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…………………………………………………………………1

Clase Principal………………………………………………………………………………...1-2

Menú Reportes…………………………………………………………………………..3

Clase CargaPokemon………………………………………………………………………..4-6

Clase CargaPokeballs………………………………………………………………………..6-9

Clase CargaGimnasios…..…………………………………………………………………9-11

Clase CargaEntrenador…..……………………………………………………………….11-13

Clase CargaAlimentos…..………………………………………………………………...13-15

Clase AsignarPokemons………………………………………………………………….16-18

Clase AsignarPokeball…………………………………………………………………….19-21

Clase AsignarPelea………………………………………………………………………..21-25

Clase AsignarAlimentos…………………………………………………………………...25-28

Clase ReporteEntrenadores…………………………………………………………...…28-30

Clase ReporteSalvajes…………………………………………………………………….31-32

Clase ReporteComidas……………………………………………………………………33-34

Clase ReportesPeleas…………………………………………………………………….35-36

Clase TopAtaque…………………………………………………………………………..36-39

Clase TopSalud……………………………………………………………………………39-41

Clase Guardar……………………………………………………………………………...42-44

Clase CargarArchivos…………………………………………………………………………45

Glosario…………………………………………………………………………………………46

Especificaciones del Software

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

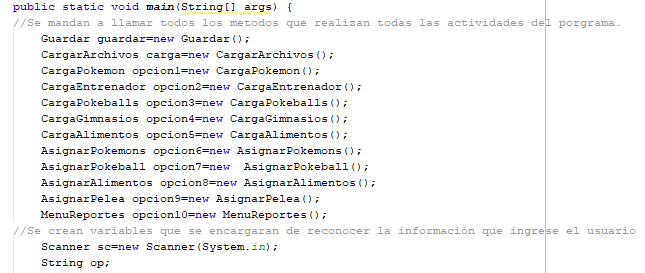
La versión jdk de Java que se utilizo es **jdk1.8.0\_111**.

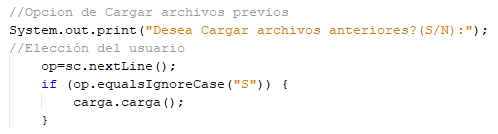
La versión de Neatbeans IDE que se utilizo es **8.2**.

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

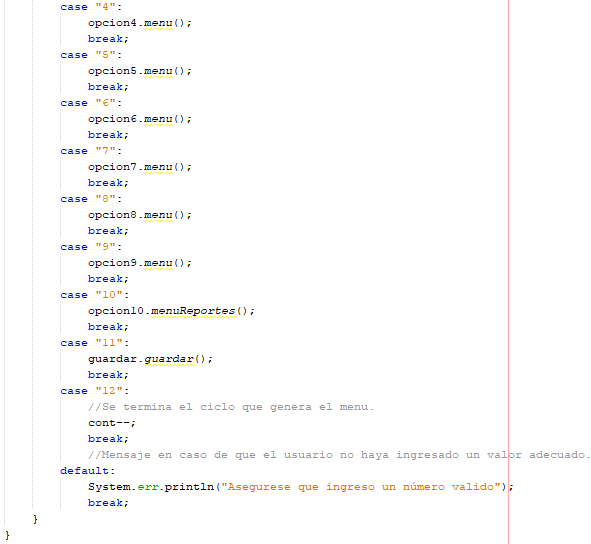
Clase Principal

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).

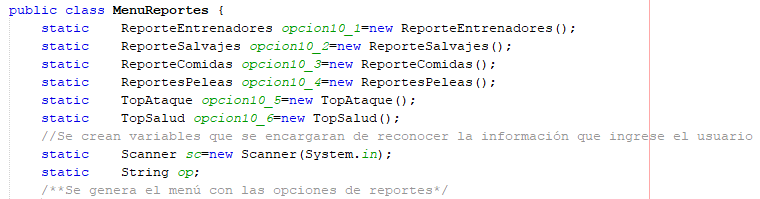


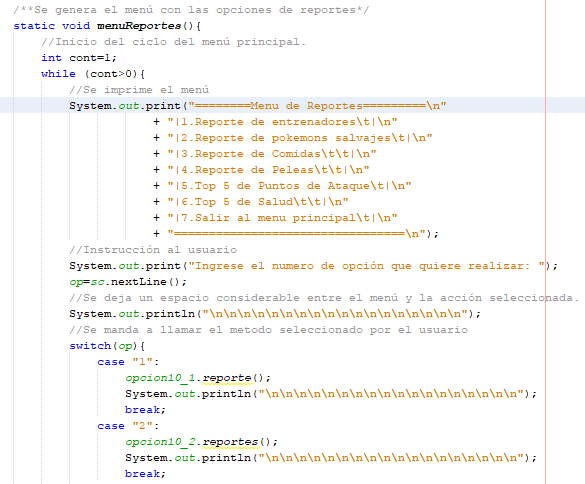
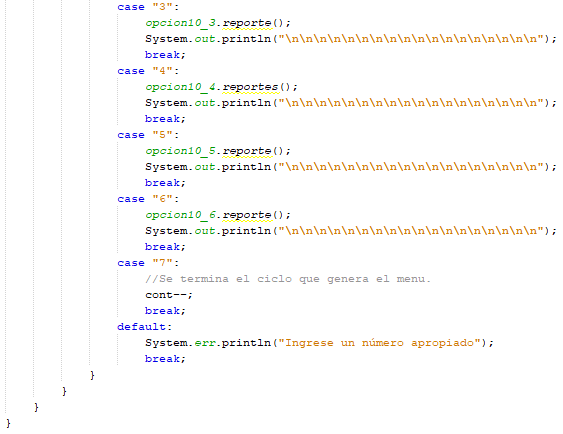






Si el usuario ingresa el número 10 se manda a llamar un metodo de la misma clase que imprime un submenú, este submenú es para la generación de reportes, este metodo se llama **menuReportes()**, este mismo metodo se encarga de mandar a llamar las clases que generan los reportes HTML.

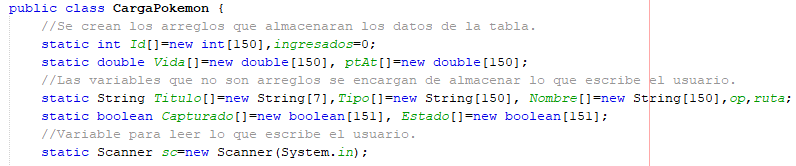


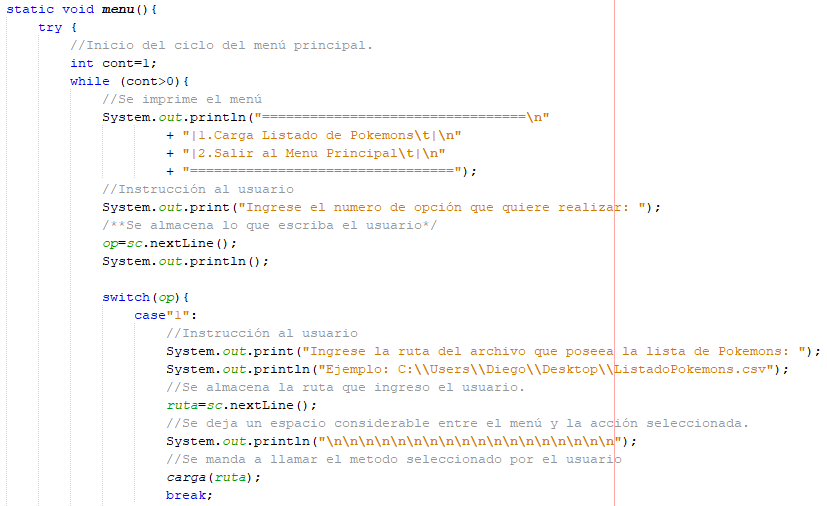
Clase CargaPokemon

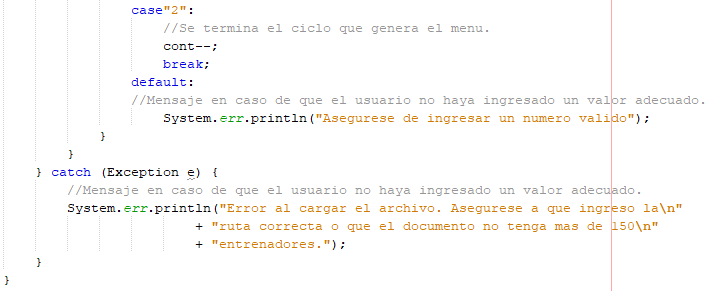
Esta clase posee los metodos necesarios para leer documentos csv que poseean un listado de los pokemons con los que se interectuara mientras el programa se mantenga encendido.

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.

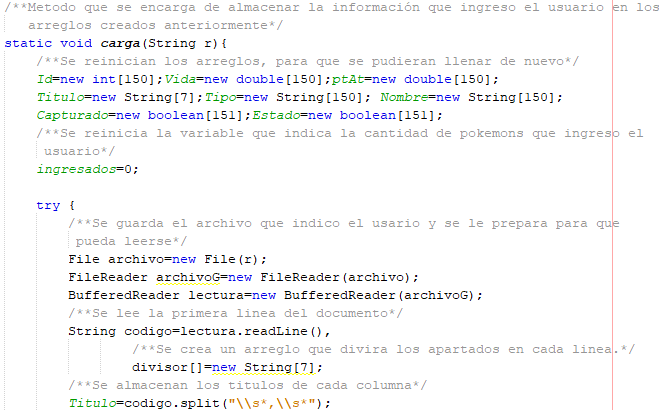


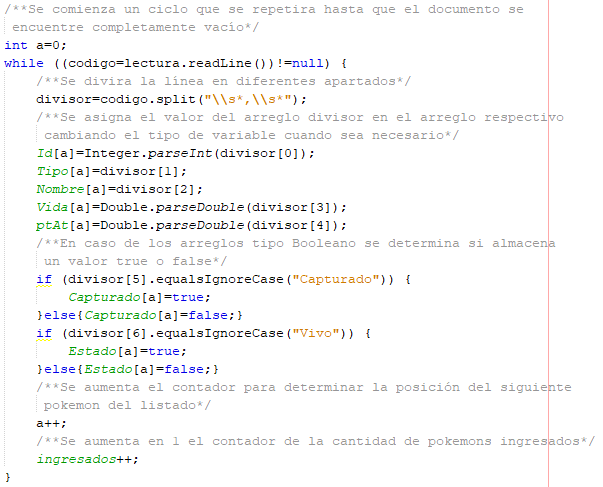
El primer metodo que la clase inicializa es **menu()**, este metodo existe en todos los metodos de *carga*. Este metodo se encarga de imprimir un submenú que muestra las dos acciones que puede realizar la clase que son leer un documento csv con el lista de pokemons y el regresar al menú principal.

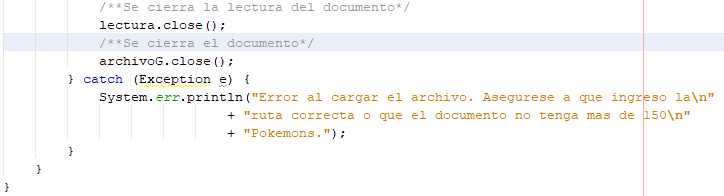




Si el usuario ingresa el número 2 se iniciara el metodo **carga(String r)**, este metodo se encarga de leer el documento y almacenar los datos del documento tanto en el arreglo que se encarga de almacenar los títulos y los demas arreglos que almacenan los datos de los pokemons. Estos arreglos se almacenan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





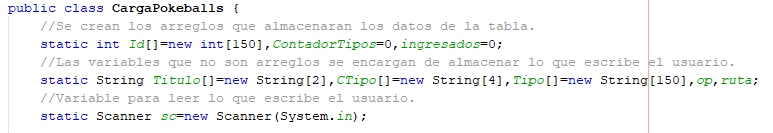


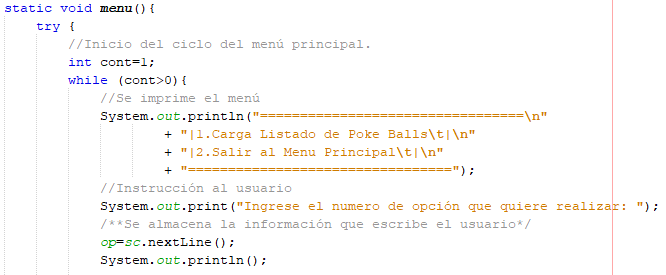
En las siguientes clases de *carga* no se volveran a explicar a profundidad los metodos de **menu** y **carga** ya que funcionan de manera parecida, diferenciandose unicamente en el texto que imprimen (en el caso de **menu**) y en la cantidad de arreglos (en el caso de **carga**). Por lo que solo se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

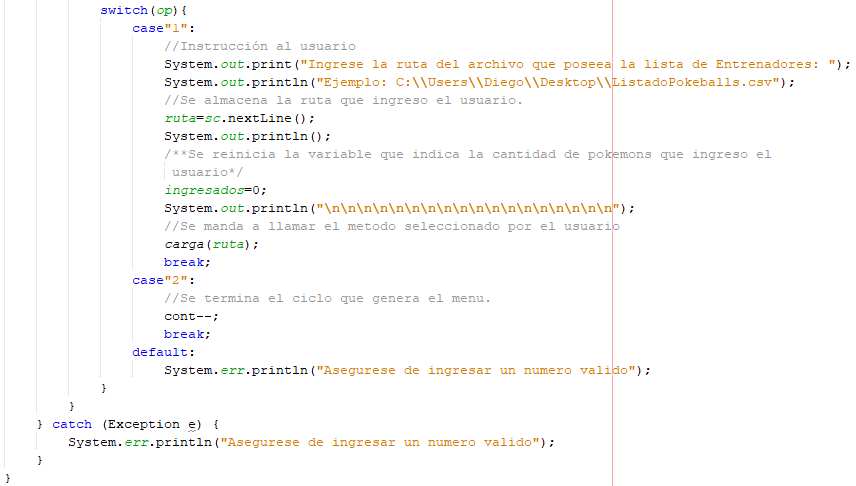
Clase CargaPokeballs

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.

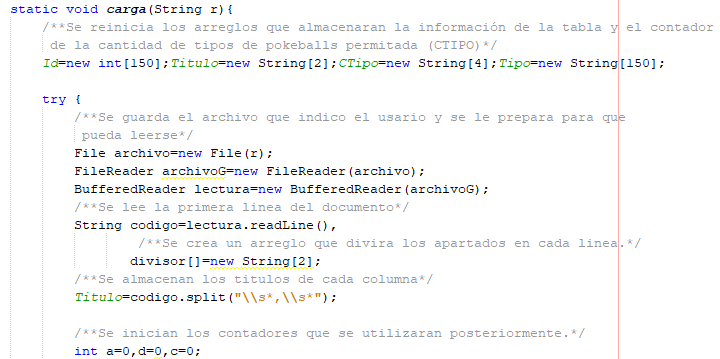
Primero se crean los arreglos que se encargaran de almacenar los datos de todas las columnas de la tabla del documento csv que ingrese el usuario. Y otras variables que serviran como contador.

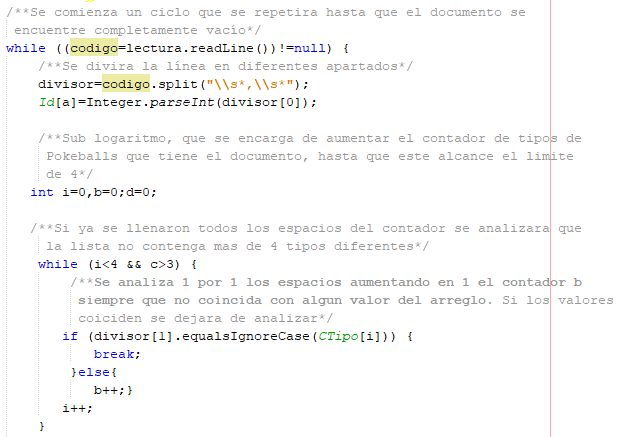
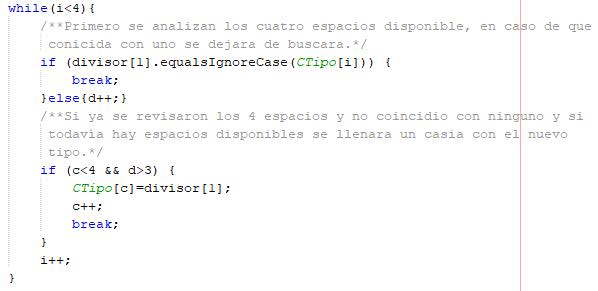


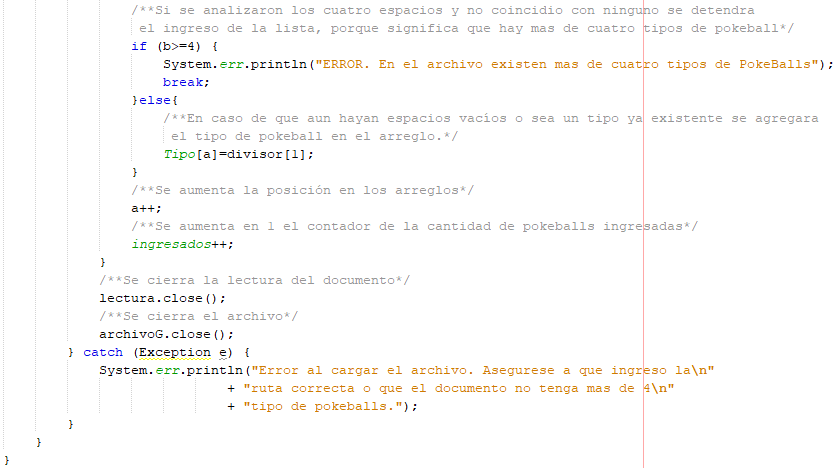




Este metodo **carga(String r)** se diferencia a los demas en que tiene un ciclo while que se encarga de verificar que solo existan 4 tipos de pokeball. Funciona almacenando cada nuevo tipo en un arreglo de 4 posiciones que se encarga de almacenar los tipos de pokeball, si se llegara a encontrar un nuevo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



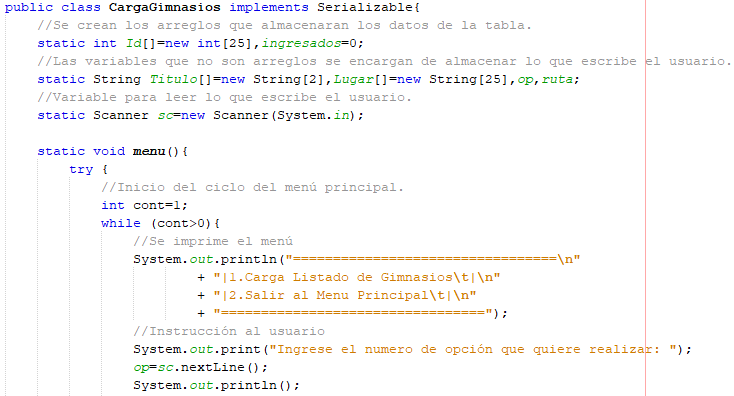
 

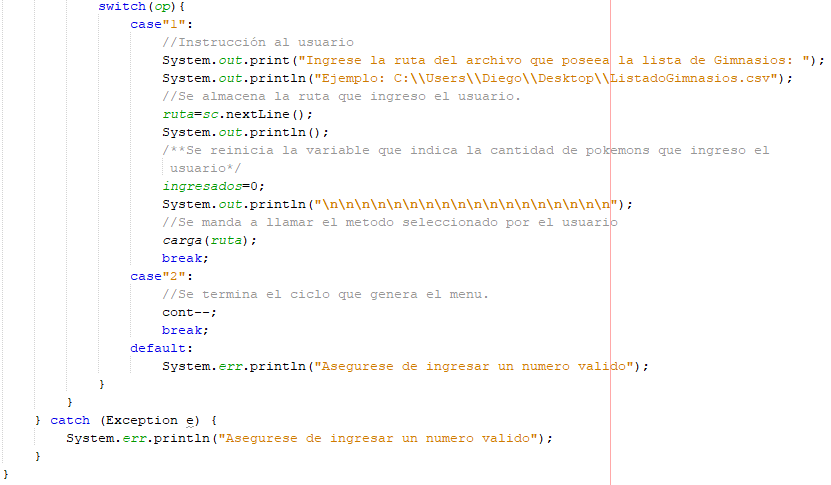


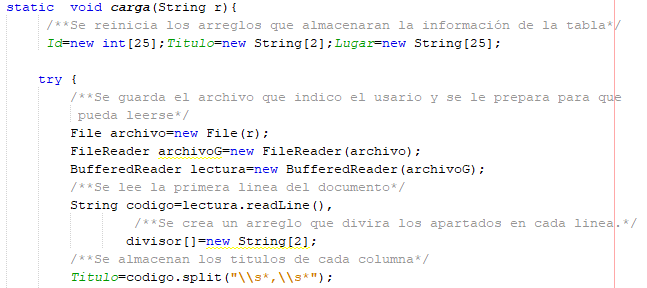
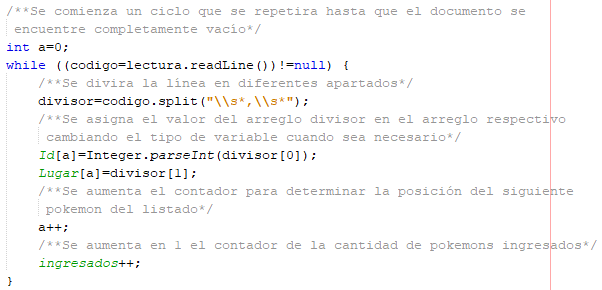
Clase CargaGimnasios

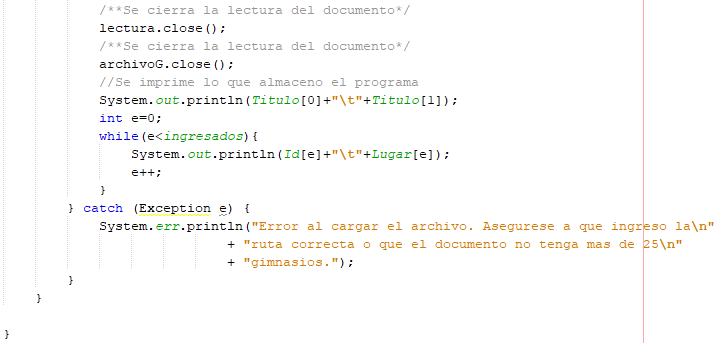
Esta clase posee los metodos necesarios para leer documentos csv que poseean un listado de los gimnasios con los que se interectuara mientras el programa se mantenga encendido.

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.





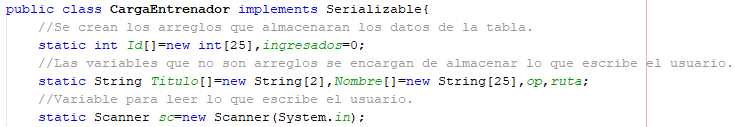
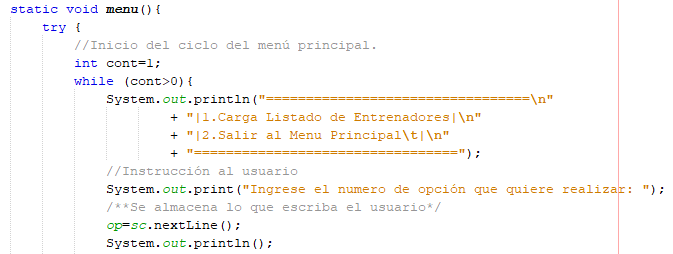
 

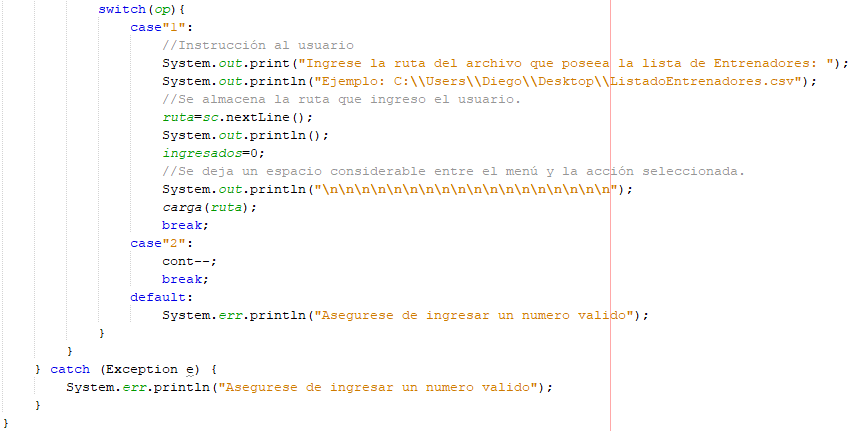


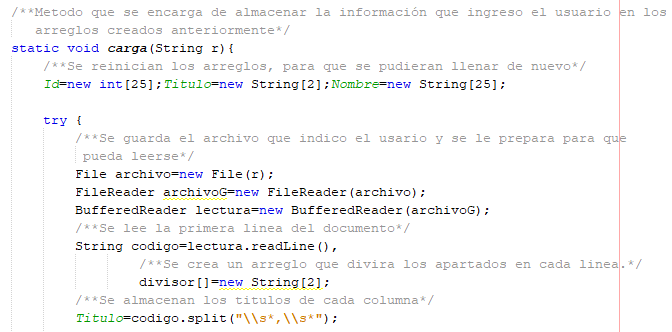
Clase CargaEntrenador

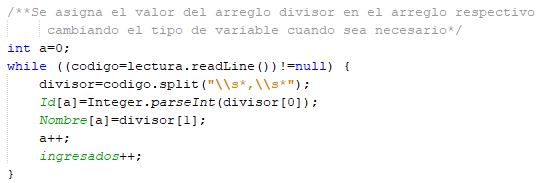
Esta clase posee los metodos necesarios para leer documentos csv que poseean un listado de los Entrenadores con los que se interectuara mientras el programa se mantenga encendido.

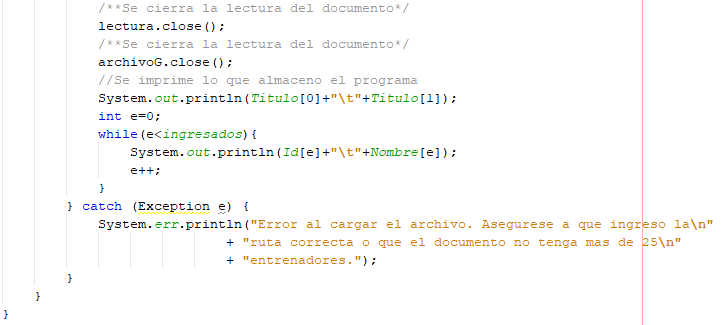
Primero se crean los arreglos que se encargaran de almacenar los datos de todas las columnas de la tabla del documento csv que ingrese el usuario. Y otras variables que serviran como contador.





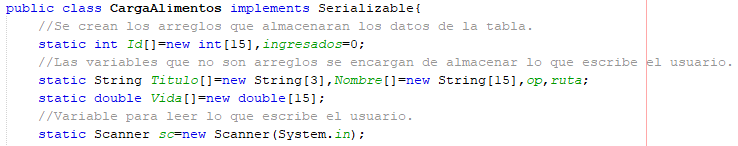


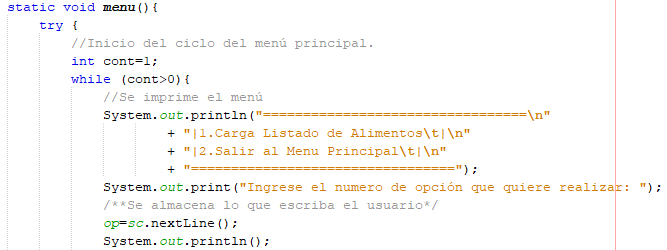


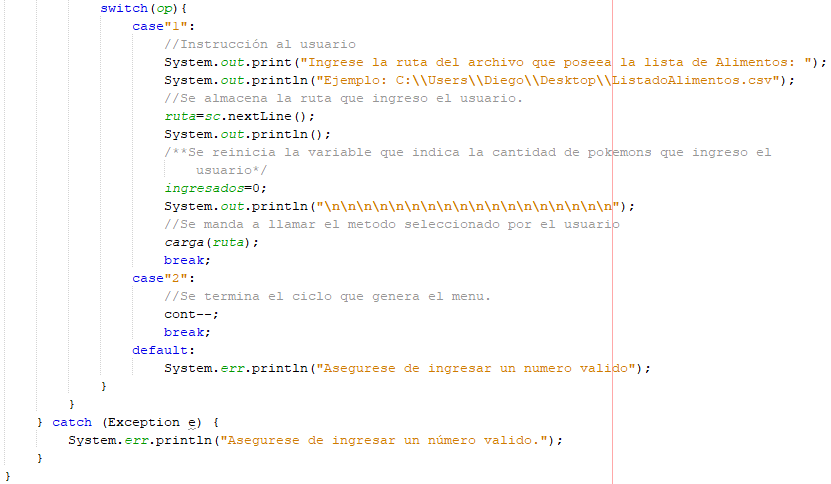
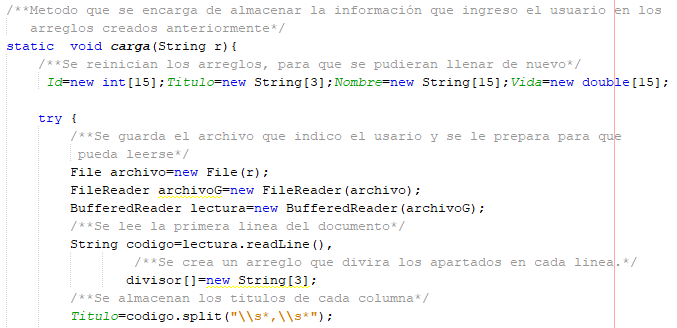
Clase CargaAlimentos

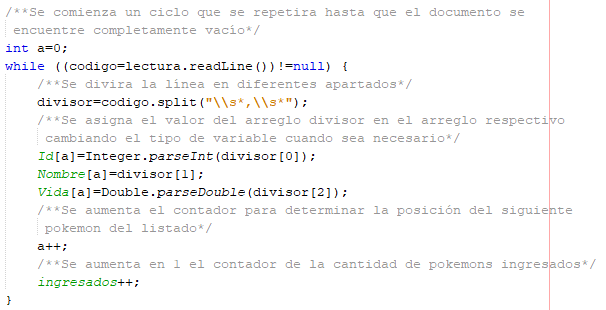
Esta clase posee los metodos necesarios para leer documentos csv que poseean un listado de los Alimentos con los que se interectuara mientras el programa se mantenga encendido.

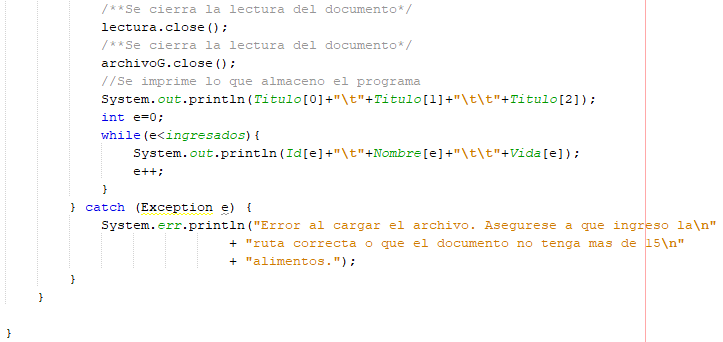
Primero se crean los arreglos que se encargaran de almacenar los datos de todas las columnas de la tabla del documento csv que ingrese el usuario. Y otras variables que serviran como contador.





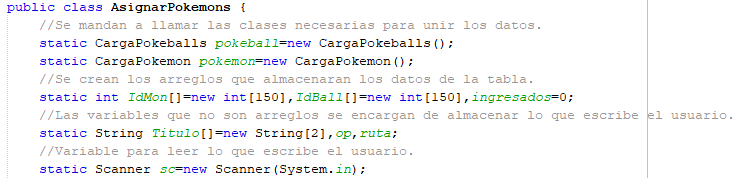


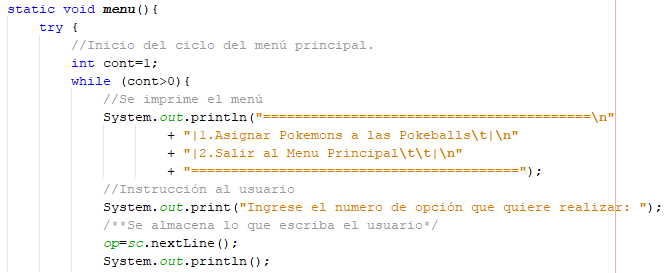


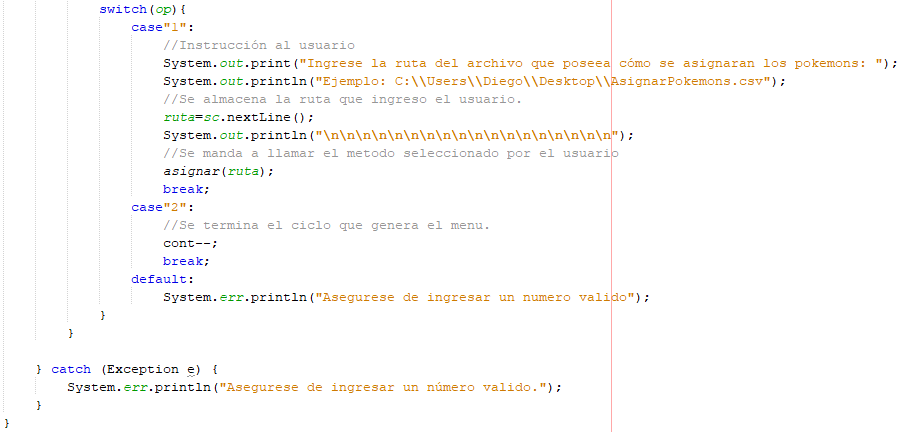
Clase AsignarPokemons

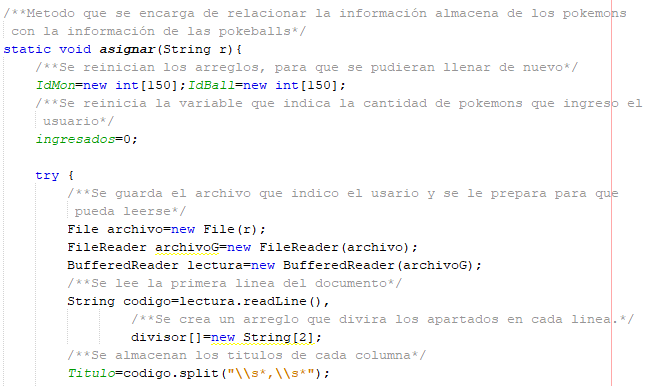
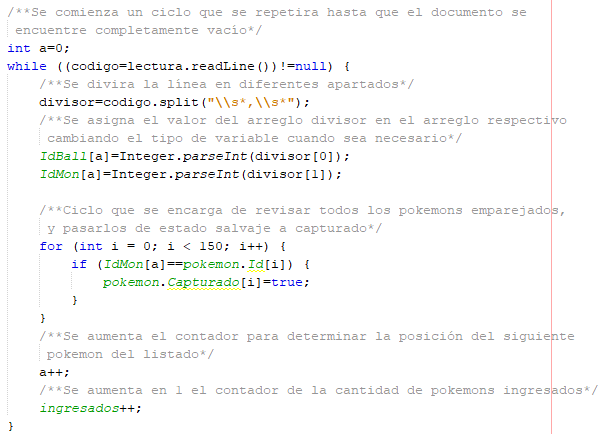
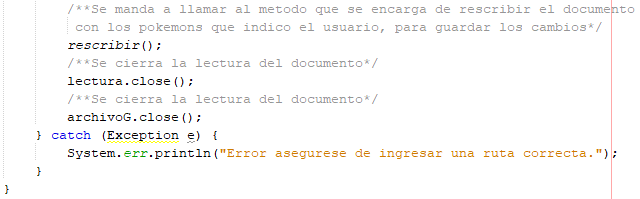
Esta clase se encarga de relacionar a un pokemon con una pokeball, para lograr esto se mandan a llamar a los arreglos de las clases *CargaPokemon* y *CargaPokeballs*, y mediante ciclos encontrar que pokemons se relacionan con que pokeballs.

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.

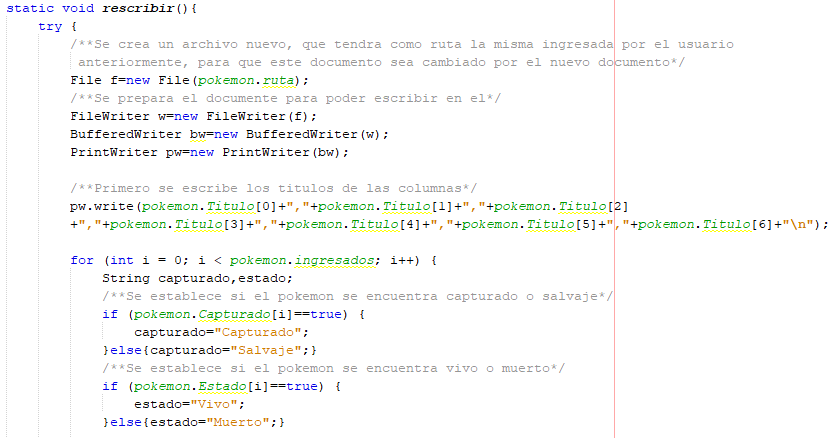
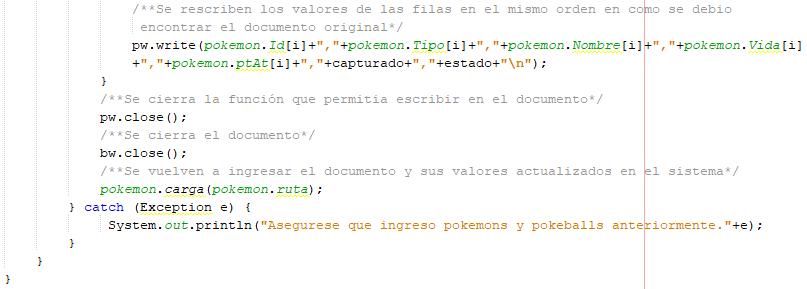






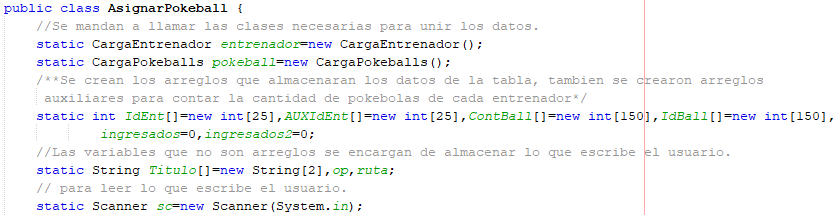
  

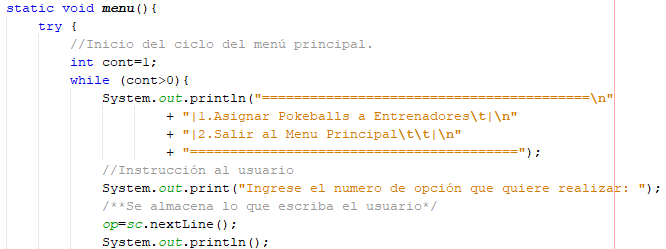
Dentro de las clases *Asignar* se encuentra un metodo que se encarga de reescribir los documentos csv ingresados originalmente por el usuario en las clases *Carga* llamadas por la clase *Asignar* en cuestion.

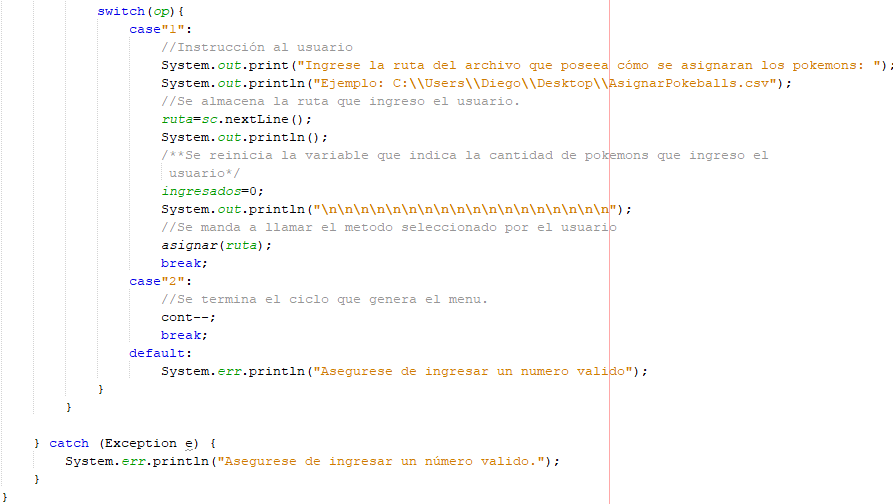
 

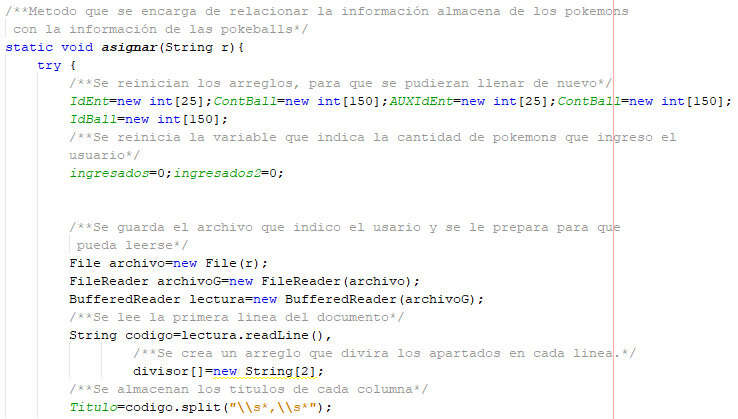
Clase AsignarPokeball

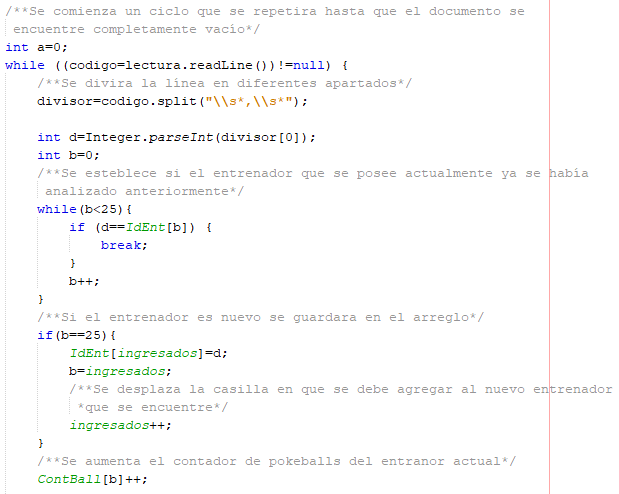
Esta clase se encarga de relacionar hasta 5 pokeballs con un entrenador, para lograr esto se mandan a llamar a los arreglos de las clases *CargaEntrenador* y *CargaPokeballs*, y mediante ciclos encontrar que pokeballs se relacionan con que entrenadores.

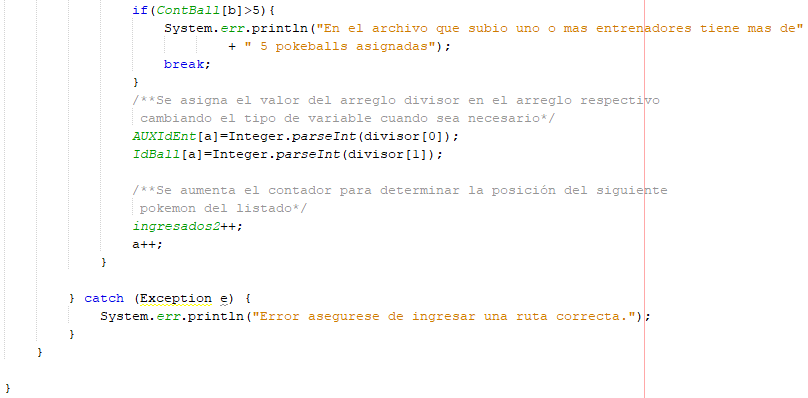








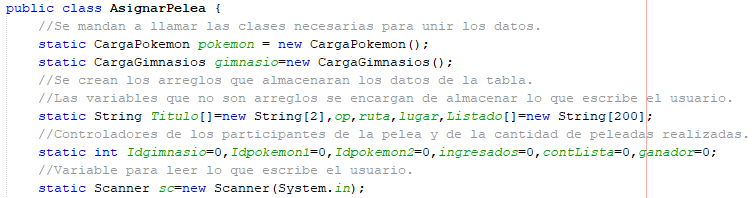
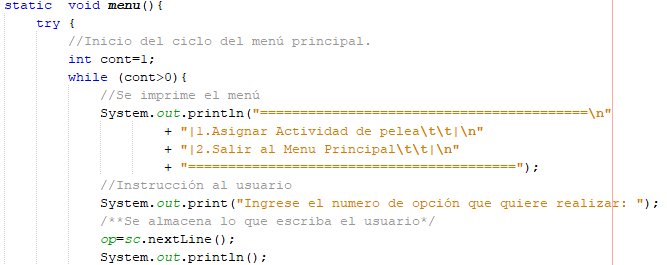


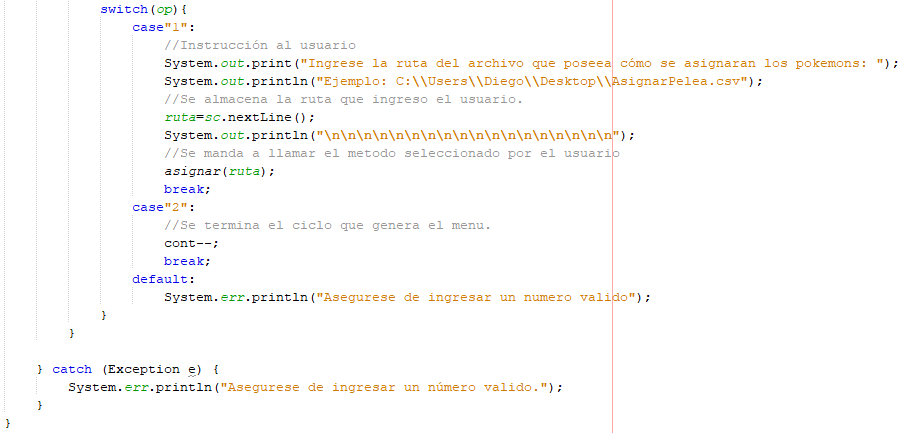


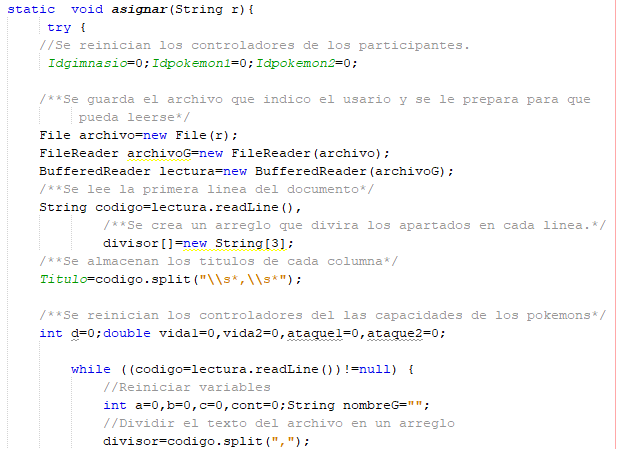
Clase AsignarPelea

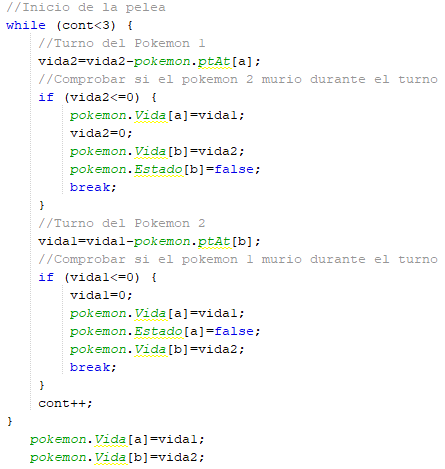
Esta clase se encarga de relacionar pokemons y hacer que peleen, para lograr esto se mandan a llamar a los arreglos de las clases *CargaPokemon* y *CargaGimnasios*.

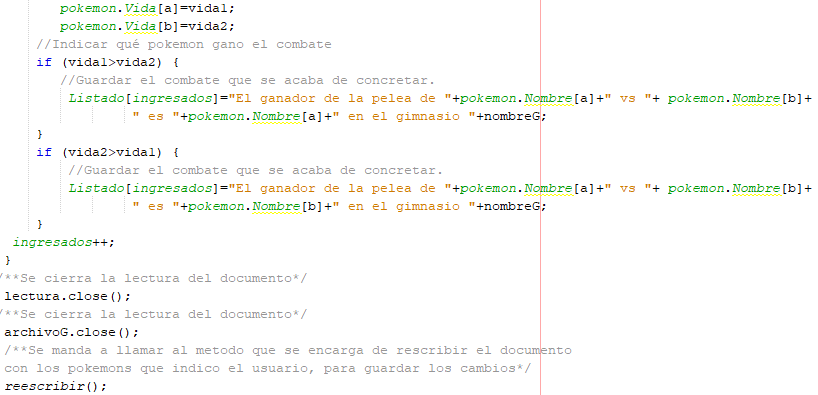
Relacionar esta información provoca que la vida de los pokemons se reduzca debido a los ataques de sus contrincantes, incluso provocando que la vida de los pokemons llegue a 0, en estos casos el estado de vivo se pasa a muerto.

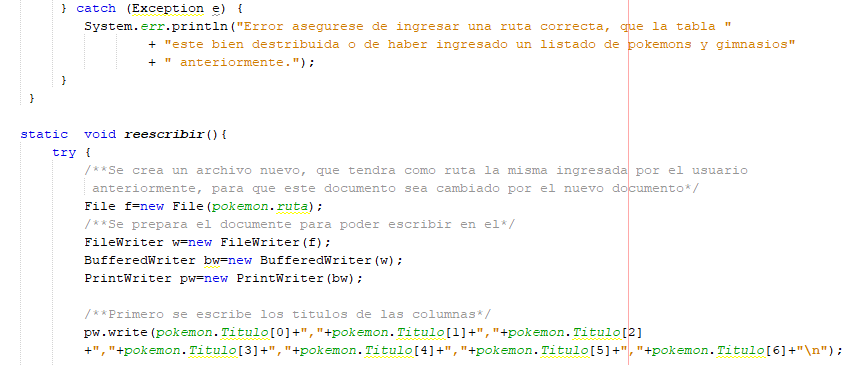
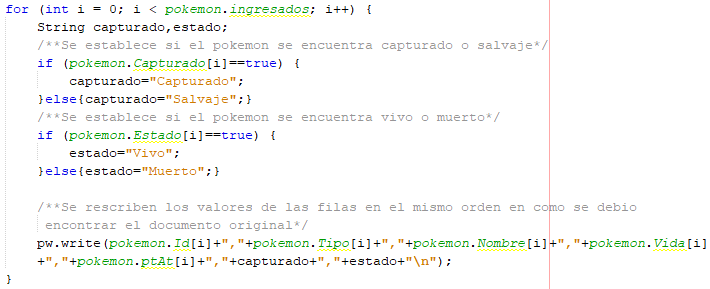
 

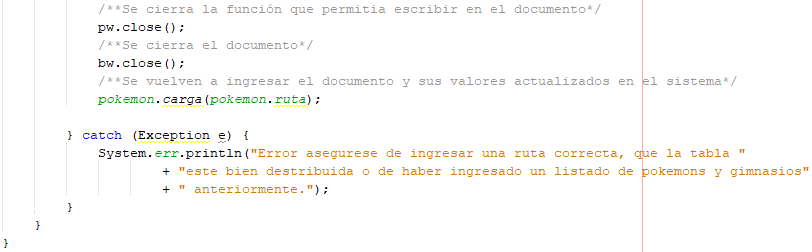






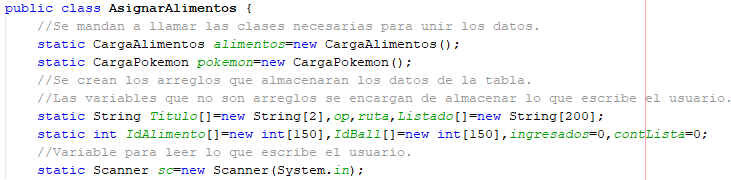
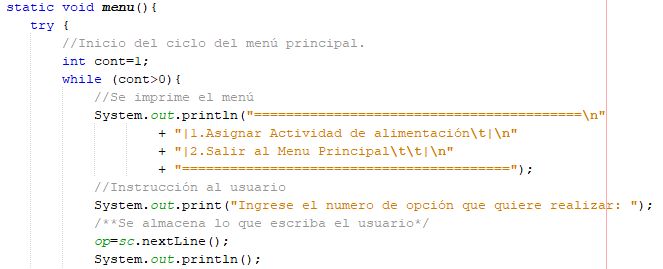
 

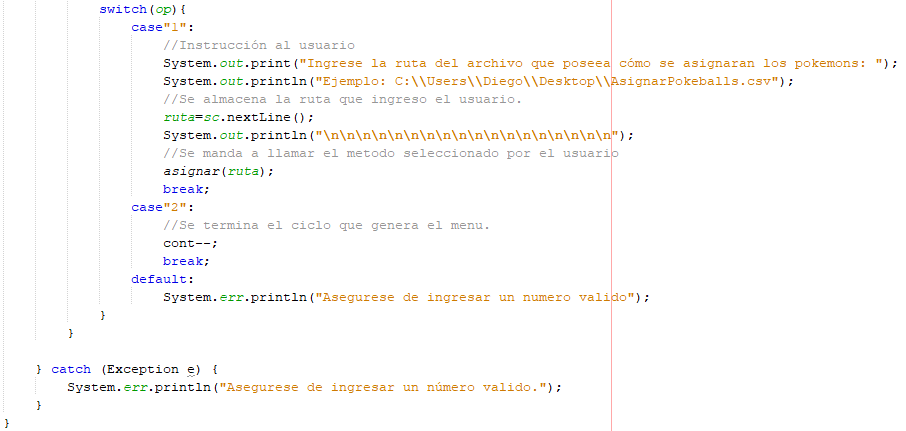


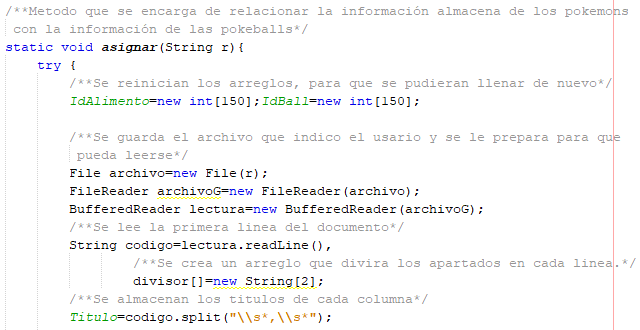
Clase AsignarAlimentos

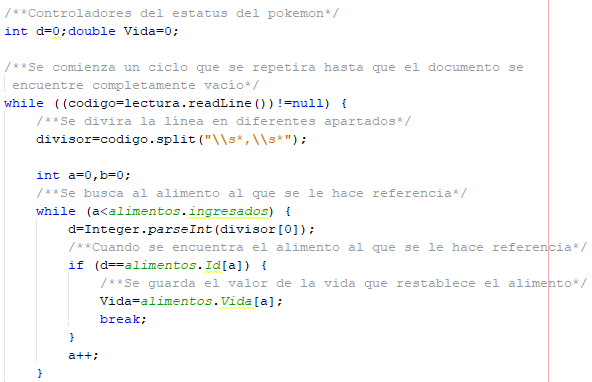
Esta clase se encarga de relacionar a un pokemon con uno o varios alimentos que consumira, para lograr esto se mandan a llamar a los arreglos de las clases *CargaPokemon* y *CargaAlimentos*.

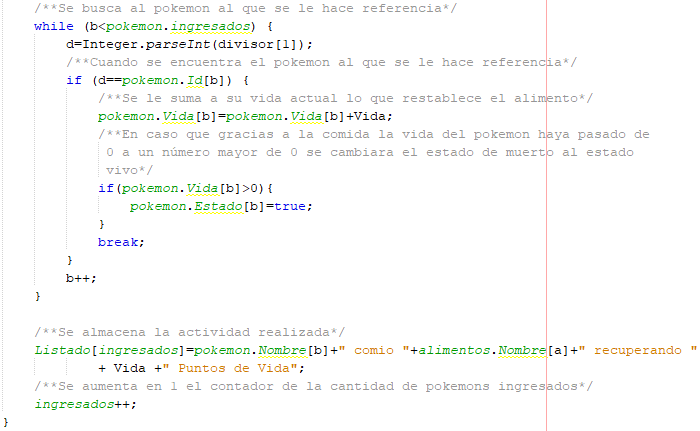
Relacionar esta información provoca que la vida de los pokemons aumente, incluso provocando que la vida de los pokemons pase de valer 0 a un número mayor, en estos casos el estado de muerto se pasa a vivo.

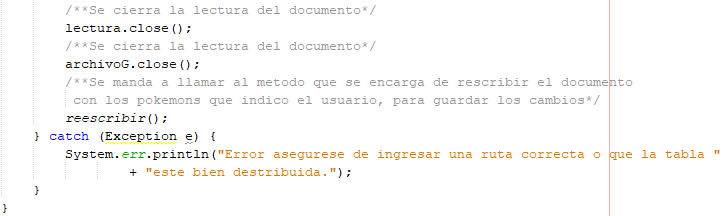
 

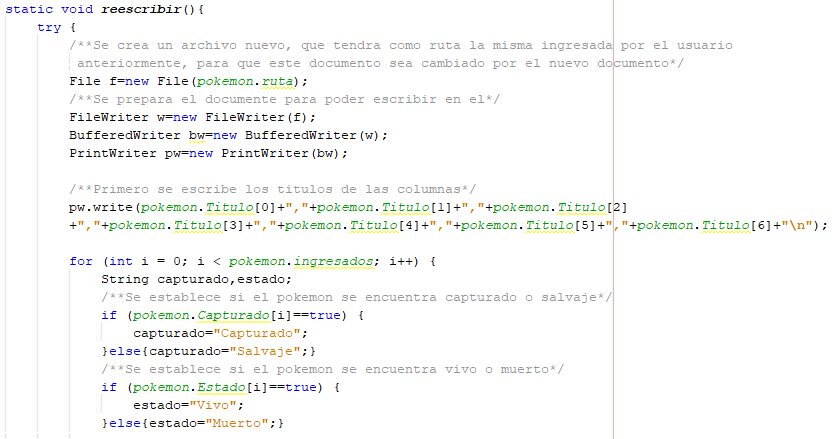


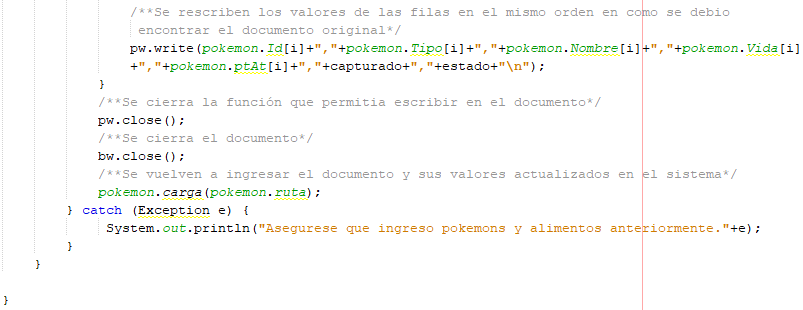






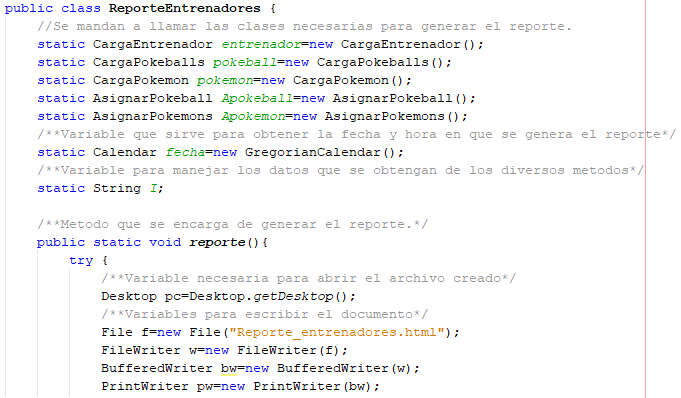


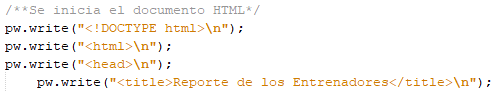


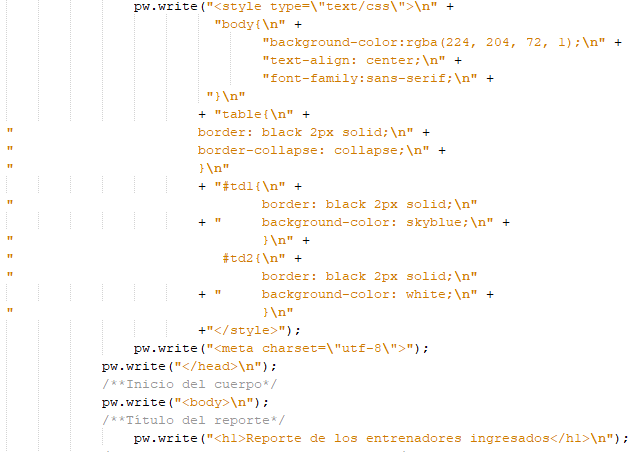


Clase ReporteEntrenadores

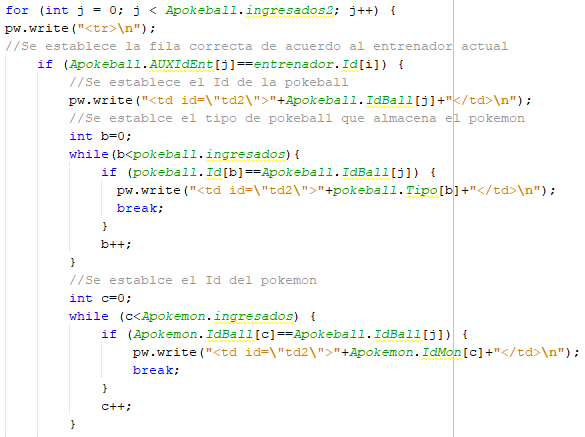
Esta clase se encarga de crear un documento HTML que muestra a todos los entrenadores *cargados* y los pokemons que se le *asignaron* para posteriormente abrirse automáticamente.



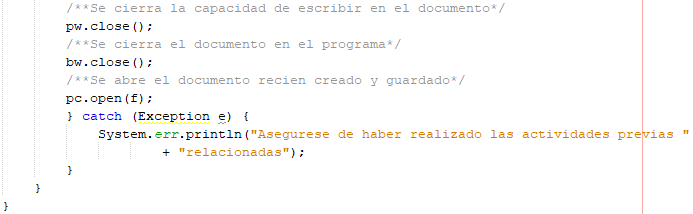






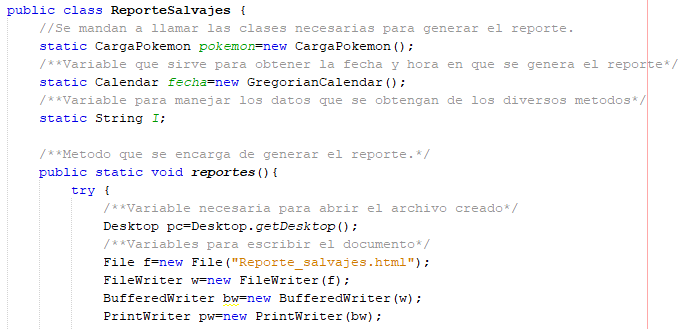


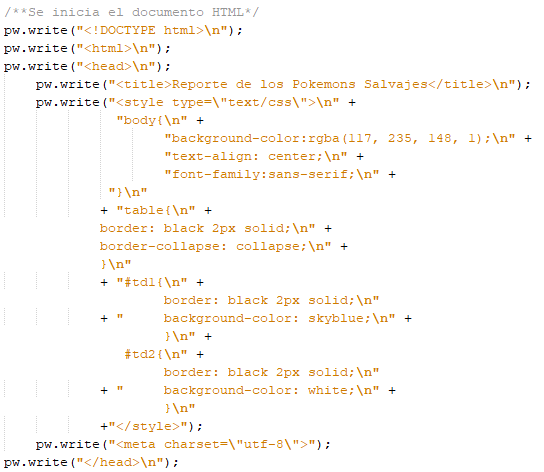


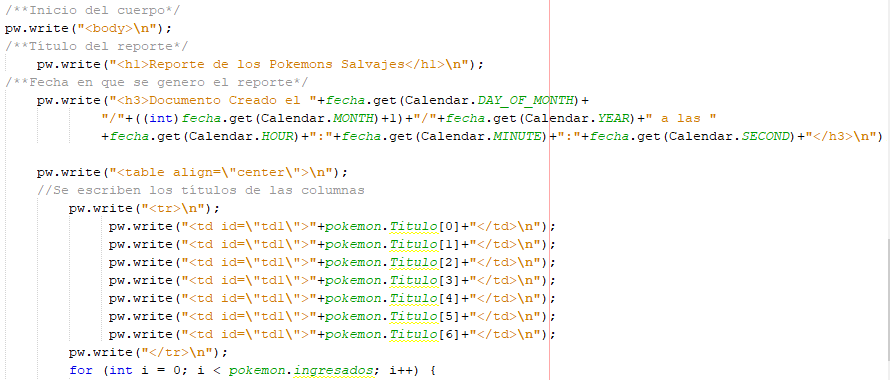


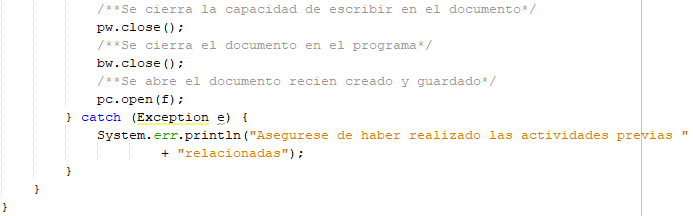
Clase ReporteSalvajes

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



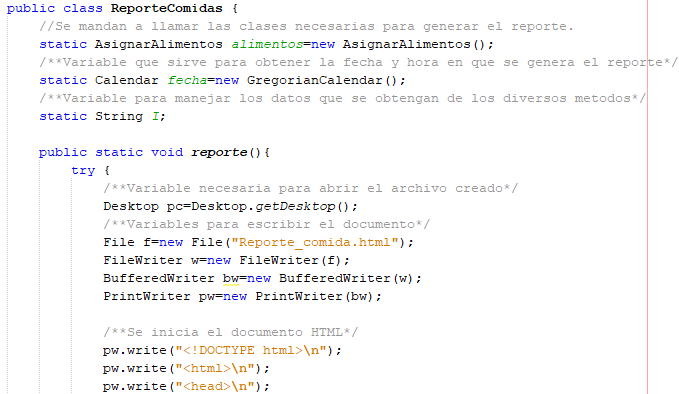


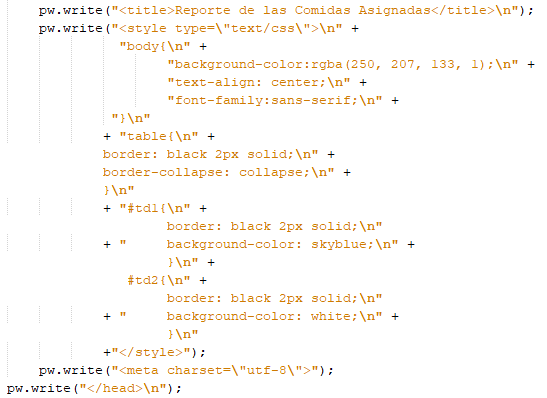


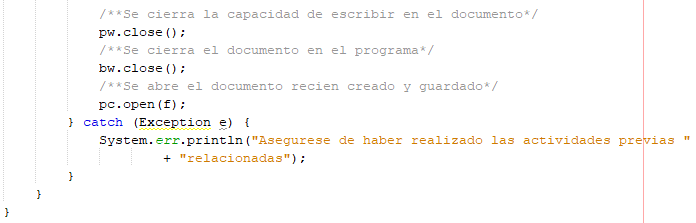
Clase ReporteComidas

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.



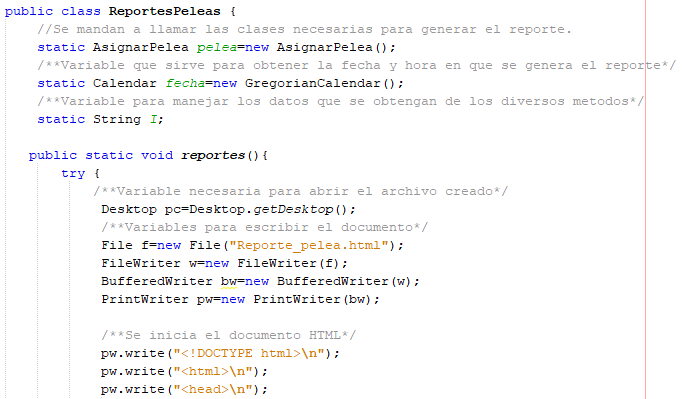






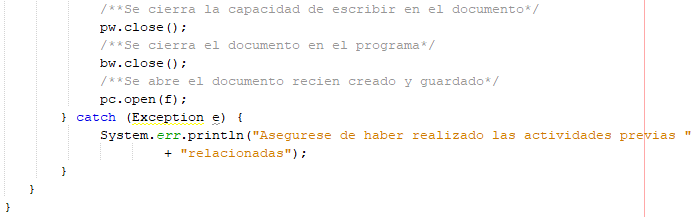
Clase ReportesPeleas

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.



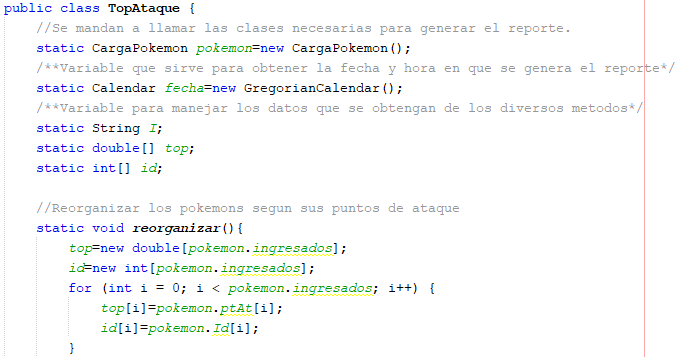


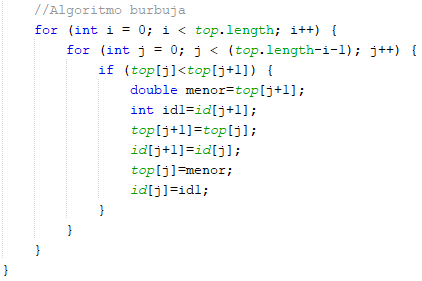


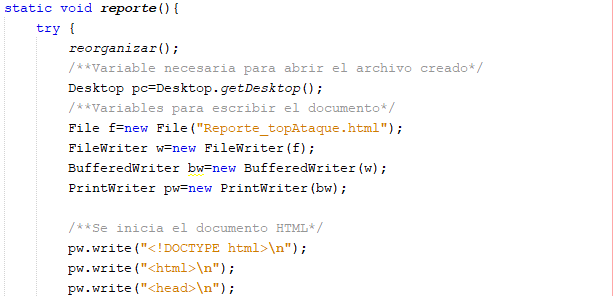


Clase TopAtaque

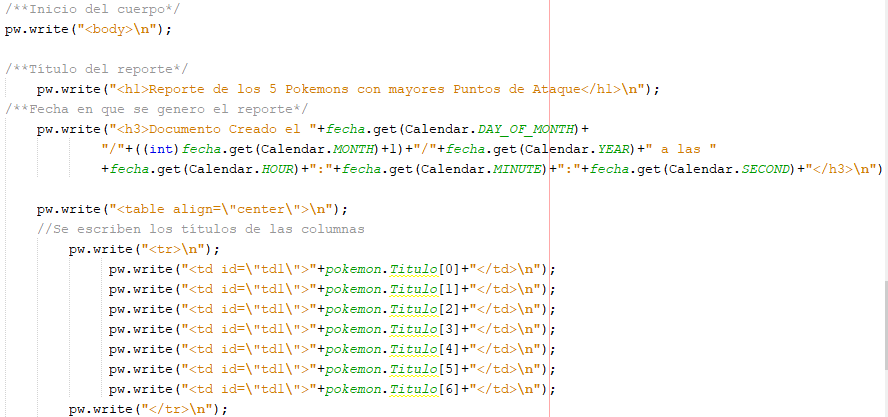
Esta clase se encarga de mostrar de forma descendente los 5 pokemons con mas puntos de ataque. Antes de generar el reporte hay un método que primero se encarga de acomodar los puntos de ataque de mayor a menor mediante el algoritmo de burbuja.



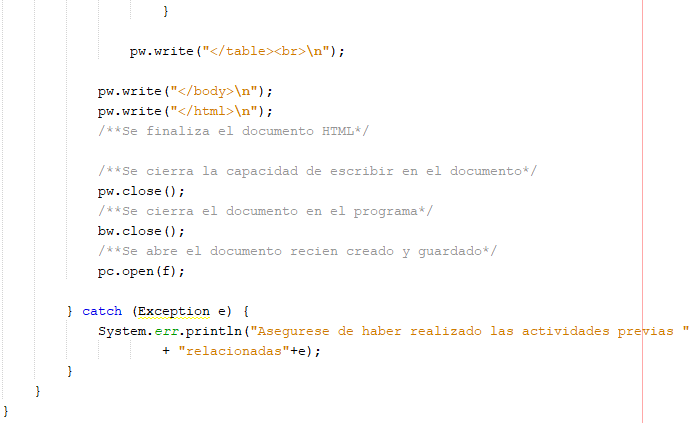






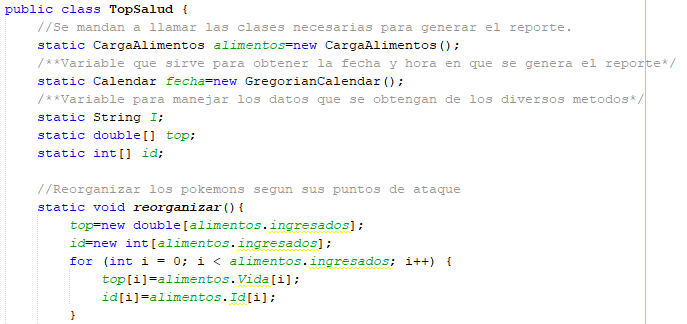


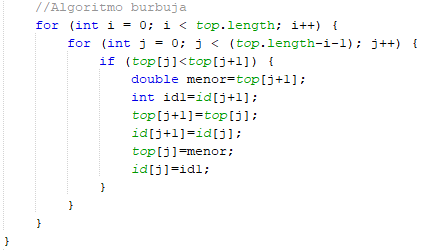


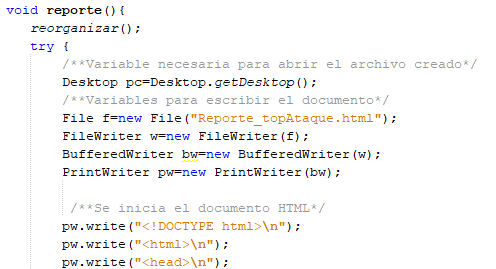
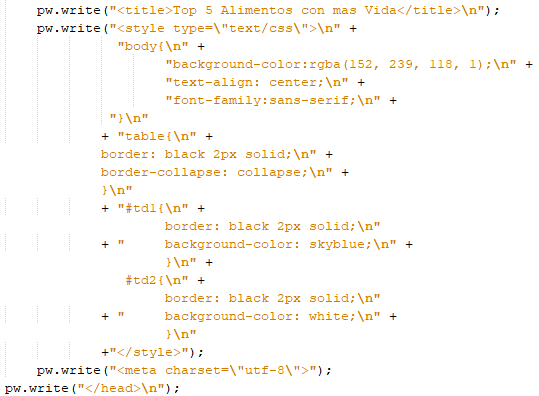


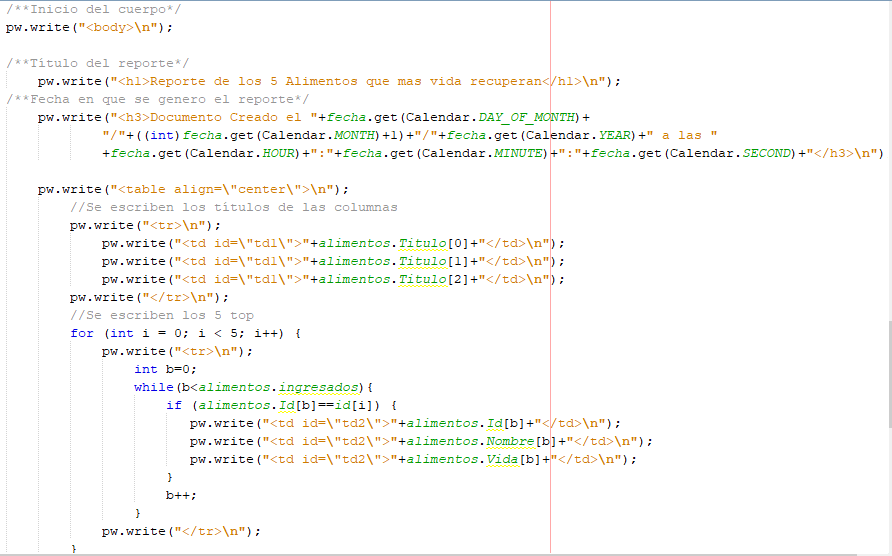
Clase TopSalud

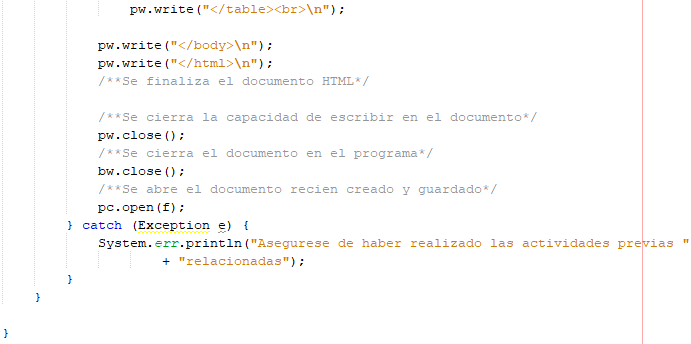
Esta clase se encarga de mostrar de forma descendente los 5 alimentos con más puntos de vida que recupera. Antes de generar el reporte hay un método que primero se encarga de acomodar los puntos de vida de mayor a menor mediante el algoritmo de burbuja.





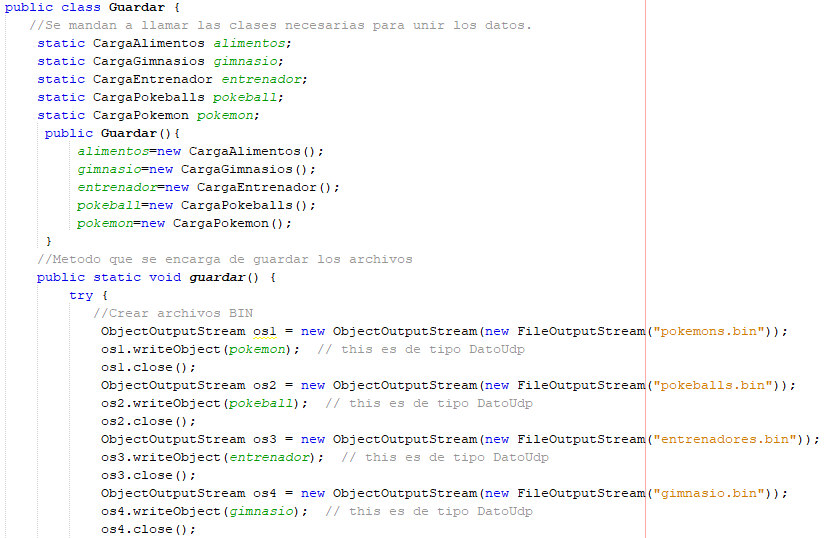


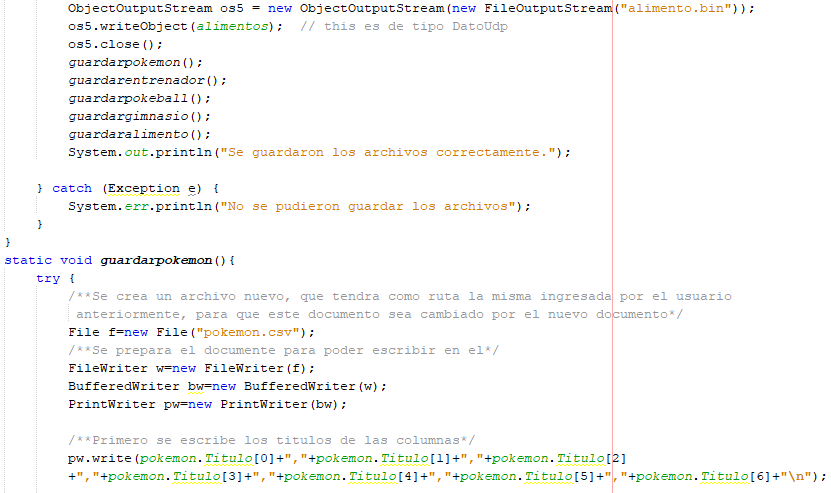


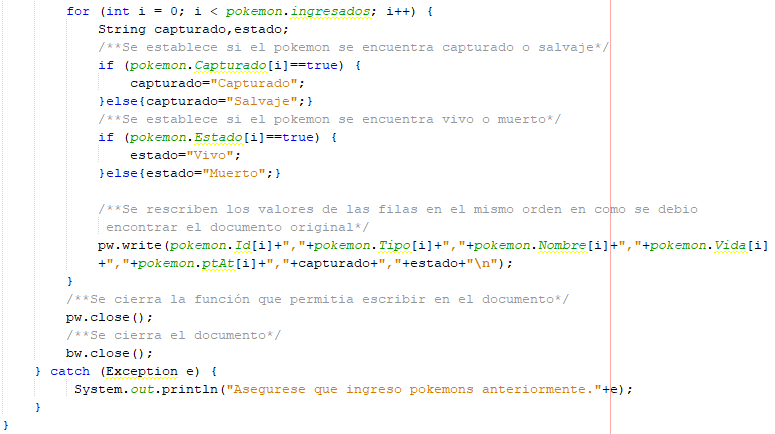
Clase Guardar

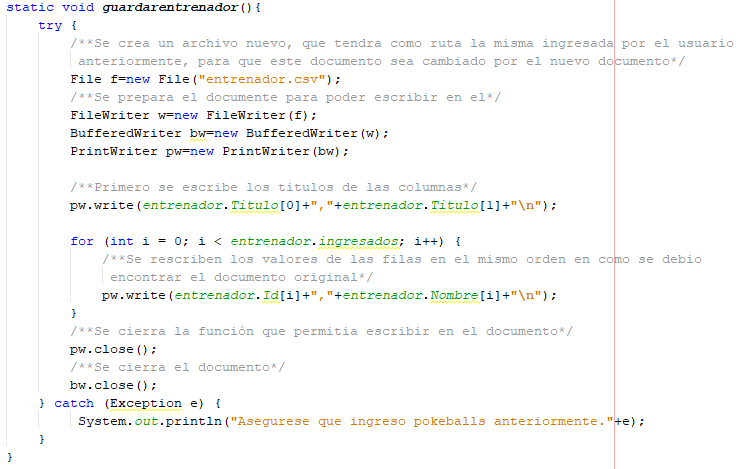
Esta clase se encarga de guardar en el sistema la información que almacenan las clases de *carga* de tal forma que después pueda ser tomada por el programa sin necesidad de tener que volver a escribir las rutas de los archivos.

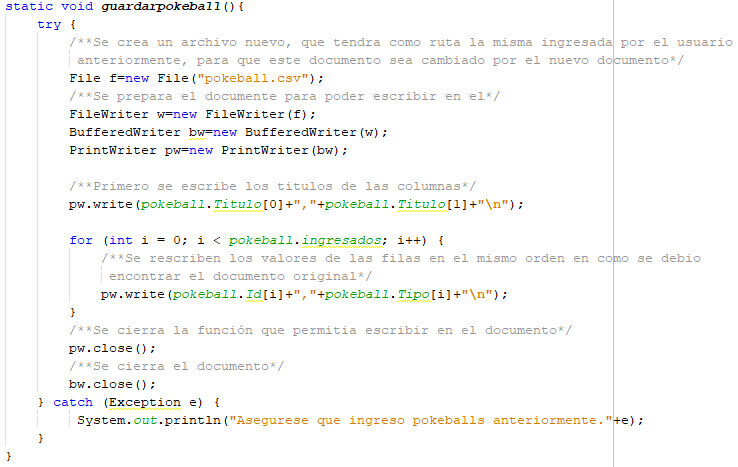
La información la guarda de dos maneras, mediante archivos *.bin* y archivos *.csv*. Estos archivos de guardado se almacenan en la carpeta que se encuentre el archivo .jar que dé inicio al programa.

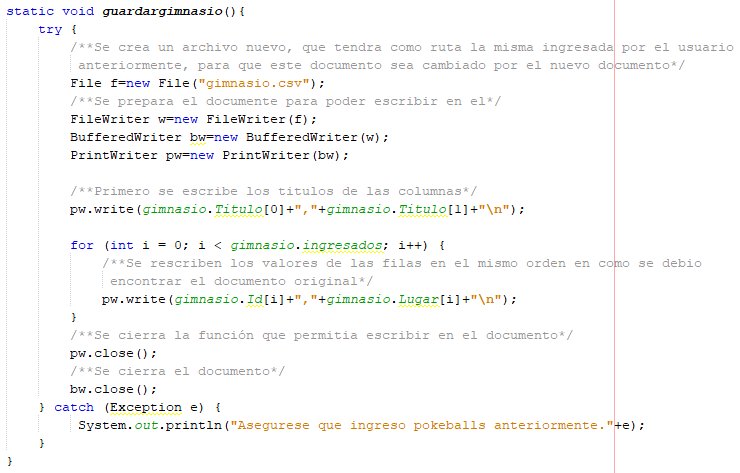


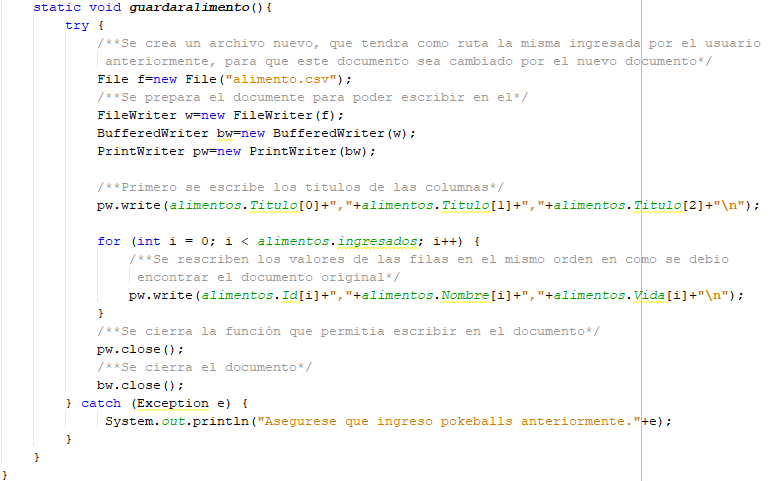






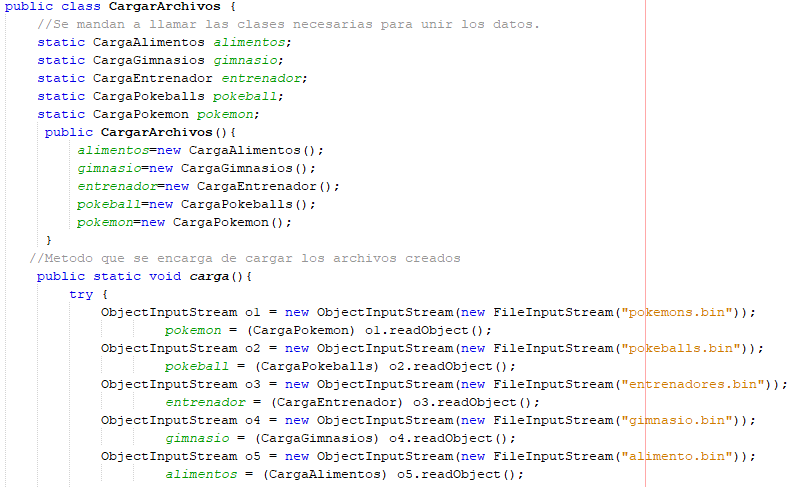


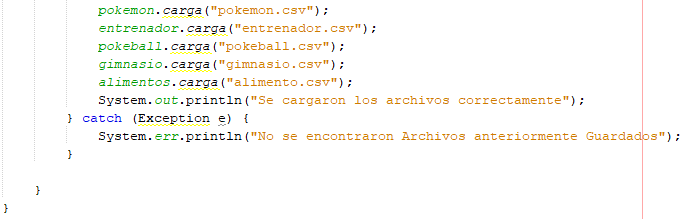




Clase CargarArchivo

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.





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](https://es.wikipedia.org/wiki/Sistema_de_numeraci%C3%B3n) [binario](https://es.wikipedia.org/wiki/Sistema_binario). La capacidad de almacenamiento de una memoria digital también se mide en bit. es la unidad mínima de información empleada en [informática](https://es.wikipedia.org/wiki/Inform%C3%A1tica), en cualquier dispositivo digital, o en la [teoría de la información](https://es.wikipedia.org/wiki/Teor%C3%ADa_de_la_informaci%C3%B3n). 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](https://es.wikipedia.org/wiki/Objeto_(programaci%C3%B3n_orientada_a_objetos)) en un medio de almacenamiento (como puede ser un [archivo](https://es.wikipedia.org/wiki/Archivo_inform%C3%A1tico), o un buffer de memoria) con el fin de transmitirlo a través de una conexión en red como una serie de [bytes](https://es.wikipedia.org/wiki/Byte) 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.