# Examen Parcial I

# DCA – 2020-2

La fundación llamada **Peludos sin hogar** se encarga de recoger perros abandonados,

tratarlos y ponerlos en adopción.

La fundación ha hecho una labor impresionante, ha logrado salvar una gran cantidad de

perros, sin embargo, se ha vuelto un tema crítico de manejar. Por esta razón la fundación

ha decidido desarrollar un aplicativo de escritorio que les ayude a:

1. Cargar la información de sus peludos, los cuales tiene los siguientes atributos: id

numérico, nombre, edad, raza y fecha de nacimiento. La información de los perros

se encuentra en dos archivos .txt **. El archivo uno contiene el id, nombre y edad del**

**perro. El archivo dos tiene id, raza y fecha de nacimiento del perro, deben tener**

**cuidado con los id no están en el mismo orden.**

1. Se desea que el orden natural de la lista se dé por el id de cada perro.
2. El programa debe guardar un archivo .txt con la lista de los perros ordenados por

cada atributo (un archivo con los perros ordenados por nombre, otro archivo

ordenados por id y así por cada atributo). El nombre y la raza de los perros debe

quedar en minúscula.

1. El programa debe tener una interfaz gráfica la cual muestre la información de los

txt, La cual debe ordenarse según la interacción que ustedes escojan (presionando

teclas, botones, etc) es libre de cada desarrollador, sin embargo, debe tener

instrucciones de cómo organizar la lista por cada atributo.

**Bonus**

1. Al seleccionar dos elementos de la lista se intercambia de posición.

Instrucciones finales

* Para cargar los archivos están en la carpeta **data/imports** y para exportar los archivos **data/exports**
* Descargar la base del código con los documentos .txt y el core de processing:

**https://github.com/felipegarcial/DCAParcialUno**

**Rúbrica**

1. **Análisis y diseño de software (30%)**
2. Contexto 10%
3. Requerimientos funcionales 35%
4. Requerimientos no funcionales 15%
5. Diagrama de clases (tener en cuenta arquitectura MVC). 40%
6. **Implementación del código (70%)**
7. Correcta implementación del diagrama de clases.5%
8. Correcta utilización de la lista adecuada según los requerimientos. 5%
9. Correcta carga de información 15%
10. Correcto guardado de información 10%
11. Correcto manejo de cadenas de texto 20%
12. Correcta implementación del ordenamiento de la lista de mascotas de manera natural y parcial. 40%
13. Por id
14. Por nombre
15. Por edad
16. Por raza
17. Calidad y orden en el código 5%

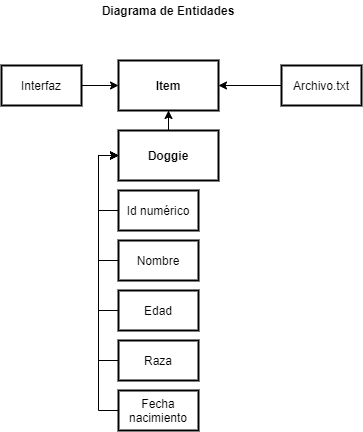
# Solución

**Contexto:**

* La fundación “Peludos sin hogar” necesita un aplicativo de escritorio que les ayude guardar y ordenar una lista la cual está dividida en dos archivos. txt que contiene los datos de los perros los cuales son su id, nombre, raza, edad y fecha de nacimiento. El aplicativo deberá guardar los datos ordenados en archivos. txt por cada atributo.

**Entidades:**

1. **Perros/Peludos/Doggie**
   1. Id numérico.
   2. Nombre.
   3. Edad.
   4. Raza.
   5. Fecha de nacimiento.
2. **Archivo. txt.**
3. **Interfaz.**

****

**RF1:** Cargar archivos.txt con la lista de perros

|  |  |
| --- | --- |
| RF1 | |
| **Descripción:** | Se debe cargar un archivo.txt que contiene los datos de los caninos. |
| **Entradas:** | Datos de los archivos txt. |
| **Salidas:** | El archivo estará cargado en el programa. |
| **PreCondiciones:** | El archivo debe de existir. |
| **Post-Condiciones:** | El archivo.txt debe poder ser llamado y editado. |

**RF2:** Crear interfaz gráfica.

|  |  |
| --- | --- |
| RF2 | |
| **Descripción:** | El programa debe de tener una interfaz gráfica para poder pintar la lista de caninos del archivo.txt. |
| **Entradas:** | Elementos gráficos e imágenes |
| **Salidas:** | Se debe de haber comparado la lista de caninos descritos del archivo.txt |
| **PreCondiciones:** | archivo.txt debe de existir y debe de haber cargado la información. |
| **Post-Condiciones:** | El programa deberá mostrar los elementos descritos por el archivo.txt y se deben de ver la interfaz con estos en la pantalla. |

**RF3:** Crear clase DoggieList

|  |  |
| --- | --- |
| RF3 | |
| **Descripción:** | Se deberá crear la clase que contenga toda la lógica de la lista del txt, además aquí es donde se debe inicializar los archivos txt. |
| **Entradas:** | Archivo.txt |
| **Salidas:** | Se debe de haber pintado la lista de caninos descritos del archivo.txt |
| **PreCondiciones:** | archivo.txt debe de existir y debe de haber cargado la información. |
| **Post-Condiciones:** | El programa deberá mostrar los elementos descritos por el archivo.txt y se deben de ver la interfaz con estos en la pantalla. |

**RF3:** Crear clase Doggie

|  |  |
| --- | --- |
| RF3 | |
| **Descripción:** | Se deberá crear la clase Doggie que contendrá todos los setters y getters |
| **Entradas:** |  |
| **Salidas:** | Se deben de tener todos los métodos. |
| **PreCondiciones:** | Debe de existir la clase DoggieList |
| **Post-Condiciones:** | Se podrá efectuar los métodos de Sort y modificación de archivo txt. |

**RF4:** Ordenar lista de cachorros.

|  |  |
| --- | --- |
| RF4 | |
| **Descripción:** | Se debe poder ordenar los ítem sin ningún inconveniente con un método en la clase DoggieList. |
| **Entradas:** | Archivo. txt con los datos cargados en la clase |
| **Salidas:** | Se podrá usar la lista de cachorros para ordenar los datos del txt. |
| **PreCondiciones:** | Debe existir la clase DoggieList y el archivo.txt con los datos. |
| **Post-Condiciones:** | Se debe poder usar los datos y modificarlos. |

**RF5:** Guardar archivo txt.

|  |  |
| --- | --- |
| RF5 | |
| **Descripción:** | El programa de guardar el archivo txt modificado. |
| **Entradas:** | Datos del archivo. txt modificado y ordenado. |
| **Salidas:** | El archivo debe de estar cargado. |
| **PreCondiciones:** | Los datos deben de haberse modificado y ordenado. |
| **Post-Condiciones:** | Se creara un archivo nuevo por cada dato modificado y ordenado. |