Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Introducción a la Programación y Computación 1 Sección C

Catedrático: Ing. Moisés Velásquez

Tutor académico: Carlos Campanero, William Corado

Práctica no. 2

Manual Técnico

Nombre: Diego Enrique Arriaga Meléndez

Canet: 202003892

Índice

Especificaciones del Software

Hp Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz 1.50 GHz, RAM de 8.00 GB

La versión jdk de Java que se utilizo es **jdk1.8.0_111**.

La versión de Neatbeans IDE que se utilizo es **8.2**.

El programa se diseño para funcionar en una aplicación **.jar**.

Clase Principal

Esta clase sirve como puente entre todas las funciones del programa. Lo primero que realiza el programa una vez se inicia es imprimir un menú que muestra todas las acciones que puede realizar el programa, este menú se despliega de manera cíclica hasta que el usuario indique lo contrario. Una vez el usuario haya ingresado en la consola un número que indique que opción quiere realizar la clase mandara a llamar la clase que realice la opción que quiere el usuario, dentro de estas opciones se encuentra la de concluir con el programa, lo que provoca que el ciclo que genera el menú se cierre (Todos los submenús del programa funcionan de la misma manera).

```
public static void main(String[] args) {
//Se mandan a llamar todos los metodos que realizan todas las actividades del porgrama.
    Guardar guardar=new Guardar();
    CargarArchivos carga=new CargarArchivos();
    CargaPokemon opcion1=new CargaPokemon();
    CargaEntrenador opcion2=new CargaEntrenador();
    CargaPokeballs opcion3=new CargaPokeballs();
    CargaGimnasios opcion4=new CargaGimnasios();
    CargaAlimentos opcion5=new CargaAlimentos();
    AsignarPokemons opcion6=new AsignarPokemons();
    AsignarPokeball opcion7=new  AsignarPokeball();
    AsignarAlimentos opcion8=new AsignarAlimentos();
    AsignarPelea opcion9=new AsignarPelea();
    MenuReportes opcion10=new MenuReportes();
//Se crean variables que se encargaran de reconocer la información que ingrese el usuario
    Scanner sc=new Scanner(System.in);
    String op;
```

```
    //Opcion de Cargar archivos previos
    System.out.print("Desea Cargar archivos anteriores?(S/N):");
    //Elección del usuario
        op=sc.nextLine();
        if (op.equalsIgnoreCase("S")) {
            carga.carga();
        }
```

```
//Inicio del ciclo del menú principal.
int cont=1;
while (cont>0){
    //Se imprime el menú
    System.out.print("=========Menu de Inicio==========\n"
                    + "|1.Carga de Pokemons\t\t|\n"
                    + "|2.Carga de Entrenadores\t|\n"
                    + "|3.Carga de Poke balls\t\t|\n"
                    + "|4.Carga de Gimnasios\t\t|\n"
                    + "|5.Carga de Alimentos\t\t|\n"
                    + "|6.Asignar Pokemons\t\t|\n"
                    + "|7.Asignar Pokeballs\t\t|\n"
                    + "|8.Asignar Actividad de Comida\t|\n"
                    + "|9.Asignar Actividad de Pelea\t|\n"
                    + "|10.Reportes\t\t\t|\n"
                    + "|11.Guardar Archivos\t\t|\n"
                    + "|12.Salir del Programa\t\t|\n"
                    + "================================\n");
    //Instrucción al usuario
    System.out.print("Ingrese el numero de opción que quiere realizar: ");
    op=sc.nextLine();
    //Se deja un espacio considerable entre el menú y la acción seleccionada.
    System.out.println("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
    //Se manda a llamar el metodo seleccionado por el usuario
    switch(op){
        case "1":
            opcion1.menu();
            break;
        case "2":
            opcion2.menu();
            break;
        case "3":
            opcion3.menu();
            break;
        case "4":
            opcion4.menu();
            break;
        case "5":
            opcion5.menu();
            break;
        case "6":
            opcion6.menu();
            break;
        case "7":
            opcion7.menu();
            break;
        case "8":
            opcion8.menu();
            break;
        case "9":
            opcion9.menu();
            break;
        case "10":
            opcion10.menuReportes();
            break;
        case "11":
            guardar.guardar();
            break;
        case "12":
            //Se termina el ciclo que genera el menu.
            cont--;
            break;
            //Mensaje en caso de que el usuario no haya ingresado un valor adecuado.
        default:
            System.err.println("Asegurese que ingreso un número valido");
            break;
    }
}
```

Si el usuario ingresa el número 10 se manda a llamar un metodo de la misma clase que imprime un submenú, este submenú es para la generación de reportes, este metodo se llama **menuReportes()**, este mismo metodo se encarga de mandar a llamar las clases que generan los reportes HTML.

```
public class MenuReportes {
    static    ReporteEntrenadores opcion10_1=new ReporteEntrenadores();
    static    ReporteSalvajes opcion10_2=new ReporteSalvajes();
    static    ReporteComidas opcion10_3=new ReporteComidas();
    static    ReportesPeleas opcion10_4=new ReportesPeleas();
    static    TopAtaque opcion10_5=new TopAtaque();
    static    TopSalud opcion10_6=new TopSalud();
    //Se crean variables que se encargaran de reconocer la información que ingrese el usuario
    static    Scanner sc=new Scanner(System.in);
    static    String op;
    /**Se genera el menú con las opciones de reportes*/
```

```
/**Se genera el menú con las opciones de reportes*/
static void menuReportes(){
    //Inicio del ciclo del menú principal.
    int cont=1;
    while (cont>0){
        //Se imprime el menú
        System.out.print("========Menu de Reportes=========\n"
                        + "|1.Reporte de entrenadores\t|\n"
                        + "|2.Reporte de pokemons salvajes\t|\n"
                        + "|3.Reporte de Comidas\t\t|\n"
                        + "|4.Reporte de Peleas\t\t|\n"
                        + "|5.Top 5 de Puntos de Ataque\t|\n"
                        + "|6.Top 5 de Salud\t\t|\n"
                        + "|7.Salir al menu principal\t|\n"
                        + "================================\n");
        //Instrucción al usuario
        System.out.print("Ingrese el numero de opción que quiere realizar: ");
        op=sc.nextLine();
        //Se deja un espacio considerable entre el menú y la acción seleccionada.
        System.out.println("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
        //Se manda a llamar el metodo seleccionado por el usuario
        switch(op){
            case "1":
                opcion10_1.reporte();
                System.out.println("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
                break;
            case "2":
                opcion10_2.reportes();
                System.out.println("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
                break;
                case "3":
                    opcion10_3.reporte();
                    System.out.println("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
                    break;
                case "4":
                    opcion10_4.reportes();
                    System.out.println("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
                    break;
                case "5":
                    opcion10_5.reporte();
                    System.out.println("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
                    break;
                case "6":
                    opcion10_6.reporte();
                    System.out.println("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
                    break;
                case "7":
                    //Se termina el ciclo que genera el menu.
                    cont--;
                    break;
                default:
                    System.err.println("Ingrese un número apropiado");
                    break;
            }
        }
    }
}
```

Clase CargaPokemon

Esta clase posee los metodos necesarios para leer documentos csv que poseean un listado de los pokemons con los que se interectuara mientras el programa se mantenga encendido.

Primero se crean los arreglos que se encargaran de almacenar los datos de todas las columnas de la tabla del documento csv que ingrese el usuario. Y otras variables que serviran como contador.

```
public class CargaPokemon {
    //Se crean los arreglos que almacenaran los datos de la tabla.
    static int Id[]=new int[150],ingresados=0;
    static double Vida[]=new double[150], ptAt[]=new double[150];
    //Las variables que no son arreglos se encargan de almacenar lo que escribe el usuario.
    static String Titulo[]=new String[7],Tipo[]=new String[150], Nombre[]=new String[150],op,ruta;
    static boolean Capturado[]=new boolean[151], Estado[]=new boolean[151];
    //Variable para leer lo que escribe el usuario.
    static Scanner sc=new Scanner(System.in);
```

El primer metodo que la clase inicializa es **menu()**, este metodo existe en todos los metodos de *carga*. Este metodo se encarga de imprimir un submenú que muestra las dos acciones que puede realizar la clase que son leer un documento csv con el lista de pokemons y el regresar al menú principal.

```
static void menu(){
    try {
        //Inicio del ciclo del menú principal.
        int cont=1;
        while (cont>0){
            //Se imprime el menú
            System.out.println("================================\n"
                    + "|1.Carga Listado de Pokemons\t|\n"
                    + "|2.Salir al Menu Principal\t|\n"
                    + "================================");
            //Instrucción al usuario
            System.out.print("Ingrese el numero de opción que quiere realizar: ");
            /**Se almacena lo que escriba el usuario*/
            op=sc.nextLine();
            System.out.println();

            switch(op){
                case"1":
                    //Instrucción al usuario
                    System.out.print("Ingrese la ruta del archivo que poseea la lista de Pokemons: ");
                    System.out.println("Ejemplo: C:\\Users\\Diego\\Desktop\\ListadoPokemons.csv");
                    //Se almacena la ruta que ingreso el usuario.
                    ruta=sc.nextLine();
                    //Se deja un espacio considerable entre el menú y la acción seleccionada.
                    System.out.println("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
                    //Se manda a llamar el metodo seleccionado por el usuario
                    carga(ruta);
                    break;
```

```
                case"2":
                    //Se termina el ciclo que genera el menu.
                    cont--;
                    break;
                default:
                //Mensaje en caso de que el usuario no haya ingresado un valor adecuado.
                    System.err.println("Asegurese de ingresar un numero valido");
            }
        }
    } catch (Exception e) {
        //Mensaje en caso de que el usuario no haya ingresado un valor adecuado.
        System.err.println("Error al cargar el archivo. Asegurese a que ingreso la\n"
                + "ruta correcta o que el documento no tenga mas de 150\n"
                + "entrenadores.");
    }
}
```

Si el usuario ingresa el número 2 se iniciara el metodo **carga(String r)**, este metodo se encarga de leer el documento y almacenar los datos del documento tanto en el arreglo que se encarga de almacenar los títulos y los demas arreglos que almacenan los datos de los pokemons. Estos arreglos se almacenan de manera paralela, por lo que la misma posición de diferentes arreglos pertencen al mismo pokemon.

```
/**Metodo que se encarga de almacenar la información que ingreso el usuario en los
   arreglos creados anteriormente*/
static void carga(String r){
    /**Se reinician los arreglos, para que se pudieran llenar de nuevo*/
    Id=new int[150];Vida=new double[150];ptAt=new double[150];
    Titulo=new String[7];Tipo=new String[150]; Nombre=new String[150];
    Capturado=new boolean[151];Estado=new boolean[151];
    /**Se reinicia la variable que indica la cantidad de pokemons que ingreso el
     usuario*/
    ingresados=0;

    try {
        /**Se guarda el archivo que indico el usario y se le prepara para que
         pueda leerse*/
        File archivo=new File(r);
        FileReader archivoG=new FileReader(archivo);
        BufferedReader lectura=new BufferedReader(archivoG);
        /**Se lee la primera linea del documento*/
        String codigo=lectura.readLine(),
                /**Se crea un arreglo que divira los apartados en cada linea.*/
                divisor[]=new String[7];
        /**Se almacenan los titulos de cada columna*/
        Titulo=codigo.split("\\s*,\\s*");
```

```
            /**Se comienza un ciclo que se repetira hasta que el documento se
             encuentre completamente vacío*/
            int a=0;
            while ((codigo=lectura.readLine())!=null) {
                /**Se divira la línea en diferentes apartados*/
                divisor=codigo.split("\\s*,\\s*");
                /**Se asigna el valor del arreglo divisor en el arreglo respectivo
                 cambiando el tipo de variable cuando sea necesario*/
                Id[a]=Integer.parseInt(divisor[0]);
                Tipo[a]=divisor[1];
                Nombre[a]=divisor[2];
                Vida[a]=Double.parseDouble(divisor[3]);
                ptAt[a]=Double.parseDouble(divisor[4]);
                /**En caso de los arreglos tipo Booleano se determina si almacena
                 un valor true o false*/
                if (divisor[5].equalsIgnoreCase("Capturado")) {
                    Capturado[a]=true;
                }else{Capturado[a]=false;}
                if (divisor[6].equalsIgnoreCase("Vivo")) {
                    Estado[a]=true;
                }else{Estado[a]=false;}
                /**Se aumenta el contador para determinar la posición del siguiente
                 pokemon del listado*/
                a++;
                /**Se aumenta en 1 el contador de la cantidad de pokemons ingresados*/
                ingresados++;
            }
```

```
            /**Se cierra la lectura del documento*/
            lectura.close();
            /**Se cierra el documento*/
            archivoG.close();
        } catch (Exception e) {
            System.err.println("Error al cargar el archivo. Asegurese a que ingreso la\n"
                    + "ruta correcta o que el documento no tenga mas de 150\n"
                    + "Pokemons.");
        }
    }
}
```

En las siguientes clases de *carga* no se volveran a explicar a profundidad los metodos de **menu** y **carga** ya que funcionan de manera parecida, diferenciandose unicamente en el texto que imprimen (en el caso de **menu**) y en la cantidad de arreglos (en el caso de **carga**). Por lo que solo se explicara las diferencias significativas.

Clase CargaPokeballs

Esta clase posee los metodos necesarios para leer documentos csv que poseean un listado de las pokeballs con las que se interectuara mientras el programa se mantenga encendido.

Primero se crean los arreglos que se encargaran de almacenar los datos de todas las columnas de la tabla del documento csv que ingrese el usuario. Y otras variables que serviran como contador.

```
public class CargaPokeballs {
    //Se crean los arreglos que almacenaran los datos de la tabla.
    static int Id[]=new int[150],ContadorTipos=0,ingresados=0;
    //Las variables que no son arreglos se encargan de almacenar lo que escribe el usuario.
    static String Titulo[]=new String[2],CTipo[]=new String[4],Tipo[]=new String[150],op,ruta;
    //Variable para leer lo que escribe el usuario.
    static Scanner sc=new Scanner(System.in);
```

```
static void menu(){
    try {
        //Inicio del ciclo del menú principal.
        int cont=1;
        while (cont>0){
            //Se imprime el menú
            System.out.println("==================================\n"
                    + "|1.Carga Listado de Poke Balls\t|\n"
                    + "|2.Salir al Menu Principal\t|\n"
                    + "==================================");
            //Instrucción al usuario
            System.out.print("Ingrese el numero de opción que quiere realizar: ");
            /**Se almacena la información que escribe el usuario*/
            op=sc.nextLine();
            System.out.println();
```

```
            switch(op){
                case"1":
                    //Instrucción al usuario
                    System.out.print("Ingrese la ruta del archivo que poseea la lista de Entrenadores: ");
                    System.out.println("Ejemplo: C:\\Users\\Diego\\Desktop\\ListadoPokeballs.csv");
                    //Se almacena la ruta que ingreso el usuario.
                    ruta=sc.nextLine();
                    System.out.println();
                    /**Se reinicia la variable que indica la cantidad de pokemons que ingreso el
                     usuario*/
                    ingresados=0;
                    System.out.println("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
                    //Se manda a llamar el metodo seleccionado por el usuario
                    carga(ruta);
                    break;
                case"2":
                    //Se termina el ciclo que genera el menu.
                    cont--;
                    break;
                default:
                    System.err.println("Asegurese de ingresar un numero valido");
            }
        }
    } catch (Exception e) {
        System.err.println("Asegurese de ingresar un numero valido");
    }
}
```

Este metodo **carga(String r)** se diferencia a los demas en que tiene un ciclo while que se encarga de verificar que solo existan 4 tipos de pokeball. Funciona almacenando cada nuevo tipo en un arreglo de 4 posiciones que se encarga de almacenar los tipos de pokeball, si se llegara a encontrar

un nuevo tipo y el arreglo ya se lleno el programa terminara el ciclo *while* (mediante un *break*) y le dira al usuario que el archivo csv tiene mas de 4 tipos de pokeball.

```
static void carga(String r){
    /**Se reinicia los arreglos que almacenaran la información de la tabla y el contador
     de la cantidad de tipos de pokeballs permitada (CTIPO)*/
    Id=new int[150];Titulo=new String[2];CTipo=new String[4];Tipo=new String[150];

    try {
        /**Se guarda el archivo que indico el usario y se le prepara para que
         pueda leerse*/
        File archivo=new File(r);
        FileReader archivoG=new FileReader(archivo);
        BufferedReader lectura=new BufferedReader(archivoG);
        /**Se lee la primera linea del documento*/
        String codigo=lectura.readLine(),
                /**Se crea un arreglo que divira los apartados en cada linea.*/
               divisor[]=new String[2];
        /**Se almacenan los titulos de cada columna*/
        Titulo=codigo.split("\\s*,\\s*");

        /**Se inician los contadores que se utilizaran posteriormente.*/
        int a=0,d=0,c=0;

 /**Se comienza un ciclo que se repetira hasta que el documento se
  encuentre completamente vacío*/
 while ((codigo=lectura.readLine())!=null) {
     /**Se divira la línea en diferentes apartados*/
     divisor=codigo.split("\\s*,\\s*");
     Id[a]=Integer.parseInt(divisor[0]);

     /**Sub logaritmo, que se encarga de aumentar el contador de tipos de
      Pokeballs que tiene el documento, hasta que este alcance el limite
      de 4*/
    int i=0,b=0;d=0;

    /**Si ya se llenaron todos los espacios del contador se analizara que
     la lista no contenga mas de 4 tipos diferentes*/
     while (i<4 && c>3) {
         /**Se analiza 1 por 1 los espacios aumentando en 1 el contador b
          siempre que no coincida con algun valor del arreglo. Si los valores
          coiciden se dejara de analizar*/
        if (divisor[1].equalsIgnoreCase(CTipo[i])) {
           break;
         }else{
           b++;}
        i++;
     }
 while(i<4){
     /**Primero se analizan los cuatro espacios disponible, en caso de que
      conicida con uno se dejara de buscara.*/
     if (divisor[1].equalsIgnoreCase(CTipo[i])) {
         break;
     }else{d++;}
     /**Si ya se revisaron los 4 espacios y no coincidio con ninguno y si
      todavía hay espacios disponibles se llenara un casia con el nuevo
      tipo.*/
     if (c<4 && d>3) {
         CTipo[c]=divisor[1];
         c++;
         break;
     }
     i++;
 }
```

```
                /**Si se analizaron los cuatro espacios y no coincidio con ninguno se detendra
                 el ingreso de la lista, porque significa que hay mas de cuatro tipos de pokeball*/
                if (b>=4) {
                    System.err.println("ERROR. En el archivo existen mas de cuatro tipos de PokeBalls");
                    break;
                }else{
                    /**En caso de que aun hayan espacios vacíos o sea un tipo ya existente se agregara
                     el tipo de pokeball en el arreglo.*/
                    Tipo[a]=divisor[1];
                }
                /**Se aumenta la posición en los arreglos*/
                a++;
                /**Se aumenta en 1 el contador de la cantidad de pokeballs ingresadas*/
                ingresados++;
            }
            /**Se cierra la lectura del documento*/
            lectura.close();
            /**Se cierra el archivo*/
            archivoG.close();
        } catch (Exception e) {
            System.err.println("Error al cargar el archivo. Asegurese a que ingreso la\n"
                    + "ruta correcta o que el documento no tenga mas de 4\n"
                    + "tipo de pokeballs.");
        }
    }
}
```

Clase CargaGimnasios

Esta clase posee los metodos necesarios para leer documentos csv que poseean un listado de los gimnasios con los que se interectuara mientras el programa se mantenga encendido.

Primero se crean los arreglos que se encargaran de almacenar los datos de todas las columnas de la tabla del documento csv que ingrese el usuario. Y otras variables que serviran como contador.

```
public class CargaGimnasios implements Serializable{
    //Se crean los arreglos que almacenaran los datos de la tabla.
    static int Id[]=new int[25],ingresados=0;
    //Las variables que no son arreglos se encargan de almacenar lo que escribe el usuario.
    static String Titulo[]=new String[2],Lugar[]=new String[25],op,ruta;
    //Variable para leer lo que escribe el usuario.
    static Scanner sc=new Scanner(System.in);

    static void menu(){
        try {
            //Inicio del ciclo del menú principal.
            int cont=1;
            while (cont>0){
                //Se imprime el menú
                System.out.println("==================================\n"
                        + "|1.Carga Listado de Gimnasios\t|\n"
                        + "|2.Salir al Menu Principal\t|\n"
                        + "==================================");
                //Instrucción al usuario
                System.out.print("Ingrese el numero de opción que quiere realizar: ");
                op=sc.nextLine();
                System.out.println();
```

```
            switch(op){
                case"1":
                    //Instrucción al usuario
                    System.out.print("Ingrese la ruta del archivo que poseea la lista de Gimnasios: ");
                    System.out.println("Ejemplo: C:\\Users\\Diego\\Desktop\\ListadoGimnasios.csv");
                    //Se almacena la ruta que ingreso el usuario.
                    ruta=sc.nextLine();
                    System.out.println();
                    /**Se reinicia la variable que indica la cantidad de pokemons que ingreso el
                     usuario*/
                    ingresados=0;
                    System.out.println("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
                    //Se manda a llamar el metodo seleccionado por el usuario
                    carga(ruta);
                    break;
                case"2":
                    //Se termina el ciclo que genera el menu.
                    cont--;
                    break;
                default:
                    System.err.println("Asegurese de ingresar un numero valido");
            }
        }
    } catch (Exception e) {
        System.err.println("Asegurese de ingresar un numero valido");
    }
}
```

```
static  void carga(String r){
    /**Se reinicia los arreglos que almacenaran la información de la tabla*/
    Id=new int[25];Titulo=new String[2];Lugar=new String[25];

    try {
        /**Se guarda el archivo que indico el usario y se le prepara para que
         pueda leerse*/
        File archivo=new File(r);
        FileReader archivoG=new FileReader(archivo);
        BufferedReader lectura=new BufferedReader(archivoG);
        /**Se lee la primera linea del documento*/
        String codigo=lectura.readLine(),
                /**Se crea un arreglo que divira los apartados en cada linea.*/
               divisor[]=new String[2];
        /**Se almacenan los titulos de cada columna*/
        Titulo=codigo.split("\\s*,\\s*");
```

```
/**Se comienza un ciclo que se repetira hasta que el documento se
 encuentre completamente vacío*/
int a=0;
while ((codigo=lectura.readLine())!=null) {
    /**Se divira la línea en diferentes apartados*/
    divisor=codigo.split("\\s*,\\s*");
    /**Se asigna el valor del arreglo divisor en el arreglo respectivo
     cambiando el tipo de variable cuando sea necesario*/
    Id[a]=Integer.parseInt(divisor[0]);
    Lugar[a]=divisor[1];
    /**Se aumenta el contador para determinar la posición del siguiente
     pokemon del listado*/
    a++;
    /**Se aumenta en 1 el contador de la cantidad de pokemons ingresados*/
    ingresados++;
}
```

```
            /**Se cierra la lectura del documento*/
            lectura.close();
            /**Se cierra la lectura del documento*/
            archivoG.close();
            //Se imprime lo que almaceno el programa
            System.out.println(Titulo[0]+"\t"+Titulo[1]);
            int e=0;
            while(e<ingresados){
                System.out.println(Id[e]+"\t"+Lugar[e]);
                e++;
            }
        } catch (Exception e) {
            System.err.println("Error al cargar el archivo. Asegurese a que ingreso la\n"
                    + "ruta correcta o que el documento no tenga mas de 25\n"
                    + "gimnasios.");
        }
    }

}
```

Clase CargaEntrenador

Esta clase posee los metodos necesarios para leer documentos csv que poseean un listado de los Entrenadores con los que se interectuara mientras el programa se mantenga encendido.

Primero se crean los arreglos que se encargaran de almacenar los datos de todas las columnas de la tabla del documento csv que ingrese el usuario. Y otras variables que serviran como contador.

```
public class CargaEntrenador implements Serializable{
    //Se crean los arreglos que almacenaran los datos de la tabla.
    static int Id[]=new int[25],ingresados=0;
    //Las variables que no son arreglos se encargan de almacenar lo que escribe el usuario.
    static String Titulo[]=new String[2],Nombre[]=new String[25],op,ruta;
    //Variable para leer lo que escribe el usuario.
    static Scanner sc=new Scanner(System.in);
  static void menu(){
      try {
          //Inicio del ciclo del menú principal.
          int cont=1;
          while (cont>0){
              System.out.println("==================================\n"
                      + "|1.Carga Listado de Entrenadores|\n"
                      + "|2.Salir al Menu Principal\t|\n"
                      + "==================================");
              //Instrucción al usuario
              System.out.print("Ingrese el numero de opción que quiere realizar: ");
              /**Se almacena lo que escriba el usuario*/
              op=sc.nextLine();
              System.out.println();
```

```
            switch(op){
                case"1":
                    //Instrucción al usuario
                    System.out.print("Ingrese la ruta del archivo que poseea la lista de Entrenadores: ");
                    System.out.println("Ejemplo: C:\\Users\\Diego\\Desktop\\ListadoEntrenadores.csv");
                    //Se almacena la ruta que ingreso el usuario.
                    ruta=sc.nextLine();
                    System.out.println();
                    ingresados=0;
                    //Se deja un espacio considerable entre el menú y la acción seleccionada.
                    System.out.println("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
                    carga(ruta);
                    break;
                case"2":
                    cont--;
                    break;
                default:
                    System.err.println("Asegurese de ingresar un numero valido");
            }
        }
    } catch (Exception e) {
        System.err.println("Asegurese de ingresar un numero valido");
    }
}
```

```
/**Metodo que se encarga de almacenar la información que ingreso el usuario en los
   arreglos creados anteriormente*/
static void carga(String r){
    /**Se reinician los arreglos, para que se pudieran llenar de nuevo*/
    Id=new int[25];Titulo=new String[2];Nombre=new String[25];

    try {
        /**Se guarda el archivo que indico el usario y se le prepara para que
         pueda leerse*/
        File archivo=new File(r);
        FileReader archivoG=new FileReader(archivo);
        BufferedReader lectura=new BufferedReader(archivoG);
        /**Se lee la primera linea del documento*/
        String codigo=lectura.readLine(),
                /**Se crea un arreglo que divira los apartados en cada linea.*/
                divisor[]=new String[2];
        /**Se almacenan los titulos de cada columna*/
        Titulo=codigo.split("\\s*,\\s*");

    /**Se asigna el valor del arreglo divisor en el arreglo respectivo
         cambiando el tipo de variable cuando sea necesario*/
    int a=0;
    while ((codigo=lectura.readLine())!=null) {
        divisor=codigo.split("\\s*,\\s*");
        Id[a]=Integer.parseInt(divisor[0]);
        Nombre[a]=divisor[1];
        a++;
        ingresados++;
    }
```

```
            /**Se cierra la lectura del documento*/
            lectura.close();
            /**Se cierra la lectura del documento*/
            archivoG.close();
            //Se imprime lo que almaceno el programa
            System.out.println(Titulo[0]+"\t"+Titulo[1]);
            int e=0;
            while(e<ingresados){
                System.out.println(Id[e]+"\t"+Nombre[e]);
                e++;
            }
        } catch (Exception e) {
            System.err.println("Error al cargar el archivo. Asegurese a que ingreso la\n"
                    + "ruta correcta o que el documento no tenga mas de 25\n"
                    + "entrenadores.");
        }
    }
}
```

Clase CargaAlimentos

Esta clase posee los metodos necesarios para leer documentos csv que poseean un listado de los Alimentos con los que se interectuara mientras el programa se mantenga encendido.

Primero se crean los arreglos que se encargaran de almacenar los datos de todas las columnas de la tabla del documento csv que ingrese el usuario. Y otras variables que serviran como contador.

```
public class CargaAlimentos implements Serializable{
    //Se crean los arreglos que almacenaran los datos de la tabla.
    static int Id[]=new int[15],ingresados=0;
    //Las variables que no son arreglos se encargan de almacenar lo que escribe el usuario.
    static String Titulo[]=new String[3],Nombre[]=new String[15],op,ruta;
    static double Vida[]=new double[15];
    //Variable para leer lo que escribe el usuario.
    static Scanner sc=new Scanner(System.in);
```

```
static void menu(){
    try {
        //Inicio del ciclo del menú principal.
        int cont=1;
        while (cont>0){
            //Se imprime el menú
            System.out.println("==================================\n"
                    + "|1.Carga Listado de Alimentos\t|\n"
                    + "|2.Salir al Menu Principal\t|\n"
                    + "==================================");
            System.out.print("Ingrese el numero de opción que quiere realizar: ");
            /**Se almacena lo que escriba el usuario*/
            op=sc.nextLine();
            System.out.println();
```

```
            switch(op){
                case"1":
                    //Instrucción al usuario
                    System.out.print("Ingrese la ruta del archivo que poseea la lista de Alimentos: ");
                    System.out.println("Ejemplo: C:\\Users\\Diego\\Desktop\\ListadoAlimentos.csv");
                    //Se almacena la ruta que ingreso el usuario.
                    ruta=sc.nextLine();
                    System.out.println();
                    /**Se reinicia la variable que indica la cantidad de pokemons que ingreso el
                       usuario*/
                    ingresados=0;
                    System.out.println("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
                    //Se manda a llamar el metodo seleccionado por el usuario
                    carga(ruta);
                    break;
                case"2":
                    //Se termina el ciclo que genera el menu.
                    cont--;
                    break;
                default:
                    System.err.println("Asegurese de ingresar un numero valido");
            }
        }
    } catch (Exception e) {
        System.err.println("Asegurese de ingresar un número valido.");
    }
}
/**Metodo que se encarga de almacenar la información que ingreso el usuario en los
   arreglos creados anteriormente*/
static  void carga(String r){
    /**Se reinician los arreglos, para que se pudieran llenar de nuevo*/
     Id=new int[15];Titulo=new String[3];Nombre=new String[15];Vida=new double[15];

    try {
        /**Se guarda el archivo que indico el usario y se le prepara para que
         pueda leerse*/
        File archivo=new File(r);
        FileReader archivoG=new FileReader(archivo);
        BufferedReader lectura=new BufferedReader(archivoG);
        /**Se lee la primera linea del documento*/
        String codigo=lectura.readLine(),
                /**Se crea un arreglo que divira los apartados en cada linea.*/
               divisor[]=new String[3];
        /**Se almacenan los titulos de cada columna*/
        Titulo=codigo.split("\\s*,\\s*");
```

```
/**Se comienza un ciclo que se repetira hasta que el documento se
 encuentre completamente vacío*/
int a=0;
while ((codigo=lectura.readLine())!=null) {
    /**Se divira la línea en diferentes apartados*/
    divisor=codigo.split("\\s*,\\s*");
    /**Se asigna el valor del arreglo divisor en el arreglo respectivo
     cambiando el tipo de variable cuando sea necesario*/
    Id[a]=Integer.parseInt(divisor[0]);
    Nombre[a]=divisor[1];
    Vida[a]=Double.parseDouble(divisor[2]);
    /**Se aumenta el contador para determinar la posición del siguiente
     pokemon del listado*/
    a++;
    /**Se aumenta en 1 el contador de la cantidad de pokemons ingresados*/
    ingresados++;
}
```

```
            /**Se cierra la lectura del documento*/
            lectura.close();
            /**Se cierra la lectura del documento*/
            archivoG.close();
            //Se imprime lo que almaceno el programa
            System.out.println(Titulo[0]+"\t"+Titulo[1]+"\t\t"+Titulo[2]);
            int e=0;
            while(e<ingresados){
                System.out.println(Id[e]+"\t"+Nombre[e]+"\t\t"+Vida[e]);
                e++;
            }
        } catch (Exception e) {
            System.err.println("Error al cargar el archivo. Asegurese a que ingreso la\n"
                    + "ruta correcta o que el documento no tenga mas de 15\n"
                    + "alimentos.");
        }
    }

}
```

Clase AsignarPokemons

Esta clase se encarga de relacionar a un pokemon con una pokeball, para lograr esto se mandan a llamar a los arreglos de las clases *CargaPokemon* y *CargaPokeballs*, y mediante ciclos encontrar que pokemons se relacionan con que pokeballs.

Relacionar esta información provoca que los pokemons que se encuentren en estado salvaje pasen a estar en estado capturado cuando el programa encuentre que se le asigno una pokeball.

```
public class AsignarPokemons {
    //Se mandan a llamar las clases necesarias para unir los datos.
    static CargaPokeballs pokeball=new CargaPokeballs();
    static CargaPokemon pokemon=new CargaPokemon();
    //Se crean los arreglos que almacenaran los datos de la tabla.
    static int IdMon[]=new int[150],IdBall[]=new int[150],ingresados=0;
    //Las variables que no son arreglos se encargan de almacenar lo que escribe el usuario.
    static String Titulo[]=new String[2],op,ruta;
    //Variable para leer lo que escribe el usuario.
    static Scanner sc=new Scanner(System.in);
```

```
static void menu(){
    try {
        //Inicio del ciclo del menú principal.
        int cont=1;
        while (cont>0){
            //Se imprime el menú
            System.out.println("========================================\n"
                    + "|1.Asignar Pokemons a las Pokeballs\t|\n"
                    + "|2.Salir al Menu Principal\t\t|\n"
                    + "========================================");
            //Instrucción al usuario
            System.out.print("Ingrese el numero de opción que quiere realizar: ");
            /**Se almacena lo que escriba el usuario*/
            op=sc.nextLine();
            System.out.println();
```

```
            switch(op){
                case"1":
                    //Instrucción al usuario
                    System.out.print("Ingrese la ruta del archivo que poseea cómo se asignaran los pokemons: ");
                    System.out.println("Ejemplo: C:\\Users\\Diego\\Desktop\\AsignarPokemons.csv");
                    //Se almacena la ruta que ingreso el usuario.
                    ruta=sc.nextLine();
                    System.out.println("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
                    //Se manda a llamar el metodo seleccionado por el usuario
                    asignar(ruta);
                    break;
                case"2":
                    //Se termina el ciclo que genera el menu.
                    cont--;
                    break;
                default:
                    System.err.println("Asegurese de ingresar un numero valido");
            }
        }

    } catch (Exception e) {
        System.err.println("Asegurese de ingresar un número valido.");
    }
}
```

```
/**Metodo que se encarga de relacionar la información almacena de los pokemons
 con la información de las pokeballs*/
static void asignar(String r){
    /**Se reinician los arreglos, para que se pudieran llenar de nuevo*/
    IdMon=new int[150];IdBall=new int[150];
    /**Se reinicia la variable que indica la cantidad de pokemons que ingreso el
     usuario*/
    ingresados=0;

    try {
        /**Se guarda el archivo que indico el usario y se le prepara para que
         pueda leerse*/
        File archivo=new File(r);
        FileReader archivoG=new FileReader(archivo);
        BufferedReader lectura=new BufferedReader(archivoG);
        /**Se lee la primera linea del documento*/
        String codigo=lectura.readLine(),
                /**Se crea un arreglo que divira los apartados en cada linea.*/
                divisor[]=new String[2];
        /**Se almacenan los titulos de cada columna*/
        Titulo=codigo.split("\\s*,\\s*");
/**Se comienza un ciclo que se repetira hasta que el documento se
 encuentre completamente vacío*/
int a=0;
while ((codigo=lectura.readLine())!=null) {
    /**Se divira la línea en diferentes apartados*/
    divisor=codigo.split("\\s*,\\s*");
    /**Se asigna el valor del arreglo divisor en el arreglo respectivo
     cambiando el tipo de variable cuando sea necesario*/
    IdBall[a]=Integer.parseInt(divisor[0]);
    IdMon[a]=Integer.parseInt(divisor[1]);

    /**Ciclo que se encarga de revisar todos los pokemons emparejados,
     y pasarlos de estado salvaje a capturado*/
    for (int i = 0; i < 150; i++) {
        if (IdMon[a]==pokemon.Id[i]) {
            pokemon.Capturado[i]=true;
        }
    }
    /**Se aumenta el contador para determinar la posición del siguiente
     pokemon del listado*/
    a++;
    /**Se aumenta en 1 el contador de la cantidad de pokemons ingresados*/
    ingresados++;
}
        /**Se manda a llamar al metodo que se encarga de rescribir el documento
         con los pokemons que indico el usuario, para guardar los cambios*/
        rescribir();
        /**Se cierra la lectura del documento*/
        lectura.close();
        /**Se cierra la lectura del documento*/
        archivoG.close();
    } catch (Exception e) {
        System.err.println("Error asegurese de ingresar una ruta correcta.");
    }
}
```

Dentro de las clases *Asignar* se encuentra un metodo que se encarga de reescribir los documentos csv ingresados originalmente por el usuario en las clases *Carga* llamadas por la clase *Asignar* en cuestion.

```
static void rescribir(){
    try {
        /**Se crea un archivo nuevo, que tendra como ruta la misma ingresada por el usuario
         anteriormente, para que este documento sea cambiado por el nuevo documento*/
        File f=new File(pokemon.ruta);
        /**Se prepara el documente para poder escribir en el*/
        FileWriter w=new FileWriter(f);
        BufferedWriter bw=new BufferedWriter(w);
        PrintWriter pw=new PrintWriter(bw);

        /**Primero se escribe los titulos de las columnas*/
        pw.write(pokemon.Titulo[0]+","+pokemon.Titulo[1]+","+pokemon.Titulo[2]
        +","+pokemon.Titulo[3]+","+pokemon.Titulo[4]+","+pokemon.Titulo[5]+","+pokemon.Titulo[6]+"\n");

        for (int i = 0; i < pokemon.ingresados; i++) {
            String capturado,estado;
            /**Se establece si el pokemon se encuentra capturado o salvaje*/
            if (pokemon.Capturado[i]==true) {
                capturado="Capturado";
            }else{capturado="Salvaje";}
            /**Se establece si el pokemon se encuentra vivo o muerto*/
            if (pokemon.Estado[i]==true) {
                estado="Vivo";
            }else{estado="Muerto";}
                /**Se rescriben los valores de las filas en el mismo orden en como se debio
                 encontrar el documento original*/
                pw.write(pokemon.Id[i]+","+pokemon.Tipo[i]+","+pokemon.Nombre[i]+","+pokemon.Vida[i]
                +","+pokemon.ptAt[i]+","+capturado+","+estado+"\n");
            }
            /**Se cierra la función que permitia escribir en el documento*/
            pw.close();
            /**Se cierra el documento*/
            bw.close();
            /**Se vuelven a ingresar el documento y sus valores actualizados en el sistema*/
            pokemon.carga(pokemon.ruta);
        } catch (Exception e) {
            System.out.println("Asegurese que ingreso pokemons y pokeballs anteriormente."+e);
        }
    }
}
```

Clase AsignarPokeball

Esta clase se encarga de relacionar hasta 5 pokeballs con un entrenador, para lograr esto se mandan a llamar a los arreglos de las clases *CargaEntrenador* y *CargaPokeballs*, y mediante ciclos encontrar que pokeballs se relacionan con que entrenadores.

```
public class AsignarPokeball {
    //Se mandan a llamar las clases necesarias para unir los datos.
    static CargaEntrenador entrenador=new CargaEntrenador();
    static CargaPokeballs pokeball=new CargaPokeballs();
    /**Se crean los arreglos que almacenaran los datos de la tabla, tambien se crearon arreglos
     auxiliares para contar la cantidad de pokebolas de cada entrenador*/
    static int IdEnt[]=new int[25],AUXIdEnt[]=new int[25],ContBall[]=new int[150],IdBall[]=new int[150],
            ingresados=0,ingresados2=0;
    //Las variables que no son arreglos se encargan de almacenar lo que escribe el usuario.
    static String Titulo[]=new String[2],op,ruta;
    // para leer lo que escribe el usuario.
    static Scanner sc=new Scanner(System.in);
```

```
static void menu(){
    try {
        //Inicio del ciclo del menú principal.
        int cont=1;
        while (cont>0){
            System.out.println("========================================\n"
                    + "|1.Asignar Pokeballs a Entrenadores\t|\n"
                    + "|2.Salir al Menu Principal\t\t|\n"
                    + "========================================");
            //Instrucción al usuario
            System.out.print("Ingrese el numero de opción que quiere realizar: ");
            /**Se almacena lo que escriba el usuario*/
            op=sc.nextLine();
            System.out.println();
```

```
            switch(op){
                case"1":
                    //Instrucción al usuario
                    System.out.print("Ingrese la ruta del archivo que poseea cómo se asignaran los pokemons: ");
                    System.out.println("Ejemplo: C:\\Users\\Diego\\Desktop\\AsignarPokeballs.csv");
                    //Se almacena la ruta que ingreso el usuario.
                    ruta=sc.nextLine();
                    System.out.println();
                    /**Se reinicia la variable que indica la cantidad de pokemons que ingreso el
                     usuario*/
                    ingresados=0;
                    System.out.println("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
                    //Se manda a llamar el metodo seleccionado por el usuario
                    asignar(ruta);
                    break;
                case"2":
                    //Se termina el ciclo que genera el menu.
                    cont--;
                    break;
                default:
                    System.err.println("Asegurese de ingresar un numero valido");
            }
        }

    } catch (Exception e) {
        System.err.println("Asegurese de ingresar un número valido.");
    }
}
```

```
/**Metodo que se encarga de relacionar la información almacena de los pokemons
 con la información de las pokeballs*/
static void asignar(String r){
    try {
        /**Se reinician los arreglos, para que se pudieran llenar de nuevo*/
        IdEnt=new int[25];ContBall=new int[150];AUXIdEnt=new int[25];ContBall=new int[150];
        IdBall=new int[150];
        /**Se reinicia la variable que indica la cantidad de pokemons que ingreso el
        usuario*/
        ingresados=0;ingresados2=0;


        /**Se guarda el archivo que indico el usario y se le prepara para que
         pueda leerse*/
        File archivo=new File(r);
        FileReader archivoG=new FileReader(archivo);
        BufferedReader lectura=new BufferedReader(archivoG);
        /**Se lee la primera linea del documento*/
        String codigo=lectura.readLine(),
                /**Se crea un arreglo que divira los apartados en cada linea.*/
                divisor[]=new String[2];
        /**Se almacenan los titulos de cada columna*/
        Titulo=codigo.split("\\s*,\\s*");
```

```
  /**Se comienza un ciclo que se repetira hasta que el documento se
   encuentre completamente vacío*/
  int a=0;
  while ((codigo=lectura.readLine())!=null) {
      /**Se divira la línea en diferentes apartados*/
      divisor=codigo.split("\\s*,\\s*");

      int d=Integer.parseInt(divisor[0]);
      int b=0;
      /**Se esteblece si el entrenador que se posee actualmente ya se había
       analizado anteriormente*/
      while(b<25){
          if (d==IdEnt[b]) {
              break;
          }
          b++;
      }
      /**Si el entrenador es nuevo se guardara en el arreglo*/
      if(b==25){
          IdEnt[ingresados]=d;
          b=ingresados;
          /**Se desplaza la casilla en que se debe agregar al nuevo entrenador
           *que se encuentre*/
          ingresados++;
      }
      /**Se aumenta el contador de pokeballs del entranor actual*/
      ContBall[b]++;
```

```
                if(ContBall[b]>5){
                    System.err.println("En el archivo que subio uno o mas entrenadores tiene mas de"
                            + " 5 pokeballs asignadas");
                    break;
                }
                /**Se asigna el valor del arreglo divisor en el arreglo respectivo
                 cambiando el tipo de variable cuando sea necesario*/
                AUXIdEnt[a]=Integer.parseInt(divisor[0]);
                IdBall[a]=Integer.parseInt(divisor[1]);

                /**Se aumenta el contador para determinar la posición del siguiente
                 pokemon del listado*/
                ingresados2++;
                a++;
            }

        } catch (Exception e) {
            System.err.println("Error asegurese de ingresar una ruta correcta.");
        }
    }

}
```

Clase AsignarPelea

Esta clase se encarga de relacionar pokemons y hacer que peleen, para lograr esto se mandan a llamar a los arreglos de las clases *CargaPokemon* y *CargaGimnasios*.

Relacionar esta información provoca que la vida de los pokemons se reduzca debido a los ataques de sus contrincantes, incluso provocando que la vida de los pokemons llegue a 0, en estos casos el estado de vivo se pasa a muerto.

```
public class AsignarPelea {
    //Se mandan a llamar las clases necesarias para unir los datos.
    static CargaPokemon pokemon = new CargaPokemon();
    static CargaGimnasios gimnasio=new CargaGimnasios();
    //Se crean los arreglos que almacenaran los datos de la tabla.
    //Las variables que no son arreglos se encargan de almacenar lo que escribe el usuario.
    static String Titulo[]=new String[2],op,ruta,lugar,Listado[]=new String[200];
    //Controladores de los participantes de la pelea y de la cantidad de peleadas realizadas.
    static int Idgimnasio=0,Idpokemon1=0,Idpokemon2=0,ingresados=0,contLista=0,ganador=0;
    //Variable para leer lo que escribe el usuario.
    static Scanner sc=new Scanner(System.in);
static  void menu(){
    try {
        //Inicio del ciclo del menú principal.
        int cont=1;
        while (cont>0){
            //Se imprime el menú
            System.out.println("=========================================\n"
                    + "|1.Asignar Actividad de pelea\t\t|\n"
                    + "|2.Salir al Menu Principal\t\t|\n"
                    + "=========================================");
            //Instrucción al usuario
            System.out.print("Ingrese el numero de opción que quiere realizar: ");
            /**Se almacena lo que escriba el usuario*/
            op=sc.nextLine();
            System.out.println();
```

```
            switch(op){
                case"1":
                    //Instrucción al usuario
                    System.out.print("Ingrese la ruta del archivo que poseea cómo se asignaran los pokemons: ");
                    System.out.println("Ejemplo: C:\\Users\\Diego\\Desktop\\AsignarPelea.csv");
                    //Se almacena la ruta que ingreso el usuario.
                    ruta=sc.nextLine();
                    System.out.println("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
                    //Se manda a llamar el metodo seleccionado por el usuario
                    asignar(ruta);
                    break;
                case"2":
                    //Se termina el ciclo que genera el menu.
                    cont--;
                    break;
                default:
                    System.err.println("Asegurese de ingresar un numero valido");
            }
        }

    } catch (Exception e) {
        System.err.println("Asegurese de ingresar un número valido.");
    }
}
```

```
    static  void asignar(String r){
        try {
        //Se reinician los controladores de los participantes.
        Idgimnasio=0;Idpokemon1=0;Idpokemon2=0;

        /**Se guarda el archivo que indico el usario y se le prepara para que
            pueda leerse*/
        File archivo=new File(r);
        FileReader archivoG=new FileReader(archivo);
        BufferedReader lectura=new BufferedReader(archivoG);
        /**Se lee la primera linea del documento*/
        String codigo=lectura.readLine(),
                /**Se crea un arreglo que divira los apartados en cada linea.*/
                divisor[]=new String[3];
        /**Se almacenan los titulos de cada columna*/
        Titulo=codigo.split("\\s*,\\s*");

        /**Se reinician los controladores del las capacidades de los pokemons*/
        int d=0;double vida1=0,vida2=0,ataque1=0,ataque2=0;

            while ((codigo=lectura.readLine())!=null) {
                //Reiniciar variables
                int a=0,b=0,c=0,cont=0;String nombreG="";
                //Dividir el texto del archivo en un arreglo
                divisor=codigo.split(",");
```

```
//Asignar cada valor del arreglo a su respectiva variable
Idgimnasio=Integer.parseInt(divisor[0]);
Idpokemon1=Integer.parseInt(divisor[1]);
Idpokemon2=Integer.parseInt(divisor[2]);

//Encontrar nombre del Gimnasio
while (c<gimnasio.ingresados) {
    if (Idgimnasio==gimnasio.Id[c]) {
        nombreG=gimnasio.Lugar[c];
        break;
    }
    c++;
}

//Encontrar la vida de cada pokemon asignado
while (a<pokemon.ingresados) {
    if (Idpokemon1==pokemon.Id[a]) {
        vida1=pokemon.Vida[a];
        break;
    }
    a++;
}
while (b<pokemon.ingresados) {
    if (Idpokemon2==pokemon.Id[b]) {
        vida2=pokemon.Vida[b];
        break;
    }
    b++;
}
//Inicio de la pelea
while (cont<3) {
    //Turno del Pokemon 1
    vida2=vida2-pokemon.ptAt[a];
    //Comprobar si el pokemon 2 murio durante el turno
    if (vida2<=0) {
        pokemon.Vida[a]=vida1;
        vida2=0;
        pokemon.Vida[b]=vida2;
        pokemon.Estado[b]=false;
        break;
    }
    //Turno del Pokemon 2
    vida1=vida1-pokemon.ptAt[b];
    //Comprobar si el pokemon 1 murio durante el turno
    if (vida1<=0) {
        vida1=0;
        pokemon.Vida[a]=vida1;
        pokemon.Estado[a]=false;
        pokemon.Vida[b]=vida2;
        break;
    }
    cont++;
}
   pokemon.Vida[a]=vida1;
   pokemon.Vida[b]=vida2;
```

```
            pokemon.Vida[a]=vida1;
            pokemon.Vida[b]=vida2;
          //Indicar qué pokemon gano el combate
          if (vida1>vida2) {
             //Guardar el combate que se acaba de concretar.
              Listado[ingresados]="El ganador de la pelea de "+pokemon.Nombre[a]+" vs "+ pokemon.Nombre[b]+
                      " es "+pokemon.Nombre[a]+" en el gimnasio "+nombreG;
          }
          if (vida2>vida1) {
              //Guardar el combate que se acaba de concretar.
              Listado[ingresados]="El ganador de la pelea de "+pokemon.Nombre[a]+" vs "+ pokemon.Nombre[b]+
                      " es "+pokemon.Nombre[b]+" en el gimnasio "+nombreG;
          }
        ingresados++;
       }
       /**Se cierra la lectura del documento*/
       lectura.close();
       /**Se cierra la lectura del documento*/
       archivoG.close();
       /**Se manda a llamar al metodo que se encarga de rescribir el documento
       con los pokemons que indico el usuario, para guardar los cambios*/
       reescribir();

          } catch (Exception e) {
             System.err.println("Error asegurese de ingresar una ruta correcta, que la tabla "
                     + "este bien destribuida o de haber ingresado un listado de pokemons y gimnasios"
                     + " anteriormente.");
          }
      }

      static  void reescribir(){
          try {
              /**Se crea un archivo nuevo, que tendra como ruta la misma ingresada por el usuario
               anteriormente, para que este documento sea cambiado por el nuevo documento*/
              File f=new File(pokemon.ruta);
              /**Se prepara el documente para poder escribir en el*/
              FileWriter w=new FileWriter(f);
              BufferedWriter bw=new BufferedWriter(w);
              PrintWriter pw=new PrintWriter(bw);

              /**Primero se escribe los titulos de las columnas*/
              pw.write(pokemon.Titulo[0]+","+pokemon.Titulo[1]+","+pokemon.Titulo[2]
              +","+pokemon.Titulo[3]+","+pokemon.Titulo[4]+","+pokemon.Titulo[5]+","+pokemon.Titulo[6]+"\n");
      for (int i = 0; i < pokemon.ingresados; i++) {
          String capturado,estado;
          /**Se establece si el pokemon se encuentra capturado o salvaje*/
          if (pokemon.Capturado[i]==true) {
              capturado="Capturado";
          }else{capturado="Salvaje";}
          /**Se establece si el pokemon se encuentra vivo o muerto*/
          if (pokemon.Estado[i]==true) {
              estado="Vivo";
          }else{estado="Muerto";}

          /**Se rescriben los valores de las filas en el mismo orden en como se debio
           encontrar el documento original*/
          pw.write(pokemon.Id[i]+","+pokemon.Tipo[i]+","+pokemon.Nombre[i]+","+pokemon.Vida[i]
          +","+pokemon.ptAt[i]+","+capturado+","+estado+"\n");
      }
```

```
            /**Se cierra la función que permitia escribir en el documento*/
            pw.close();
            /**Se cierra el documento*/
            bw.close();
            /**Se vuelven a ingresar el documento y sus valores actualizados en el sistema*/
            pokemon.carga(pokemon.ruta);

        } catch (Exception e) {
            System.err.println("Error asegurese de ingresar una ruta correcta, que la tabla "
                    + "este bien destribuida o de haber ingresado un listado de pokemons y gimnasios"
                    + " anteriormente.");
        }
    }
}
```

Clase AsignarAlimentos

Esta clase se encarga de relacionar a un pokemon con uno o varios alimentos que consumira, para lograr esto se mandan a llamar a los arreglos de las clases *CargaPokemon* y *CargaAlimentos*.

Relacionar esta información provoca que la vida de los pokemons aumente, incluso provocando que la vida de los pokemons pase de valer 0 a un número mayor, en estos casos el estado de muerto se pasa a vivo.

```
public class AsignarAlimentos {
    //Se mandan a llamar las clases necesarias para unir los datos.
    static CargaAlimentos alimentos=new CargaAlimentos();
    static CargaPokemon pokemon=new CargaPokemon();
    //Se crean los arreglos que almacenaran los datos de la tabla.
    //Las variables que no son arreglos se encargan de almacenar lo que escribe el usuario.
    static String Titulo[]=new String[2],op,ruta,Listado[]=new String[200];
    static int IdAlimento[]=new int[150],IdBall[]=new int[150],ingresados=0,contLista=0;
    //Variable para leer lo que escribe el usuario.
    static Scanner sc=new Scanner(System.in);
 static void menu(){
    try {
        //Inicio del ciclo del menú principal.
        int cont=1;
        while (cont>0){
            //Se imprime el menú
            System.out.println("=========================================\n"
                    + "|1.Asignar Actividad de alimentación\t|\n"
                    + "|2.Salir al Menu Principal\t\t|\n"
                    + "=========================================");
            //Instrucción al usuario
            System.out.print("Ingrese el numero de opción que quiere realizar: ");
            /**Se almacena lo que escriba el usuario*/
            op=sc.nextLine();
            System.out.println();
```

```
            switch(op){
                case"1":
                    //Instrucción al usuario
                    System.out.print("Ingrese la ruta del archivo que poseea cómo se asignaran los pokemons: ");
                    System.out.println("Ejemplo: C:\\Users\\Diego\\Desktop\\AsignarPokeballs.csv");
                    //Se almacena la ruta que ingreso el usuario.
                    ruta=sc.nextLine();
                    System.out.println("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
                    //Se manda a llamar el metodo seleccionado por el usuario
                    asignar(ruta);
                    break;
                case"2":
                    //Se termina el ciclo que genera el menu.
                    cont--;
                    break;
                default:
                    System.err.println("Asegurese de ingresar un numero valido");
            }
        }

    } catch (Exception e) {
        System.err.println("Asegurese de ingresar un número valido.");
    }
}

/**Metodo que se encarga de relacionar la información almacena de los pokemons
 con la información de las pokeballs*/
static void asignar(String r){
    try {
        /**Se reinician los arreglos, para que se pudieran llenar de nuevo*/
        IdAlimento=new int[150];IdBall=new int[150];

        /**Se guarda el archivo que indico el usario y se le prepara para que
         pueda leerse*/
        File archivo=new File(r);
        FileReader archivoG=new FileReader(archivo);
        BufferedReader lectura=new BufferedReader(archivoG);
        /**Se lee la primera linea del documento*/
        String codigo=lectura.readLine(),
                /**Se crea un arreglo que divira los apartados en cada linea.*/
                divisor[]=new String[2];
        /**Se almacenan los titulos de cada columna*/
        Titulo=codigo.split("\\s*,\\s*");

        /**Controladores del estatus del pokemon*/
        int d=0;double Vida=0;

        /**Se comienza un ciclo que se repetira hasta que el documento se
         encuentre completamente vacío*/
        while ((codigo=lectura.readLine())!=null) {
            /**Se divira la línea en diferentes apartados*/
            divisor=codigo.split("\\s*,\\s*");

            int a=0,b=0;
            /**Se busca al alimento al que se le hace referencia*/
            while (a<alimentos.ingresados) {
                d=Integer.parseInt(divisor[0]);
                /**Cuando se encuentra el alimento al que se le hace referencia*/
                if (d==alimentos.Id[a]) {
                    /**Se guarda el valor de la vida que restablece el alimento*/
                    Vida=alimentos.Vida[a];
                    break;
                }
                a++;
            }
```

```
        /**Se busca al pokemon al que se le hace referencia*/
        while (b<pokemon.ingresados) {
            d=Integer.parseInt(divisor[1]);
            /**Cuando se encuentra el pokemon al que se le hace referencia*/
            if (d==pokemon.Id[b]) {
                /**Se le suma a su vida actual lo que restablece el alimento*/
                pokemon.Vida[b]=pokemon.Vida[b]+Vida;
                /**En caso que gracias a la comida la vida del pokemon haya pasado de
                 0 a un número mayor de 0 se cambiara el estado de muerto al estado
                 vivo*/
                if(pokemon.Vida[b]>0){
                    pokemon.Estado[b]=true;
                }
                break;
            }
            b++;
        }

        /**Se almacena la actividad realizada*/
        Listado[ingresados]=pokemon.Nombre[b]+" comio "+alimentos.Nombre[a]+" recuperando "
                + Vida +" Puntos de Vida";
        /**Se aumenta en 1 el contador de la cantidad de pokemons ingresados*/
        ingresados++;
    }

        /**Se cierra la lectura del documento*/
        lectura.close();
        /**Se cierra la lectura del documento*/
        archivoG.close();
        /**Se manda a llamar al metodo que se encarga de rescribir el documento
         con los pokemons que indico el usuario, para guardar los cambios*/
        reescribir();
    } catch (Exception e) {
        System.err.println("Error asegurese de ingresar una ruta correcta o que la tabla "
                + "este bien destribuida.");
    }
}

static void reescribir(){
    try {
        /**Se crea un archivo nuevo, que tendra como ruta la misma ingresada por el usuario
         anteriormente, para que este documento sea cambiado por el nuevo documento*/
        File f=new File(pokemon.ruta);
        /**Se prepara el documente para poder escribir en el*/
        FileWriter w=new FileWriter(f);
        BufferedWriter bw=new BufferedWriter(w);
        PrintWriter pw=new PrintWriter(bw);

        /**Primero se escribe los titulos de las columnas*/
        pw.write(pokemon.Titulo[0]+","+pokemon.Titulo[1]+","+pokemon.Titulo[2]
        +","+pokemon.Titulo[3]+","+pokemon.Titulo[4]+","+pokemon.Titulo[5]+","+pokemon.Titulo[6]+"\n");

        for (int i = 0; i < pokemon.ingresados; i++) {
            String capturado,estado;
            /**Se establece si el pokemon se encuentra capturado o salvaje*/
            if (pokemon.Capturado[i]==true) {
                capturado="Capturado";
            }else{capturado="Salvaje";}
            /**Se establece si el pokemon se encuentra vivo o muerto*/
            if (pokemon.Estado[i]==true) {
                estado="Vivo";
            }else{estado="Muerto";}
```

```
                /**Se rescriben los valores de las filas en el mismo orden en como se debio
                 encontrar el documento original*/
                pw.write(pokemon.Id[i]+","+pokemon.Tipo[i]+","+pokemon.Nombre[i]+","+pokemon.Vida[i]
                +","+pokemon.ptAt[i]+","+capturado+","+estado+"\n");
            }
            /**Se cierra la función que permitia escribir en el documento*/
            pw.close();
            /**Se cierra el documento*/
            bw.close();
            /**Se vuelven a ingresar el documento y sus valores actualizados en el sistema*/
            pokemon.carga(pokemon.ruta);
        } catch (Exception e) {
            System.out.println("Asegurese que ingreso pokemons y alimentos anteriormente."+e);
        }
    }

}
```

Clase ReporteEntrenadores

Esta clase se encarga de crear un documento HTML que muestra a todos los entrenadores *cargados* y los pokemons que se le *asignaron* para posteriormente abrirse automáticamente.

```
public class ReporteEntrenadores {
    //Se mandan a llamar las clases necesarias para generar el reporte.
    static CargaEntrenador entrenador=new CargaEntrenador();
    static CargaPokeballs pokeball=new CargaPokeballs();
    static CargaPokemon pokemon=new CargaPokemon();
    static AsignarPokeball Apokeball=new AsignarPokeball();
    static AsignarPokemons Apokemon=new AsignarPokemons();
    /**Variable que sirve para obtener la fecha y hora en que se genera el reporte*/
    static Calendar fecha=new GregorianCalendar();
    /**Variable para manejar los datos que se obtengan de los diversos metodos*/
    static String I;

    /**Metodo que se encarga de generar el reporte.*/
    public static void reporte(){
        try {
            /**Variable necesaria para abrir el archivo creado*/
            Desktop pc=Desktop.getDesktop();
            /**Variables para escribir el documento*/
            File f=new File("Reporte_entrenadores.html");
            FileWriter w=new FileWriter(f);
            BufferedWriter bw=new BufferedWriter(w);
            PrintWriter pw=new PrintWriter(bw);

        /**Se inicia el documento HTML*/
        pw.write("<!DOCTYPE html>\n");
        pw.write("<html>\n");
        pw.write("<head>\n");
            pw.write("<title>Reporte de los Entrenadores</title>\n");
```

```
                    pw.write("<style type=\"text/css\">\n" +
                            "body{\n" +
                                "background-color:rgba(224, 204, 72, 1);\n" +
                                "text-align: center;\n" +
                                "font-family:sans-serif;\n" +
                            "}\n"
                            + "table{\n" +
        "                   border: black 2px solid;\n" +
        "                   border-collapse: collapse;\n" +
        "                   }\n"
                            + "#td1{\n" +
        "                           border: black 2px solid;\n"
                            + "     background-color: skyblue;\n" +
        "                           }\n" +
        "                     #td2{\n" +
        "                           border: black 2px solid;\n"
                            + "     background-color: white;\n" +
        "                           }\n"
                            +"</style>");
                    pw.write("<meta charset=\"utf-8\">");
                pw.write("</head>\n");
                /**Inicio del cuerpo*/
                pw.write("<body>\n");
                /**Titulo del reporte*/
                    pw.write("<h1>Reporte de los entrenadores ingresados</h1>\n");
```

```
/**Fecha en que se genero el reporte*/
    pw.write("<h3>Documento Creado el "+fecha.get(Calendar.DAY_OF_MONTH)+
            "/"+((int)fecha.get(Calendar.MONTH)+1)+"/"+fecha.get(Calendar.YEAR)+" a las "
            +fecha.get(Calendar.HOUR)+":"+fecha.get(Calendar.MINUTE)+":"+fecha.get(Calendar.SECOND)+"</h3>\n")

    for (int i = 0; i < entrenador.ingresados; i++) {
        pw.write("<p>Entrenador "+entrenador.Nombre[i]+"</p>");
            //Se determina el entrenador en el metodo de Asignar Pokeball
            int a=0;
            while(a<Apokeball.ingresados){
                if(entrenador.Id[i]==Apokeball.IdEnt[a]){break;}
                a++;
            }
        pw.write("<p>Posee "+Apokeball.ContBall[a]+" pokeballs. Sus pokemones son:</p>\n");
        pw.write("<table align=\"center\">\n");
        //Se imprime los titulos
            pw.write("<tr>\n");
                pw.write("<td id=\"td1\">ID Pokeball</td>\n");
                pw.write("<td id=\"td1\">Tipo Pokeball</td>\n");
                pw.write("<td id=\"td1\">ID Pokemon</td>\n");
                pw.write("<td id=\"td1\">Pokemon</td>\n");
                pw.write("<td id=\"td1\">Vida</td>\n");
                pw.write("<td id=\"td1\">Estado</td>\n");
            pw.write("</tr>\n");
```

```
            for (int j = 0; j < Apokeball.ingresados2; j++) {
            pw.write("<tr>\n");
            //Se establece la fila correcta de acuerdo al entrenador actual
                if (Apokeball.AUXIdEnt[j]==entrenador.Id[i]) {
                    //Se establece el Id de la pokeball
                    pw.write("<td id=\"td2\">"+Apokeball.IdBall[j]+"</td>\n");
                    //Se establce el tipo de pokeball que almacena el pokemon
                    int b=0;
                    while(b<pokeball.ingresados){
                        if (pokeball.Id[b]==Apokeball.IdBall[j]) {
                          pw.write("<td id=\"td2\">"+pokeball.Tipo[b]+"</td>\n");
                          break;
                        }
                        b++;
                    }
                    //Se establce el Id del pokemon
                    int c=0;
                    while (c<Apokemon.ingresados) {
                        if (Apokemon.IdBall[c]==Apokeball.IdBall[j]) {
                            pw.write("<td id=\"td2\">"+Apokemon.IdMon[c]+"</td>\n");
                            break;
                        }
                        c++;
                    }

                      //Se establce el nombre, vida y estado del pokemon
                      int d=0;
                      while (d<pokemon.ingresados) {
                          if (pokemon.Id[d]==Apokemon.IdMon[c]) {
                              pw.write("<td id=\"td2\">"+pokemon.Nombre[d]+"</td>\n");
                              pw.write("<td id=\"td2\">"+pokemon.Vida[d]+"</td>\n");
                              String estado="";
                              if (pokemon.Estado[d]==true) {
                                  estado="Vivo";
                              }else{estado="Muerto";}
                              pw.write("<td id=\"td2\">"+estado+"</td>\n");
                              break;
                          }
                          d++;
                      }
                  }
              pw.write("</tr>\n");
              }

          pw.write("</table><br>\n");
      }

  pw.write("</body>\n");
  pw.write("</html>\n");
  /**Se finaliza el documento HTML*/

      /**Se cierra la capacidad de escribir en el documento*/
      pw.close();
      /**Se cierra el documento en el programa*/
      bw.close();
      /**Se abre el documento recien creado y guardado*/
      pc.open(f);
      } catch (Exception e) {
          System.err.println("Asegurese de haber realizado las actividades previas "
                  + "relacionadas");
      }
  }
}
```

Clase ReporteSalvajes

Esta clase se encarga de crear un documento HTML que muestra a todos los entrenadores los pokemons que se encuentren en estado salvaje.

```
public class ReporteSalvajes {
    //Se mandan a llamar las clases necesarias para generar el reporte.
    static CargaPokemon pokemon=new CargaPokemon();
    /**Variable que sirve para obtener la fecha y hora en que se genera el reporte*/
    static Calendar fecha=new GregorianCalendar();
    /**Variable para manejar los datos que se obtengan de los diversos metodos*/
    static String I;

    /**Metodo que se encarga de generar el reporte.*/
    public static void reportes(){
        try {
            /**Variable necesaria para abrir el archivo creado*/
            Desktop pc=Desktop.getDesktop();
            /**Variables para escribir el documento*/
            File f=new File("Reporte_salvajes.html");
            FileWriter w=new FileWriter(f);
            BufferedWriter bw=new BufferedWriter(w);
            PrintWriter pw=new PrintWriter(bw);

     /**Se inicia el documento HTML*/
     pw.write("<!DOCTYPE html>\n");
     pw.write("<html>\n");
     pw.write("<head>\n");
         pw.write("<title>Reporte de los Pokemons Salvajes</title>\n");
         pw.write("<style type=\"text/css\">\n" +
                  "body{\n" +
                       "background-color:rgba(117, 235, 148, 1);\n" +
                       "text-align: center;\n" +
                       "font-family:sans-serif;\n" +
                  "}\n"
                + "table{\n" +
                  border: black 2px solid;\n" +
                  border-collapse: collapse;\n" +
                  }\n"
                  + "#td1{\n" +
                          border: black 2px solid;\n"
                  + "     background-color: skyblue;\n" +
                          }\n" +
                     #td2{\n" +
                          border: black 2px solid;\n"
                  + "     background-color: white;\n" +
                          }\n"
                  +"</style>");
         pw.write("<meta charset=\"utf-8\">");
     pw.write("</head>\n");
```

```
/**Inicio del cuerpo*/
pw.write("<body>\n");
/**Título del reporte*/
    pw.write("<h1>Reporte de los Pokemons Salvajes</h1>\n");
/**Fecha en que se genero el reporte*/
    pw.write("<h3>Documento Creado el "+fecha.get(Calendar.DAY_OF_MONTH)+
            "/"+((int)fecha.get(Calendar.MONTH)+1)+"/"+fecha.get(Calendar.YEAR)+" a las "
            +fecha.get(Calendar.HOUR)+":"+fecha.get(Calendar.MINUTE)+":"+fecha.get(Calendar.SECOND)+"</h3>\n")

    pw.write("<table align=\"center\">\n");
    //Se escriben los títulos de las columnas
        pw.write("<tr>\n");
            pw.write("<td id=\"td1\">"+pokemon.Titulo[0]+"</td>\n");
            pw.write("<td id=\"td1\">"+pokemon.Titulo[1]+"</td>\n");
            pw.write("<td id=\"td1\">"+pokemon.Titulo[2]+"</td>\n");
            pw.write("<td id=\"td1\">"+pokemon.Titulo[3]+"</td>\n");
            pw.write("<td id=\"td1\">"+pokemon.Titulo[4]+"</td>\n");
            pw.write("<td id=\"td1\">"+pokemon.Titulo[5]+"</td>\n");
            pw.write("<td id=\"td1\">"+pokemon.Titulo[6]+"</td>\n");
        pw.write("</tr>\n");
        for (int i = 0; i < pokemon.ingresados; i++) {
                    //Se determina si el pokemon es salvaje
                    if(pokemon.Capturado[i]==false){
                    pw.write("<tr>\n");
                        pw.write("<td id=\"td2\">"+pokemon.Id[i]+"</td>\n");
                        pw.write("<td id=\"td2\">"+pokemon.Tipo[i]+"</td>\n");
                        pw.write("<td id=\"td2\">"+pokemon.Nombre[i]+"</td>\n");
                        pw.write("<td id=\"td2\">"+pokemon.Vida[i]+"</td>\n");
                        pw.write("<td id=\"td2\">"+pokemon.ptAt[i]+"</td>\n");
                        pw.write("<td id=\"td2\">Salvaje</td>\n");
                        //Se determina si el pokemon esta vivo o muerto
                        String estado;
                        if (pokemon.Estado[i]==true) {
                           estado="Vivo";
                        }else{estado="Muerto";}
                        pw.write("<td id=\"td2\">"+estado+"</td>\n");
                    pw.write("</tr>\n");
                    }
            }
            pw.write("</table><br>\n");

        pw.write("</body>\n");
        pw.write("</html>\n");
        /**Se finaliza el documento HTML*/
            /**Se cierra la capacidad de escribir en el documento*/
            pw.close();
            /**Se cierra el documento en el programa*/
            bw.close();
            /**Se abre el documento recien creado y guardado*/
            pc.open(f);
        } catch (Exception e) {
            System.err.println("Asegurese de haber realizado las actividades previas "
                    + "relacionadas");
        }
    }
}
```

Clase ReporteComidas

Esta clase se encarga de crear un documento HTML que muestra todas las actividades de comida que se han realizado antes de generarse el programa.

```
public class ReporteComidas {
    //Se mandan a llamar las clases necesarias para generar el reporte.
    static AsignarAlimentos alimentos=new AsignarAlimentos();
    /**Variable que sirve para obtener la fecha y hora en que se genera el reporte*/
    static Calendar fecha=new GregorianCalendar();
    /**Variable para manejar los datos que se obtengan de los diversos metodos*/
    static String I;

    public static void reporte(){
        try {
            /**Variable necesaria para abrir el archivo creado*/
            Desktop pc=Desktop.getDesktop();
            /**Variables para escribir el documento*/
            File f=new File("Reporte_comida.html");
            FileWriter w=new FileWriter(f);
            BufferedWriter bw=new BufferedWriter(w);
            PrintWriter pw=new PrintWriter(bw);

            /**Se inicia el documento HTML*/
            pw.write("<!DOCTYPE html>\n");
            pw.write("<html>\n");
            pw.write("<head>\n");

        pw.write("<title>Reporte de las Comidas Asignadas</title>\n");
        pw.write("<style type=\"text/css\">\n" +
                  "body{\n" +
                       "background-color:rgba(250, 207, 133, 1);\n" +
                       "text-align: center;\n" +
                       "font-family:sans-serif;\n" +
                 "}\n"
                + "table{\n" +
                border: black 2px solid;\n" +
                border-collapse: collapse;\n" +
                }\n"
                + "#td1{\n" +
                       border: black 2px solid;\n"
                + "    background-color: skyblue;\n" +
                       }\n" +
                   #td2{\n" +
                       border: black 2px solid;\n"
                + "    background-color: white;\n" +
                       }\n"
                +"</style>");
        pw.write("<meta charset=\"utf-8\">");
     pw.write("</head>\n");
```

```
pw.write("<body>\n");
/**Titulo del reporte*/
    pw.write("<h1>Reporte de las Comidas Asignadas</h1>\n");
/**Fecha en que se genero el reporte*/
    pw.write("<h3>Documento Creado el "+fecha.get(Calendar.DAY_OF_MONTH)+
            "/"+((int)fecha.get(Calendar.MONTH)+1)+"/"+fecha.get(Calendar.YEAR)+" a las "
            +fecha.get(Calendar.HOUR)+":"+fecha.get(Calendar.MINUTE)+":"+fecha.get(Calendar.SECOND)+"</h3>\n")

    pw.write("<table align=\"center\">\n");
        pw.write("<tr>\n");
        //Se escriben los títulos
            pw.write("<td id=\"td1\">No.</td>\n");
            pw.write("<td id=\"td1\">Actividad de comida</td>\n");
        pw.write("</tr>\n");

        //Se escriben todas las actividades
        for (int i = 0; i < alimentos.ingresados; i++) {
        pw.write("<tr>\n");
            pw.write("<td id=\"td2\">"+(i+1)+"</td>\n");
            pw.write("<td id=\"td2\">"+alimentos.Listado[i]+"</td>\n");
        pw.write("</tr>\n");
        }

    pw.write("</table><br>\n");

pw.write("</body>\n");
pw.write("</html>\n");
/**Se finaliza el documento HTML*/
```

```
            /**Se cierra la capacidad de escribir en el documento*/
            pw.close();
            /**Se cierra el documento en el programa*/
            bw.close();
            /**Se abre el documento recien creado y guardado*/
            pc.open(f);
        } catch (Exception e) {
            System.err.println("Asegurese de haber realizado las actividades previas "
                    + "relacionadas");
        }
    }
}
```

Clase ReportesPeleas

Esta clase se encarga de crear un documento HTML que muestra todas las peleas de comida que se han realizado antes de generarse el programa.

```
public class ReportesPeleas {
    //Se mandan a llamar las clases necesarias para generar el reporte.
    static AsignarPelea pelea=new AsignarPelea();
    /**Variable que sirve para obtener la fecha y hora en que se genera el reporte*/
    static Calendar fecha=new GregorianCalendar();
    /**Variable para manejar los datos que se obtengan de los diversos metodos*/
    static String I;

   public static void reportes(){
       try {
           /**Variable necesaria para abrir el archivo creado*/
            Desktop pc=Desktop.getDesktop();
            /**Variables para escribir el documento*/
            File f=new File("Reporte_pelea.html");
            FileWriter w=new FileWriter(f);
            BufferedWriter bw=new BufferedWriter(w);
            PrintWriter pw=new PrintWriter(bw);

            /**Se inicia el documento HTML*/
            pw.write("<!DOCTYPE html>\n");
            pw.write("<html>\n");
            pw.write("<head>\n");

                pw.write("<title>Reporte de las Peleas Asignadas</title>\n");
                pw.write("<style type=\"text/css\">\n" +
                         "body{\n" +
                              "background-color:rgba(237, 100, 100, 1);\n" +
                              "text-align: center;\n" +
                              "font-family:sans-serif;\n" +
                         "}\n"
                        + "table{\n" +
"                       border: black 2px solid;\n" +
"                       border-collapse: collapse;\n" +
"                       }\n"
                        + "#td1{\n" +
"                               border: black 2px solid;\n"
                        + "     background-color: skyblue;\n" +
"                               }\n" +
"                          #td2{\n" +
"                               border: black 2px solid;\n"
                        + "     background-color: white;\n" +
"                               }\n"
                        +"</style>");
                pw.write("<meta charset=\"utf-8\">");
            pw.write("</head>\n");
```

```
pw.write("<body>\n");
/**Titulo del reporte*/
    pw.write("<h1>Reporte de las Peleas Asignadas</h1>\n");
/**Fecha en que se genero el reporte*/
    pw.write("<h3>Documento Creado el "+fecha.get(Calendar.DAY_OF_MONTH)+
            "/"+((int)fecha.get(Calendar.MONTH)+1)+"/"+fecha.get(Calendar.YEAR)+" a las "
            +fecha.get(Calendar.HOUR)+":"+fecha.get(Calendar.MINUTE)+":"+fecha.get(Calendar.SECOND)+"</h3>\n")

    pw.write("<table align=\"center\">\n");
        pw.write("<tr>\n");
        //Se escriben los titulos
            pw.write("<td id=\"td1\">No.</td>\n");
            pw.write("<td id=\"td1\">Pelea</td>\n");
        pw.write("</tr>\n");

        //Se escriben todas las peleas
        for (int i = 0; i < pelea.ingresados; i++) {
            pw.write("<tr>\n");
            pw.write("<td id=\"td2\">"+(i+1)+"</td>\n");
            pw.write("<td id=\"td2\">"+pelea.Listado[i]+"</td>\n");
        pw.write("</tr>\n");
        }

    pw.write("</table><br>\n");
pw.write("</body>\n");
pw.write("</html>\n");
/**Se finaliza el documento HTML*/
```

```
                /**Se cierra la capacidad de escribir en el documento*/
                pw.close();
                /**Se cierra el documento en el programa*/
                bw.close();
                /**Se abre el documento recien creado y guardado*/
                pc.open(f);
            } catch (Exception e) {
                System.err.println("Asegurese de haber realizado las actividades previas "
                        + "relacionadas");
            }
        }
    }
```

Clase TopAtaque

Esta clase se encarga de mostrar de forma descendente los 5 pokemons con mas puntos de ataque. Antes de generar el reporte hay un método que primero se encarga de acomodar los puntos de ataque de mayor a menor mediante el algoritmo de burbuja.

```
public class TopAtaque {
    //Se mandan a llamar las clases necesarias para generar el reporte.
    static CargaPokemon pokemon=new CargaPokemon();
    /**Variable que sirve para obtener la fecha y hora en que se genera el reporte*/
    static Calendar fecha=new GregorianCalendar();
    /**Variable para manejar los datos que se obtengan de los diversos metodos*/
    static String I;
    static double[] top;
    static int[] id;

    //Reorganizar los pokemons segun sus puntos de ataque
    static void reorganizar(){
        top=new double[pokemon.ingresados];
        id=new int[pokemon.ingresados];
        for (int i = 0; i < pokemon.ingresados; i++) {
            top[i]=pokemon.ptAt[i];
            id[i]=pokemon.Id[i];
        }
```

```
            //Algoritmo burbuja
            for (int i = 0; i < top.length; i++) {
                for (int j = 0; j < (top.length-i-1); j++) {
                    if (top[j]<top[j+1]) {
                        double menor=top[j+1];
                        int id1=id[j+1];
                        top[j+1]=top[j];
                        id[j+1]=id[j];
                        top[j]=menor;
                        id[j]=id1;
                    }
                }
            }
        }
static void reporte(){
    try {
        reorganizar();
        /**Variable necesaria para abrir el archivo creado*/
        Desktop pc=Desktop.getDesktop();
        /**Variables para escribir el documento*/
        File f=new File("Reporte_topAtaque.html");
        FileWriter w=new FileWriter(f);
        BufferedWriter bw=new BufferedWriter(w);
        PrintWriter pw=new PrintWriter(bw);

        /**Se inicia el documento HTML*/
        pw.write("<!DOCTYPE html>\n");
        pw.write("<html>\n");
        pw.write("<head>\n");

    pw.write("<title>Top 5 pokemons con mayores Puntos de Ataque</title>\n");
    pw.write("<style type=\"text/css\">\n" +
                "body{\n" +
                    "background-color:rgba(229, 31, 31, 1);\n" +
                    "text-align: center;\n" +
                    "font-family:sans-serif;\n" +
             "}\n"
            + "table{\n" +
            border: black 2px solid;\n" +
            border-collapse: collapse;\n" +
            }\n"
            + "#td1{\n" +
                    border: black 2px solid;\n"
            + "     background-color: skyblue;\n" +
                    }\n" +
               #td2{\n" +
                    border: black 2px solid;\n"
            + "     background-color: white;\n" +
                    }\n"
            +"</style>");
    pw.write("<meta charset=\"utf-8\">");
pw.write("</head>\n");
```

```
/**Inicio del cuerpo*/
pw.write("<body>\n");

/**Titulo del reporte*/
    pw.write("<h1>Reporte de los 5 Pokemons con mayores Puntos de Ataque</h1>\n");
/**Fecha en que se genero el reporte*/
    pw.write("<h3>Documento Creado el "+fecha.get(Calendar.DAY_OF_MONTH)+
            "/"+((int)fecha.get(Calendar.MONTH)+1)+"/"+fecha.get(Calendar.YEAR)+" a las "
            +fecha.get(Calendar.HOUR)+":"+fecha.get(Calendar.MINUTE)+":"+fecha.get(Calendar.SECOND)+"</h3>\n")

    pw.write("<table align=\"center\">\n");
    //Se escriben los titulos de las columnas
        pw.write("<tr>\n");
            pw.write("<td id=\"td1\">"+pokemon.Titulo[0]+"</td>\n");
            pw.write("<td id=\"td1\">"+pokemon.Titulo[1]+"</td>\n");
            pw.write("<td id=\"td1\">"+pokemon.Titulo[2]+"</td>\n");
            pw.write("<td id=\"td1\">"+pokemon.Titulo[3]+"</td>\n");
            pw.write("<td id=\"td1\">"+pokemon.Titulo[4]+"</td>\n");
            pw.write("<td id=\"td1\">"+pokemon.Titulo[5]+"</td>\n");
            pw.write("<td id=\"td1\">"+pokemon.Titulo[6]+"</td>\n");
        pw.write("</tr>\n");
```

```
                //Se escriben los 5 top
                for (int i = 0; i < 5; i++) {
                pw.write("<tr>\n");
                    int b=0;
                    while(b<pokemon.ingresados){
                        if (pokemon.Id[b]==id[i]) {
                            pw.write("<td id=\"td2\">"+pokemon.Id[b]+"</td>\n");
                            pw.write("<td id=\"td2\">"+pokemon.Tipo[b]+"</td>\n");
                            pw.write("<td id=\"td2\">"+pokemon.Nombre[b]+"</td>\n");
                            pw.write("<td id=\"td2\">"+pokemon.Vida[b]+"</td>\n");
                            pw.write("<td id=\"td2\">"+pokemon.ptAt[b]+"</td>\n");
                            //Se determina si el pokemon esta capturado o salvaje.
                            String capturado;
                            if (pokemon.Capturado[b]==true) {
                                capturado="Capturado";
                            }else{capturado="Salvaje";}
                            pw.write("<td id=\"td2\">"+capturado+"</td>\n");
                            //Se determina si el pokemon esta vivo o muerto.
                            String estado;
                            if (pokemon.Estado[b]==true) {
                                estado="Vivo";
                            }else{estado="Muerto";}
                            pw.write("<td id=\"td2\">"+estado+"</td>\n");
                        }
                        b++;
                    }
                pw.write("</tr>\n");
```

```
                }

                pw.write("</table><br>\n");

            pw.write("</body>\n");
            pw.write("</html>\n");
            /**Se finaliza el documento HTML*/

            /**Se cierra la capacidad de escribir en el documento*/
            pw.close();
            /**Se cierra el documento en el programa*/
            bw.close();
            /**Se abre el documento recien creado y guardado*/
            pc.open(f);

        } catch (Exception e) {
            System.err.println("Asegurese de haber realizado las actividades previas "
                    + "relacionadas"+e);
        }
    }
}
```

Clase TopSalud

Esta clase se encarga de mostrar de forma descendente los 5 alimentos con más puntos de vida que recupera. Antes de generar el reporte hay un método que primero se encarga de acomodar los puntos de vida de mayor a menor mediante el algoritmo de burbuja.

```
public class TopSalud {
    //Se mandan a llamar las clases necesarias para generar el reporte.
    static CargaAlimentos alimentos=new CargaAlimentos();
    /**Variable que sirve para obtener la fecha y hora en que se genera el reporte*/
    static Calendar fecha=new GregorianCalendar();
    /**Variable para manejar los datos que se obtengan de los diversos metodos*/
    static String I;
    static double[] top;
    static int[] id;

    //Reorganizar los pokemons segun sus puntos de ataque
    static void reorganizar(){
        top=new double[alimentos.ingresados];
        id=new int[alimentos.ingresados];
        for (int i = 0; i < alimentos.ingresados; i++) {
            top[i]=alimentos.Vida[i];
            id[i]=alimentos.Id[i];
        }
```

```
        //Algoritmo burbuja
        for (int i = 0; i < top.length; i++) {
            for (int j = 0; j < (top.length-i-1); j++) {
                if (top[j]<top[j+1]) {
                    double menor=top[j+1];
                    int id1=id[j+1];
                    top[j+1]=top[j];
                    id[j+1]=id[j];
                    top[j]=menor;
                    id[j]=id1;
                }
            }
        }
    }

void reporte(){
    reorganizar();
    try {
        /**Variable necesaria para abrir el archivo creado*/
        Desktop pc=Desktop.getDesktop();
        /**Variables para escribir el documento*/
        File f=new File("Reporte_topAtaque.html");
        FileWriter w=new FileWriter(f);
        BufferedWriter bw=new BufferedWriter(w);
        PrintWriter pw=new PrintWriter(bw);

         /**Se inicia el documento HTML*/
        pw.write("<!DOCTYPE html>\n");
        pw.write("<html>\n");
        pw.write("<head>\n");
     pw.write("<title>Top 5 Alimentos con mas Vida</title>\n");
     pw.write("<style type=\"text/css\">\n" +
              "body{\n" +
                   "background-color:rgba(152, 239, 118, 1);\n" +
                   "text-align: center;\n" +
                   "font-family:sans-serif;\n" +
              "}\n"
            + "table{\n" +
            border: black 2px solid;\n" +
            border-collapse: collapse;\n" +
            }\n"
            + "#td1{\n" +
                    border: black 2px solid;\n"
            + "     background-color: skyblue;\n" +
                    }\n" +
               #td2{\n" +
                    border: black 2px solid;\n"
            + "     background-color: white;\n" +
                    }\n"
            +"</style>");
     pw.write("<meta charset=\"utf-8\">");
pw.write("</head>\n");
```

```
/**Inicio del cuerpo*/
pw.write("<body>\n");

/**Título del reporte*/
    pw.write("<h1>Reporte de los 5 Alimentos que mas vida recuperan</h1>\n");
/**Fecha en que se genero el reporte*/
    pw.write("<h3>Documento Creado el "+fecha.get(Calendar.DAY_OF_MONTH)+
            "/"+((int)fecha.get(Calendar.MONTH)+1)+"/"+fecha.get(Calendar.YEAR)+" a las "
            +fecha.get(Calendar.HOUR)+":"+fecha.get(Calendar.MINUTE)+":"+fecha.get(Calendar.SECOND)+"</h3>\n")

    pw.write("<table align=\"center\">\n");
        //Se escriben los títulos de las columnas
        pw.write("<tr>\n");
            pw.write("<td id=\"td1\">"+alimentos.Titulo[0]+"</td>\n");
            pw.write("<td id=\"td1\">"+alimentos.Titulo[1]+"</td>\n");
            pw.write("<td id=\"td1\">"+alimentos.Titulo[2]+"</td>\n");
        pw.write("</tr>\n");
        //Se escriben los 5 top
        for (int i = 0; i < 5; i++) {
            pw.write("<tr>\n");
                int b=0;
                while(b<alimentos.ingresados){
                    if (alimentos.Id[b]==id[i]) {
                       pw.write("<td id=\"td2\">"+alimentos.Id[b]+"</td>\n");
                       pw.write("<td id=\"td2\">"+alimentos.Nombre[b]+"</td>\n");
                       pw.write("<td id=\"td2\">"+alimentos.Vida[b]+"</td>\n");
                    }
                    b++;
                }
            pw.write("</tr>\n");
        }
                pw.write("</table><br>\n");

            pw.write("</body>\n");
            pw.write("</html>\n");
            /**Se finaliza el documento HTML*/

            /**Se cierra la capacidad de escribir en el documento*/
            pw.close();
            /**Se cierra el documento en el programa*/
            bw.close();
            /**Se abre el documento recien creado y guardado*/
            pc.open(f);
        } catch (Exception e) {
            System.err.println("Asegurese de haber realizado las actividades previas "
                    + "relacionadas");
        }
    }

}
```

Clase Guardar

Esta clase se encarga de guardar en el sistema la información que almacenan las clases de *carga* de tal forma que después pueda ser tomada por el programa sin necesidad de tener que volver a escribir las rutas de los archivos.

La información la guarda de dos maneras, mediante archivos *.bin* y archivos *.csv*. Estos archivos de guardado se almacenan en la carpeta que se encuentre el archivo .jar que dé inicio al programa.

```
public class Guardar {
   //Se mandan a llamar las clases necesarias para unir los datos.
    static CargaAlimentos alimentos;
    static CargaGimnasios gimnasio;
    static CargaEntrenador entrenador;
    static CargaPokeballs pokeball;
    static CargaPokemon pokemon;
     public Guardar(){
         alimentos=new CargaAlimentos();
         gimnasio=new CargaGimnasios();
         entrenador=new CargaEntrenador();
         pokeball=new CargaPokeballs();
         pokemon=new CargaPokemon();
     }
    //Metodo que se encarga de guardar los archivos
    public static void guardar() {
        try {
           //Crear archivos BIN
            ObjectOutputStream os1 = new ObjectOutputStream(new FileOutputStream("pokemons.bin"));
            os1.writeObject(pokemon);  // this es de tipo DatoUdp
            os1.close();
            ObjectOutputStream os2 = new ObjectOutputStream(new FileOutputStream("pokeballs.bin"));
            os2.writeObject(pokeball);  // this es de tipo DatoUdp
            os2.close();
            ObjectOutputStream os3 = new ObjectOutputStream(new FileOutputStream("entrenadores.bin"));
            os3.writeObject(entrenador);  // this es de tipo DatoUdp
            os3.close();
            ObjectOutputStream os4 = new ObjectOutputStream(new FileOutputStream("gimnasio.bin"));
            os4.writeObject(gimnasio);  // this es de tipo DatoUdp
            os4.close();
```

```
        ObjectOutputStream os5 = new ObjectOutputStream(new FileOutputStream("alimento.bin"));
        os5.writeObject(alimentos);  // this es de tipo DatoUdp
        os5.close();
        guardarpokemon();
        guardarentrenador();
        guardarpokeball();
        guardargimnasio();
        guardaralimento();
        System.out.println("Se guardaron los archivos correctamente.");

    } catch (Exception e) {
        System.err.println("No se pudieron guardar los archivos");
    }
}
static void guardarpokemon(){
    try {
        /**Se crea un archivo nuevo, que tendra como ruta la misma ingresada por el usuario
         anteriormente, para que este documento sea cambiado por el nuevo documento*/
        File f=new File("pokemon.csv");
        /**Se prepara el documente para poder escribir en el*/
        FileWriter w=new FileWriter(f);
        BufferedWriter bw=new BufferedWriter(w);
        PrintWriter pw=new PrintWriter(bw);

        /**Primero se escribe los titulos de las columnas*/
        pw.write(pokemon.Titulo[0]+","+pokemon.Titulo[1]+","+pokemon.Titulo[2]
        +","+pokemon.Titulo[3]+","+pokemon.Titulo[4]+","+pokemon.Titulo[5]+","+pokemon.Titulo[6]+"\n");
```

```
        for (int i = 0; i < pokemon.ingresados; i++) {
            String capturado,estado;
            /**Se establece si el pokemon se encuentra capturado o salvaje*/
            if (pokemon.Capturado[i]==true) {
                capturado="Capturado";
            }else{capturado="Salvaje";}
            /**Se establece si el pokemon se encuentra vivo o muerto*/
            if (pokemon.Estado[i]==true) {
                estado="Vivo";
            }else{estado="Muerto";}

            /**Se rescriben los valores de las filas en el mismo orden en como se debio
             encontrar el documento original*/
            pw.write(pokemon.Id[i]+","+pokemon.Tipo[i]+","+pokemon.Nombre[i]+","+pokemon.Vida[i]
            +","+pokemon.ptAt[i]+","+capturado+","+estado+"\n");
        }
        /**Se cierra la función que permitia escribir en el documento*/
        pw.close();
        /**Se cierra el documento*/
        bw.close();
    } catch (Exception e) {
         System.out.println("Asegurese que ingreso pokemons anteriormente."+e);
    }
}
```

```
static void guardarentrenador(){
    try {
        /**Se crea un archivo nuevo, que tendra como ruta la misma ingresada por el usuario
         anteriormente, para que este documento sea cambiado por el nuevo documento*/
        File f=new File("entrenador.csv");
        /**Se prepara el documente para poder escribir en el*/
        FileWriter w=new FileWriter(f);
        BufferedWriter bw=new BufferedWriter(w);
        PrintWriter pw=new PrintWriter(bw);

        /**Primero se escribe los titulos de las columnas*/
        pw.write(entrenador.Titulo[0]+","+entrenador.Titulo[1]+"\n");

        for (int i = 0; i < entrenador.ingresados; i++) {
            /**Se rescriben los valores de las filas en el mismo orden en como se debio
             encontrar el documento original*/
            pw.write(entrenador.Id[i]+","+entrenador.Nombre[i]+"\n");
        }
        /**Se cierra la función que permitia escribir en el documento*/
        pw.close();
        /**Se cierra el documento*/
        bw.close();
    } catch (Exception e) {
         System.out.println("Asegurese que ingreso pokeballs anteriormente."+e);
    }
}
```

```
static void guardarpokeball(){
    try {
        /**Se crea un archivo nuevo, que tendra como ruta la misma ingresada por el usuario
         anteriormente, para que este documento sea cambiado por el nuevo documento*/
        File f=new File("pokeball.csv");
        /**Se prepara el documente para poder escribir en el*/
        FileWriter w=new FileWriter(f);
        BufferedWriter bw=new BufferedWriter(w);
        PrintWriter pw=new PrintWriter(bw);

        /**Primero se escribe los titulos de las columnas*/
        pw.write(pokeball.Titulo[0]+","+pokeball.Titulo[1]+"\n");

        for (int i = 0; i < pokeball.ingresados; i++) {
            /**Se rescriben los valores de las filas en el mismo orden en como se debio
             encontrar el documento original*/
            pw.write(pokeball.Id[i]+","+pokeball.Tipo[i]+"\n");
        }
        /**Se cierra la función que permitia escribir en el documento*/
        pw.close();
        /**Se cierra el documento*/
        bw.close();
    } catch (Exception e) {
         System.out.println("Asegurese que ingreso pokeballs anteriormente."+e);
    }
}
```

```
static void guardargimnasio(){
    try {
        /**Se crea un archivo nuevo, que tendra como ruta la misma ingresada por el usuario
         anteriormente, para que este documento sea cambiado por el nuevo documento*/
        File f=new File("gimnasio.csv");
        /**Se prepara el documente para poder escribir en el*/
        FileWriter w=new FileWriter(f);
        BufferedWriter bw=new BufferedWriter(w);
        PrintWriter pw=new PrintWriter(bw);

        /**Primero se escribe los titulos de las columnas*/
        pw.write(gimnasio.Titulo[0]+","+gimnasio.Titulo[1]+"\n");

        for (int i = 0; i < gimnasio.ingresados; i++) {
            /**Se rescriben los valores de las filas en el mismo orden en como se debio
             encontrar el documento original*/
            pw.write(gimnasio.Id[i]+","+gimnasio.Lugar[i]+"\n");
        }
        /**Se cierra la función que permitia escribir en el documento*/
        pw.close();
        /**Se cierra el documento*/
        bw.close();
    } catch (Exception e) {
         System.out.println("Asegurese que ingreso pokeballs anteriormente."+e);
    }
}
```

```
    static void guardaralimento(){
        try {
            /**Se crea un archivo nuevo, que tendra como ruta la misma ingresada por el usuario
             anteriormente, para que este documento sea cambiado por el nuevo documento*/
            File f=new File("alimento.csv");
            /**Se prepara el documente para poder escribir en el*/
            FileWriter w=new FileWriter(f);
            BufferedWriter bw=new BufferedWriter(w);
            PrintWriter pw=new PrintWriter(bw);

            /**Primero se escribe los titulos de las columnas*/
            pw.write(alimentos.Titulo[0]+","+alimentos.Titulo[1]+","+alimentos.Titulo[2]+"\n");

            for (int i = 0; i < alimentos.ingresados; i++) {
                /**Se rescriben los valores de las filas en el mismo orden en como se debio
                 encontrar el documento original*/
                pw.write(alimentos.Id[i]+","+alimentos.Nombre[i]+","+alimentos.Vida[i]+"\n");
            }
            /**Se cierra la función que permitia escribir en el documento*/
            pw.close();
            /**Se cierra el documento*/
            bw.close();
        } catch (Exception e) {
             System.out.println("Asegurese que ingreso pokeballs anteriormente."+e);
        }
    }
}
```

Clase CargarArchivo

Esta clase se encarga de tomar los archivos *.bin* y *.csv* que se debieron guardar anteriormente y volver a cargarlos en las clases *carga*, para que estas almacenen nuevamente la última información que se guardó.

```
public class CargarArchivos {
    //Se mandan a llamar las clases necesarias para unir los datos.
    static CargaAlimentos alimentos;
    static CargaGimnasios gimnasio;
    static CargaEntrenador entrenador;
    static CargaPokeballs pokeball;
    static CargaPokemon pokemon;
     public CargarArchivos(){
         alimentos=new CargaAlimentos();
         gimnasio=new CargaGimnasios();
         entrenador=new CargaEntrenador();
         pokeball=new CargaPokeballs();
         pokemon=new CargaPokemon();
     }
   //Metodo que se encarga de cargar los archivos creados
    public static void carga(){
        try {
            ObjectInputStream o1 = new ObjectInputStream(new FileInputStream("pokemons.bin"));
                    pokemon = (CargaPokemon) o1.readObject();
            ObjectInputStream o2 = new ObjectInputStream(new FileInputStream("pokeballs.bin"));
                    pokeball = (CargaPokeballs) o2.readObject();
            ObjectInputStream o3 = new ObjectInputStream(new FileInputStream("entrenadores.bin"));
                    entrenador = (CargaEntrenador) o3.readObject();
            ObjectInputStream o4 = new ObjectInputStream(new FileInputStream("gimnasio.bin"));
                    gimnasio = (CargaGimnasios) o4.readObject();
            ObjectInputStream o5 = new ObjectInputStream(new FileInputStream("alimento.bin"));
                    alimentos = (CargaAlimentos) o5.readObject();
```

```
            pokemon.carga("pokemon.csv");
            entrenador.carga("entrenador.csv");
            pokeball.carga("pokeball.csv");
            gimnasio.carga("gimnasio.csv");
            alimentos.carga("alimento.csv");
            System.out.println("Se cargaron los archivos correctamente");
        } catch (Exception e) {
            System.err.println("No se encontraron Archivos anteriormente Guardados");
        }

    }
}
```

Glosario

Arreglo: También llamados *Array*, son estructura de datos que nos permite almacenar un conjunto de datos de un mismo tipo, el tamaño del arreglo debe definirse por el programador, este puede ser un tamaño fijo o que se adapte a las acciones del que use el programa.

Bit: Un bit es un dígito del sistema de numeración binario. La capacidad de almacenamiento de una memoria digital también se mide en bit. es la unidad mínima de información empleada en informática, en cualquier dispositivo digital, o en la teoría de la información. Con él, podemos representar dos valores cualesquiera, como verdadero o falso, abierto o cerrado, blanco o negro, norte o sur, etc.

Booleano: Es un tipo de variable que puede tener únicamente dos valores true o false.

Byte: Conjunto de 8 bits que recibe el tratamiento de una unidad y que constituye el mínimo elemento de memoria direccionable de una computadora.

Clase: Es una plantilla que define la forma de un objeto. En ella se agrupan datos y métodos que operarán sobre esos datos. En java, una clase se define con la palabra reservada *class*.

Condicional: Se refiere que una variable, arreglo o matriz debe cumplir con una condición específica (como lo puede ser almacenar un valor especifico, tener un tamaño determinado o ser mayor, menor o igual a otro valor) para llevar a cabo una acción especifica.

Inicialización: Significa asignarle algún valor, ya sea de tipo numérico, lógico o de otro tipo a una variable, arreglo o matriz.

Instanciar: Es el proceso de generar un ejemplar de una **clase**, es decir, la **clase** es como una declaración de una forma y el objeto es un caso o elemento concreto que responde a esa forma.

Método: Son subrutinas que manipulan los datos definidos por la clase y, en muchos casos, brindan acceso a esos datos.

Objeto: Se pueden entender como los espacios dentro del programa que almacenan información, y se encuentran dentro de una clase. La información que almacenan los objetos puede ser compartida entre otros objetos, puede ser cambiada, o puede servir como indicador. Los objetos pueden ser identificados como variables, arreglos, matrices, etcétera.

Serialización: Proceso de codificación de un objeto en un medio de almacenamiento (como puede ser un archivo, o un buffer de memoria) con el fin de transmitirlo a través de una conexión en red como una serie de bytes o en un formato humanamente más legible.

Static: Una clase, método o campo declarado como estático puede ser accedido o invocado sin la necesidad de tener que instanciar un objeto de la clase.