INGRESAMOS DATOS
    public void  ingresoDato () {
        System.out.println("Ingrese el nombre: ");
        name = scanner.nextLine();
        System.out.println("Ingrese el tipo : ");
        type = scanner.nextLine();
        System.out.println("Ingrese el tipo de produccion: ");
        productionClass = scanner.nextLine();
    }
    //**************POLIMORFISMO_IMPRIMIR DATOS**************
    public abstract void imprimirDat ();
}

```

Clase hija (Cultivo_GrupoB**)**

```java
package Granja_GrupoB;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;

public class Cultivo_GrupoB  extends Granja_GrupoB {
    protected Scanner scanner = new Scanner(System.in);
    //******** ATRBUTOS PROPIOS ********
    //REFERENCIADOS
    protected String  performance;
    //PRIMITIVOS
    protected int cultivationArea;
    //protected int cantCultivos;
   //*************** CONSTRUCTOR **************
    public Cultivo_GrupoB (String name, String type, String productionClass,String performance, int cultivationArea ){
    super (name, type, productionClass); //**** ATRIBUTOS HEREDADOS ****
    this.cultivationArea = cultivationArea;
    this.performance = performance;
    }
    //************* METODOS ************
    public void datCultivo (Empleados_GrupoB empleado) {
        ingresoDato ();
        do {
        System.out.println("Ingrese el area de cultivo [1-25] en metros: ");
        cultivationArea = scanner.nextInt();
        }while(cultivationArea<1 || cultivationArea>25) ;
        scanner.nextLine();
        System.out.println("Ingrese el rendimiento del cultivo: ");
        performance = scanner.nextLine();
        System.out.println("°°° DATOS IMPRESOS CULTIVOS °°°");
        empleado.empleadoDat();
        imprimirDat();
        System.out.println("AREA CULTIVO: " + cultivationArea);
        System.out.println("RENDIMIENTO CULTIVO: " + performance);
        System.out.println("-----------------------------------");
        realizarAccionesCulti ();

}
    //***************POLIMORFISMO**************
    @Override
    public void  imprimirDat() {
        System.out.println("NOMBRE : " + name);
        System.out.println("TIPO: " + type);
        System.out.println("PRODUCCION: " + productionClass);
    }
    public void realizarAccionesCulti () {
        MenuArchivo dato = new MenuArchivo (0);
        dato.accionesCultivos();
        guardarJSON ();
        String answer;
        do {
            System.out.println("¿Desea agregar otro puesto? (Si/No)");
            answer = scanner.next();

            if (answer.equalsIgnoreCase("Si")) {
                Empleados_GrupoB empleado = new Empleados_GrupoB("", "", "");
                empleado.datoEmpleado(empleado);
            } else if (answer.equalsIgnoreCase("No")) {
                return;
            } else {
                System.out.println("Si/No");
```

```
            }
        } while (true);

    }
// *************GUARDAR ARCHIVOS JSON ****************
    public void guardarJSON () {
        String filePath = "cultivo.json";
        JSONObject cultivoJSON = new JSONObject ();
        cultivoJSON.put("NOMBRE CULTIVO" , name);
        cultivoJSON.put("TIPO CULTIVO" , type);
        cultivoJSON.put("CLASE PRODUCCION CULTIVO" , productionClass);
        cultivoJSON.put("AREA CULTIVO" , cultivationArea);
        cultivoJSON.put("RENDIMIENTO CULTIVO" , performance);
        try (FileWriter file = new FileWriter (filePath)){
            file.write(cultivoJSON.toJSONString ());
            file.write("\n");
            System.out.println(" ### DATOS GUARDADOS EN JSON ###");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
   // *************LECTURA ARCHIVOS JSON ****************
    public void leerJSON() {
        String filePath = "cultivo.json";
        JSONParser parser = new JSONParser();
        try {
            JSONObject jsonCultivo = (JSONObject) parser.parse(new FileReader(filePath));
            System.out.println("Nombre del cultivo: " + jsonCultivo.get("NOMBRE CULTIVO"));
            System.out.println("Tipo de cultivo: " + jsonCultivo.get("TIPO CULTIVO"));
            System.out.println("Clase de producción del cultivo: " + jsonCultivo.get("CLASE PRODUCCION CULTIVO"));
            System.out.println("Area del cultivo: " + jsonCultivo.get("AREA CULTIVO"));
            System.out.println("Rendimiento del cultivo: " + jsonCultivo.get("RENDIMIENTO CULTIVO"));
            System.out.println("DATOS DESDE ARCHIVO JSON");
        } catch (Exception e) {
            e.printStackTrace();
            }
        }
    }
```

Clase hija (Animal_GrupoB**)**

```
package Granja_GrupoB;

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Scanner;
import java.text.DecimalFormat; //NUM A 2 DECIMALES

public   class Animal_GrupoB extends Granja_GrupoB {
    protected Scanner scanner = new Scanner(System.in);
    protected ArrayList<Animal_GrupoB> listaAnimales = new ArrayList<>(); //LISTA ANIMALES
    //PRIMITIVOS
    protected double weight;
    protected int cost, cant;

    public Animal_GrupoB(String name, String type, String productionClass, double weight, int cost, int cant) {
        super(name, type, productionClass);
        this.weight = weight;
        this.cost = cost;
        this.cant = cant;
    }

    public void agregarAnimal() {
        Animal_GrupoB nuevoAnimal = new Animal_GrupoB("", "", "", 0, 0, 0);//INSTANCIA NUEVA PARA ANIMAL
        ingresoDato(nuevoAnimal);
        do {
            System.out.println("Ingrese el peso del animal [8-100]: ");
            nuevoAnimal.weight = scanner.nextDouble();
        } while (nuevoAnimal.weight < 8 || nuevoAnimal.weight > 100);
        do {
            System.out.println("Ingrese el precio del animal [5-500]: ");
```

```
            nuevoAnimal.cost = scanner.nextInt();
        } while (nuevoAnimal.cost < 5 || nuevoAnimal.cost > 500);
        listaAnimales.add(nuevoAnimal);
        guardarCSV(nuevoAnimal); //GUARDAR EN CSV LOS NUEVOS ANIMALES
        System.out.println("### DATOS DEL ANIMAL GUARDADOS CORRECTAMENTE EN LA LISTA ###");
    }

    public void imprimirListaAnimales() {
        if (listaAnimales.isEmpty()) {
            System.out.println("La lista de animales esta vacia.");
        } else {
            System.out.println("----Lista de Animales----");
            for (Animal_GrupoB animal : listaAnimales) {
                System.out.println("Nombre: " + animal.name);
                System.out.println("Tipo: " + animal.type);
                System.out.println("Clase de producción: " + animal.productionClass);
                System.out.println("Peso: " + animal.weight);
                System.out.println("Costo: " + animal.cost);
                System.out.println("---------------------------------");
            }
            System.out.println("### LISTA DE ANIMALES IMPRESA CORRECTAMENTE ###");
        }
    }

    public void ingresoAni() {
        do {
            System.out.print("Ingrese cuantos animales desea ingresar [1-10] ?: ");
            cant = scanner.nextInt();
        } while (cant < 1 || cant > 10);

        for (int i = 0; i < cant; i++) {
            System.out.println("Animal #" + (i + 1) + ":");
            agregarAnimal();
        }
```

```
        imprimirListaAnimales();
        String answer;
        do {
            System.out.println("¿Desea agregar otro puesto? (Si/No)");
            answer = scanner.next();

            if (answer.equalsIgnoreCase("Si")) {
                Empleados_GrupoB empleado = new Empleados_GrupoB("", "", "");
                empleado.datoEmpleado(empleado);
            } else if (answer.equalsIgnoreCase("No")) {
                return;
            } else {
                System.out.println("Si/No");
            }
        } while (true);
    }

    public void ingresoDato(Animal_GrupoB animal) {
        scanner.nextLine();
        System.out.print("Ingrese el nombre del animal: ");
        animal.name = scanner.nextLine();
        System.out.print("Ingrese el tipo de animal: ");
        animal.type = scanner.nextLine();
        System.out.print("Ingrese la clase de producción: ");
        animal.productionClass = scanner.nextLine();
    }

    public void guardarCSV(Animal_GrupoB animal) {
        String filePath = "animal.csv";
        DecimalFormat df = new DecimalFormat("#.##");
        try (BufferedWriter writer = new BufferedWriter(new FileWriter(filePath, true))) {
            writer.write(animal.name + "," + animal.type + "," + animal.productionClass + "," + df.format(animal.weight) + "," + animal.cost);
            writer.newLine();
            System.out.println("### DATOS GUARDADOS CORRECTAMENTE EN CSV ###");
```

```
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

```
    public void leerCSV() {
        String filePath = "animal.csv";
        listaAnimales.clear();
        try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
            String line;
            while ((line = reader.readLine()) != null) {
                String[] data = line.split(",");
                if (data.length == 5) {
                    String name = data[0];
                    String type = data[1];
                    String productionClass = data[2];
                    double weight = Double.parseDouble(data[3]);
                    int cost = Integer.parseInt(data[4]);
                    Animal_GrupoB animal = new Animal_GrupoB(name, type, productionClass, weight, cost, 0);
                    listaAnimales.add(animal);
                }
            }
            System.out.println("### DATOS CARGADOS CORRECTAMENTE DESDE EL CSV ###");
            imprimirListaAnimales();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @Override
    public void imprimirDat() {
        System.out.println("NOMBRE : " + name);
        System.out.println("TIPO: " + type);
        System.out.println("PRODUCCION: " + productionClass);
    }
}
```

Empleados_GrupoB

```
package Granja_GrupoB;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;

public class Empleados_GrupoB {
    protected Scanner scanner = new Scanner (System.in);
    //****REFERENCIADOS ***
    protected String name, position, serviceType;

    public Empleados_GrupoB(String name, String position, String serviceType) {
        this.name = name;
        this.position = position;
        this.serviceType = serviceType;
    }

    public void datoEmpleado(Empleados_GrupoB empleado) {
            System.out.println("°°°° BIENVENID@S A LA GRANJITA GRUPO B °°°°");
            System.out.println("--    INGRESE LOS DATOS DEL EMPLEADO --");
            System.out.println("Ingrese su nombre: ");
            name = scanner.nextLine();
            boolean puestoValido = false;

            while (!puestoValido) {
                System.out.println("Ingrese su puesto (cuidador/agricola): ");
                position = scanner.nextLine();

                if (position.equalsIgnoreCase("cuidador")) {
                    String answer;
                    do {
                        System.out.println("¿Desea agregar un animal? (Si/No)");
                        answer = scanner.nextLine();
                    } while (!answer.equalsIgnoreCase("Si") && !answer.equalsIgnoreCase("No"));

                    if (answer.equalsIgnoreCase("Si")) {
                        Animal_GrupoB animal = new Animal_GrupoB("", "", "", 0, 0,0);
                        animal.ingresoAni();
                    }
                    puestoValido = true;
                } else if (position.equalsIgnoreCase("agricola")) {
                    String answer;
```

```
                    do {
                        System.out.println("¿Desea agregar un cultivo? (Si/No)");
                        answer = scanner.nextLine();
                    } while (!answer.equalsIgnoreCase("Si") && !answer.equalsIgnoreCase("No"));

                    if (answer.equalsIgnoreCase("Si")) {
                        Cultivo_GrupoB cultivo = new Cultivo_GrupoB("", "", "", "", 0);
                        cultivo.datCultivo(empleado);
                    }
                    puestoValido = true;
                } else {
                    System.out.println("Lo sentimos, no hay ese puesto en nuestra granja.");
                    String answer;
                    do {
                        System.out.println("¿Desea agregar sus servicios en esta granja? (Si/No)");
                        answer = scanner.nextLine();
                    } while (!answer.equalsIgnoreCase("Si") && !answer.equalsIgnoreCase("No"));
                    if (answer.equalsIgnoreCase("Si")) {
                        System.out.println("Puesto que trata de ofrecer :");
                        position = scanner.nextLine();
                        System.out.println("Indique de que trata su servicio: ");
                         serviceType = scanner.nextLine();
                        System.out.println("Servicio aceptado: " + position);
                        System.out.println("Descripcion: " + serviceType);
                        puestoValido = true;
                        guardarCSV();
                    } else {
                        break;
                    }
                }
            }
        }

    //GUARDAR CSV
    public void guardarCSV() {
        String filePath = "puestosNuevos.csv";
        try (BufferedWriter writer = new BufferedWriter(new FileWriter(filePath, true))) {
            writer.write(name + "," + position + "," + serviceType );
            writer.newLine();
            System.out.println("### DATOS GUARDADOS CORRECTAMENTE EN CSV ###");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void leerCSV () {
    String filePath = "puestosNuevos.csv";
    try (BufferedReader reader = new BufferedReader (new FileReader (filePath))){
        System.out.println("Datos cargados desde csv");
        String line;
        while (((line=reader.readLine()) !=null)){
            System.out.println(line);
        }
        }catch (IOException e ) {
        e.printStackTrace();
        }
    }
    public void empleadoDat () {
        System.out.println("NOMBRE EMPLEADO: " + name);
        System.out.println("PUESTO: " + position);
    }
}
```

MenuArchivo

```java
package Granja_GrupoB;
import java.util.Scanner;

public class MenuArchivo {
    private Scanner scanner = new Scanner(System.in);
    //*********ENCAPSULAMIENTO ************
    private int opm;

    public MenuArchivo(int i) {
        //***INICIAR DESDE 0
        this.opm = 0;
    }
    public void archivos() {
        do {
            System.out.println("--  LECTURA DE ARCHIVOS  --");
            System.out.println("[1]. LECTURA ANIMALES CSV.");
            System.out.println("[2]. LECTURA CULTIVOS JSON.");
            System.out.println("[3]. LECTURA PUESTOS CSV.");
            System.out.println("[4]. Salir");
            try {
                opm = Integer.parseInt(scanner.nextLine());
                switch (opm) {
                    case 1:
                        Animal_GrupoB animal = new Animal_GrupoB("", "", "", 0.0, 0,0);
                        animal.leerCSV();
                        break;
                    case 2:
                        Cultivo_GrupoB json = new Cultivo_GrupoB("", "", "", "", 0);
                        json.leerJSON();
                        break;
                    case 3:
                        Empleados_GrupoB emple = new Empleados_GrupoB ("","","");
                        emple.leerCSV();
                    break;
                case 4:
                    System.out.println("SALIO EXITOSAMENTE DE NUESTRO SERVICIO DE ARCHIVOS GRANJITA_B");
                    break;
                default:
                    System.out.println("Opcion incorrecta");
                    break;
                }
                //MANEJO DE EXCEPCIONES
            } catch (NumberFormatException e) {
                System.out.println("Ingrese un numero valido...");
            }
        } while (opm != 4);
    }

    public void menuAccionesAnimales() {
        do {
            System.out.println("¿Que accion desea realizar?");
            System.out.println("[1]. Dar de comer al animal.");
            System.out.println("[2]. Limpiar el recinto del animal.");
            System.out.println("[3]. Salir");
            //**********MANEJO ENTRADAS INCORRECTAS **********
            try {
                opm = Integer.parseInt(scanner.nextLine());
                switch (opm) {
                    case 1:
                        System.out.println("Dando de comer al animal ...");
                        break;
                    case 2:
                        System.out.println("Limpiando el recinto del animal...");
                        break;
                    case 3:
                        System.out.println("Gracias por su trabajo");
                            break;
                       default:
                            System.out.println("Opcion incorrecta");
                            break;
                    }
                    //**********MANEJO DE EXCEPCIONES********
                } catch (NumberFormatException e) {
                    System.out.println("Ingrese un numero valido...");
                }
            } while (opm != 3);
        }
```

```
78
79      public void accionesCultivos () {
80          do {
81              System.out.println("¿Que accion desea realizar?");
82              System.out.println("[1]. Riego del cultivo");
83              System.out.println("[2]. Fertilizacion del cultivo");
84              System.out.println("[3]. Salir");
85              System.out.print("Ingrese una opcion: ");
86              try {
87                  opm = Integer.parseInt(scanner.nextLine());
88                  switch (opm) {
89                  case 1:
90                        System.out.println("Realizando riego del cultivo...");
91                      break;
92                  case 2:
93                      System.out.println("Realizando fertilizacion del cultivo...");
94                      break;
95                  case 3:
96                       System.out.println("Gracias por su trabajo");
97                      break;
98                  default:
99                      System.out.println("Opcion invalida. Intente de nuevo.");
100                     break;
101                 }
102                 //**********MANEJO DE EXCEPCIONES********
103             } catch (NumberFormatException e) {
104                 System.out.println("Ingrese un numero valido...");
105             }
106         } while (opm != 3);
107     }
108 }
109
```

Ejecuciones

```
°°°° BIENVENID@S A LA GRANJITA GRUPO B °°°°
--    INGRESE LOS DATOS DEL EMPLEADO --
Ingrese su nombre:
Ainoha Fernanda
Ingrese su puesto (cuidador/agricola):
cuiDadOr
¿Desea agregar un animal? (Si/No)
a
¿Desea agregar un animal? (Si/No)
sI
Ingrese cuantos animales desea ingresar [1-10] ?: -1
Ingrese cuantos animales desea ingresar [1-10] ?: 2
Animal #1:
Ingrese el nombre del animal: Vaca
Ingrese el tipo de animal: Vaca
Ingrese la clase de producción: Leche
Ingrese el peso del animal [8-100]:
-2
Ingrese el peso del animal [8-100]:
230.34
Ingrese el peso del animal [8-100]:
89.90
Ingrese el precio del animal [5-500]:
-3
Ingrese el precio del animal [5-500]:
300
### DATOS GUARDADOS CORRECTAMENTE EN CSV ###
### DATOS DEL ANIMAL GUARDADOS CORRECTAMENTE EN LA LISTA ###
```

```
¿Desea agregar otro puesto? (Si/No)
a
Si/No
¿Desea agregar otro puesto? (Si/No)
si
°°°° BIENVENID@S A LA GRANJITA GRUPO B °°°°
--    INGRESE LOS DATOS DEL EMPLEADO --
Ingrese su nombre:
juana
Ingrese su puesto (cuidador/agricola):
agricola
¿Desea agregar un cultivo? (Si/No)
sI
Ingrese el nombre:
Ruda
Ingrese el tipo :
Rudita
Ingrese el tipo de produccion:
aguas
Ingrese el area de cultivo [1-25] en metros:
-3
Ingrese el area de cultivo [1-25] en metros:
12
Ingrese el rendimiento del cultivo:
Bueno
```

```
Animal #2:
Ingrese el nombre del animal: Gallina
Ingrese el tipo de animal: Gallina
Ingrese la clase de producción: Huevos
Ingrese el peso del animal [8-100]:
-34
Ingrese el peso del animal [8-100]:
12.50
Ingrese el precio del animal [5-500]:
-3
Ingrese el precio del animal [5-500]:
5
### DATOS GUARDADOS CORRECTAMENTE EN CSV ###
### DATOS DEL ANIMAL GUARDADOS CORRECTAMENTE EN LA LISTA ###
----Lista de Animales----
Nombre: Vaca
Tipo: Vaca
Clase de producción: Leche
Peso: 89.9
Costo: 300
----------------------------------
Nombre: Gallina
Tipo: Gallina
Clase de producción: Huevos
Peso: 12.5
Costo: 5
----------------------------------
### LISTA DE ANIMALES IMPRESA CORRECTAMENTE ###
```

```
°°° DATOS IMPRESOS CULTIVOS °°°
NOMBRE EMPLEADO: juana
PUESTO: agricola
NOMBRE : Ruda
TIPO: Rudita
PRODUCCION: aguas
AREA CULTIVO: 12
RENDIMIENTO CULTIVO: Bueno
----------------------------------
¿Que accion desea realizar?
[1]. Riego del cultivo
[2]. Fertilizacion del cultivo
[3]. Salir
Ingrese una opcion: 4
Opcion invalida. Intente de nuevo.
¿Que accion desea realizar?
[1]. Riego del cultivo
[2]. Fertilizacion del cultivo
[3]. Salir
Ingrese una opcion: 1
Realizando riego del cultivo...
¿Que accion desea realizar?
[1]. Riego del cultivo
[2]. Fertilizacion del cultivo
[3]. Salir
Ingrese una opcion: 2
Realizando fertilizacion del cultivo...
¿Que accion desea realizar?
[1]. Riego del cultivo
[2]. Fertilizacion del cultivo
[3]. Salir
Ingrese una opcion: 3
Gracias por su trabajo
```

```
¿Desea agregar otro puesto? (Si/No)
NO
--  LECTURA DE ARCHIVOS  --
[1]. LECTURA ANIMALES CSV.
[2]. LECTURA CULTIVOS JSON.
[3]. LECTURA PUESTOS CSV.
[4]. Salir
5
Opcion incorrecta
--  LECTURA DE ARCHIVOS  --
[1]. LECTURA ANIMALES CSV.
[2]. LECTURA CULTIVOS JSON.
[3]. LECTURA PUESTOS CSV.
[4]. Salir
1
### DATOS CARGADOS CORRECTAMENTE DESDE EL CSV ###
----Lista de Animales----
Nombre: d
Tipo: f
Clase de producción: f
Peso: 56.0
Costo: 45
----------------------------------
Nombre: fd
Tipo: f
Clase de producción: d
Peso: 34.0
Costo: 34
----------------------------------
Nombre: fe
Tipo: df
Clase de producción: s
Peso: 23.0
```

```
--  LECTURA DE ARCHIVOS  --
[1]. LECTURA ANIMALES CSV.
[2]. LECTURA CULTIVOS JSON.
[3]. LECTURA PUESTOS CSV.
[4]. Salir
3
Datos cargados desde csv
fernanda,sf,sf
si,s,d
--  LECTURA DE ARCHIVOS  --
[1]. LECTURA ANIMALES CSV.
[2]. LECTURA CULTIVOS JSON.
[3]. LECTURA PUESTOS CSV.
[4]. Salir
4
SALIO EXITOSAMENTE DE NUESTRO SERVICIO DE ARCHIVOS GRANJITA_B
```

```
----------------------------------
Nombre: Vaca
Tipo: Vaca
Clase de producción: Leche
Peso: 89.9
Costo: 300
----------------------------------
Nombre: Gallina
Tipo: Gallina
Clase de producción: Huevos
Peso: 12.5
Costo: 5
----------------------------------
### LISTA DE ANIMALES IMPRESA CORRECTAMENTE ###
--  LECTURA DE ARCHIVOS  --
[1]. LECTURA ANIMALES CSV.
[2]. LECTURA CULTIVOS JSON.
[3]. LECTURA PUESTOS CSV.
[4]. Salir
2
Nombre del cultivo: Ruda
Tipo de cultivo: Rudita
Clase de producción del cultivo: aguas
Area del cultivo: 12
Rendimiento del cultivo: Bueno
DATOS DESDE ARCHIVO JSON
```

4. Conclusiones

- La entrada y salida de datos por consola son aspectos críticos para el funcionamiento correcto de una aplicación.
- Al encapsular los atributos y proporcionar métodos específicos para acceder y modificar estos datos, se promueve una mejor organización del código y se evita la exposición innecesaria de la implementación interna de la clase.
- La utilización de métodos en la POO permite modularizar el código y promover la reutilización. Al dividir la funcionalidad en pequeñas unidades lógicas, se facilita la comprensión del sistema y se promueve la claridad y la simplicidad del código.

5. Recomendaciones

- Comentar nuestro código, la documentación clara y completa es esencial para comprender y mantener el código de manera efectiva. Incluir comentarios detallados, explicaciones sobre el funcionamiento de los métodos y clases, así como ejemplos de uso, ayuda a los usuario/programadores a trabajar de manera más eficiente y a reducir el tiempo dedicado a la resolución de problemas.
- Manejar excepciones de manera adecuada es fundamental para garantizar la estabilidad y seguridad del programa.
- Al crear métodos debemos mantenerlos enfocados en realizar una tarea específica y bien definida. Evitar la creación de métodos demasiado largos o que realicen múltiples acciones diferentes.

6. Bibliografía/ Referencia

NO APLICA

7. Anexos:

NO APLICA

8. Legalización de documento

Nombres y Apellidos: Ainoha Fernanda Santos Nugra

CI: 2300304637

Firma:

Nombres y Apellidos: Klay Shande Manzanillas Delgado

CI: 2400070906

Firma: