REQUERIMIENTOS FUNCIONALES

NOMBRE	**R1:** Permitir registrar clubes.
RESUMEN	El programa permite registrar Clubes con unos parámetros preestablecidos.
ENTRADAS	Nombre Fecha de creación Tipos de mascotas
SALIDAS	El programa registra el club en el sistema.

NOMBRE	**R2:** Permitir registrar dueños de mascotas
RESUMEN	El programa permite registrar dueños de mascotas con unos parámetros preestablecidos.
ENTRADAS	Nombre Apellidos Fecha de nacimiento Tipo de mascota que prefiere
SALIDAS	El programa registra los dueños de las mascotas en el sistema.

NOMBRE	**R3:** Permitir registrar mascotas.
RESUMEN	El programa permite registrar mascotas según unos parámetros preestablecidos
ENTRADAS	Identificación Nombre Fecha de nacimiento Género Tipo

SALIDAS	El programa registra la mascota en el sistema.

NOMBRE	**R4:** Almacenar la información de los clubes en un archivo plano.
RESUMEN	El programa permite almacenar la información de los clubes en un archivo plano.
ENTRADAS	
SALIDAS	El programa almacena la información de los clubes en el archivo plano.

NOMBRE	**R5:** Utilizar archivos con serialización para la información de los dueños y de las mascotas.
RESUMEN	El programa permite almacenar la información de las mascotas y los dueños en archivos serializados.
ENTRADAS	
SALIDAS	El programa almacena la información de las mascotas y de los dueños en archivos serializables.

NOMBRE	**R6:** Generar listados ordenados de los clubes, los dueños y las mascotas según el criterio elegido por el usuario.
RESUMEN	El programa permite generar listados ordenados según un criterio utilizado.
ENTRADAS	Criterio de ordenamiento
SALIDAS	El programa genera el listado ordenado.

NOMBRE	**R7:** Tener opciones de búsqueda por un criterio específico para la búsqueda de datos.
RESUMEN	El programa permite hacer búsquedas de la información del sistemas según el criterio elegido por el usuario
ENTRADAS	Criterio de ordenamiento
SALIDAS	El programa ha realizado la búsqueda del elemento.

NOMBRE	**R8:** Verificar que el dueño exista.
RESUMEN	El programa permite al usuario saber si el dueño ingresado existe en el sistema
ENTRADAS	Ninguna
SALIDAS	SI: La mascota se ha registrado correctamente. NO: La mascota no se ha registrado

NOMBRE	**R9:** Eliminar un club, una mascota o un dueño
RESUMEN	El programa permite al usuario eliminar un club, una mascota o un dueño
ENTRADAS	Nombre o Identificación
SALIDAS	Club, mascota o dueño eliminado

REQUERIMIENTOS NO FUNCIONALES

NOMBRE	**RNF1:Utilizar métodos de ordenamiento**
RESUMEN	El programa debe tener métodos de ordenamiento clásicos.
ENTRADAS	Ninguna
SALIDAS	El programa con métodos de ordenamientos clásicos.

NOMBRE	**RNF2: Utilizar métodos de búsqueda**
RESUMEN	El programa debe tener métodos de búsqueda clásicos.
ENTRADAS	Ninguna
SALIDAS	El programa con métodos de búsqueda clásicos.

NOMBRE	**RNF3: Utilizar las técnicas de comparación que incluyan la interfaz Comparable y Comparator**
RESUMEN	El programa debe tener métodos con la interfaz Comparable y Comparator
ENTRADAS	Ninguna
SALIDAS	Programa con interfaz Comparable y Comparator.

TRAZABILIDAD

REQUERIMIENTO	MÉTODO	CLASES
R1: Permitir registrar clubes.	**+ toAddAnewClub(String id, String name, String foundationDate, String type):void**	**Investor**
R2: Permitir registrar dueños de mascotas	**+addOwnerToAClub(String id, String name, String lastName, String fDate, int favtype):void**	**Club**
R3: Permitir registrar mascotas.	**+ addPet(String id, String name, int genre, int type, String date):void**	**Owner**
R4: Almacenar la información de los clubes en un archivo plano.	**+loadClubsInformation():void**	**Investor**
R5: Utilizar archivos con serialización para la información de los dueños y de las mascotas.	**+ OwnerDeserialize():ArrayList<Owner>** **+ OwnersSerialize():void**	**Club**
R6: Generar listados ordenados de los clubes, los dueños y las mascotas según el criterio elegido por el usuario.	**+ bubbleSortAmountOwner():void** **+ selectionSortId():void** **+ selectionSortName():void** **+ selectionSortFoundationDate():void** **+ selectionSortforFavPet():void** **+ compareTo(Pet p):int** **+ compare(Owner owner1, Owner owner2):int** **+ compareByFav(Owner owner1, Owner owner2):int** **+ compareTo(Owner owner):int** **+ compareToLName(Owner owner):int** **+ compareToId(Owner owner):int** **+ compareToDate(Owner owner):int**	**Club** **Pet** **Owner**
R7: Tener opciones de búsqueda por un criterio específico para la búsqueda de datos.	**+ searchClubByTheName(String name):Club** **+ searchClubByTheDateOfFundation(String dateOfFundation):Club** **+ searchClubByThId(String id):Club** **+ toSearchAPet(String nameOfThePet):Pet**	**Club** **Pet** **Owner**
R9: Eliminar un club, una mascota o un dueño	**deleteOwnerFromSystem(String id): void**	**Owner**

	deleteASpecificPet(String petToDelete):void **toRemoveAPetFromAnOwner(String petId):void**	

TRAZABILIDAD PARA REQUERIMIENTOS NO FUNCIONALES

REQUERIMIENTO	**METODOS**	**CLASES**
RNF1:Utilizar métodos de ordenamiento	**bubbleSortAmountOwner():void** **selectionSortId():void** **selectionSortName():void** **selectionSortFoundationDate():void** **selectionSortforFavPet():void**	**Investor**
RNF2: Utilizar métodos de búsqueda	**binaryByName(String name):Club** **searchClubByTheName(String name):Club** **searchClubByTheDateOfFundation(String dateOfFundation):Club** **searchClubByThId(String id):Club**	**Investor**
RNF3: Utilizar las técnicas de comparación que incluyan la interfaz Comparable y Comparator	**comparatorSortByIdOfTheOwner():void**	**Investor**

DISEÑO DE CASOS DE PRUEBA

Objetivo de la prueba: Verificar el correcto funcionamiento del método de ordenamiento				
CLASE	**MÉTODO**	**ESCENARIO**	**VALORES DE ENTRADA**	**RESULTADO**
Pet	compareToTest	setUpEscenario1()		El primer elemento de la ArrayList ordenado es el esperado

Objetivo de la prueba: Verificar el correcto funcionamiento del método de añadir mascota				
CLASE	**MÉTODO**	**ESCENARIO**	**VALORES DE ENTRADA**	**RESULTADO**
Owner	addPetTest	setUpEscenario1()		El total de los elementos del arraylist son uno.

Objetivo de la prueba: Verificar el correcto funcionamiento del método de añadir mascota				
CLASE	**MÉTODO**	**ESCENARIO**	**VALORES DE ENTRADA**	**RESULTADO**
Owner	compareTest	setUpEscenario1()		Al tener el mismo numero de elementos cada ArrayList, el método devuelve 0.

Objetivo de la prueba: Verificar el correcto funcionamiento del método de añadir mascota				
CLASE	**MÉTODO**	**ESCENARIO**	**VALORES DE ENTRADA**	**RESULTADO**
Owner	compareByFavTest	setUpEscenario1()		Al objeto del escenario tener una preferencia mayor a la cual se le estaba comparando con, el resultado da -1.

Objetivo de la prueba: Verificar el correcto funcionamiento del método de añadir mascota				
CLASE	**MÉTODO**	**ESCENARIO**	**VALORES DE ENTRADA**	**RESULTADO**
Owner	orderBySelection	setUpEscenario1()		Se organizó por el metodo de organización establecido.

Objetivo de la prueba: Verificar el correcto funcionamiento del método de añadir mascota				
CLASE	**MÉTODO**	**ESCENARIO**	**VALORES DE ENTRADA**	**RESULTADO**
Owner	petFavComparationTest()	setUpEscenario1()		Al los clubs tener el mismo gusto de mascotas, se recibió un cero como resultado del método comparteTo.