

# Trabajo Práctico 1 — Smalltalk

[7507/9502] Algoritmos y Programación III

Curso 2

Primer cuatrimestre de 2024

Alumno:	Pueyrredon, Manuel
Número de padrón:	111014
Email:	mapueyrredon@fi.uba.ar

## Índice

<b>1. Introducción</b>	<b>2</b>
<b>2. Supuestos</b>	<b>2</b>
<b>3. Modelo de dominio</b>	<b>2</b>
<b>4. Diagramas de clase</b>	<b>2</b>
<b>5. Detalles de implementación</b>	<b>4</b>
5.1. ModeloViejo . . . . .	4
5.2. CriterioFuerte . . . . .	4
<b>6. Excepciones</b>	<b>4</b>
6.1. CriterioNoValidoException . . . . .	4
6.2. ModeloNoValidoException . . . . .	5
6.3. TransformacionNoValidaException . . . . .	5
6.4. KiNoValidoException . . . . .	5
6.5. PeleadorNoRegistradoException . . . . .	5
<b>7. Diagramas de secuencia</b>	<b>5</b>

## 1. Introducción

El presente informe reúne la documentación de la solución del primer trabajo práctico de la materia Algoritmos y Programación III que consiste en desarrollar una aplicación de un sistema de una máquina. La cual permite reunir información de todos los peleadores que habitan el mundo y su nivel de pelea. Después de la llegada de los Saiyajin, se descubre la tecnología de los visores, que permite tomar una lectura aproximada del nivel de pelea de cada peleador. Inicialmente, solo se cuenta con los modelos antiguos, los cuales presentan una falla al superar cierta lectura. Sin embargo, después de la llegada de Freezer, se obtiene el nuevo modelo que no posee dicha falla.

## 2. Supuestos

Para la creación del sistema del rastreador tuvimos que con nuestro criterio suponer algunas cosas. La más importante, es que el ki de un Peleador, no puede ser menor o igual a 0. También pudimos suponer que los nombres de los peleadores, criterios o modelos a elegir otorgados por el usuario, tienen que ser exactamente igual al de los Tests, incluyendo mayúsculas. Por último, supusimos que si 2 peleadores son registrados con el mismo ki, el criterio elegirá el primer peleador que se registre.

## 3. Modelo de dominio

El diseño general de trabajo consiste en una clase principal `AlgoRastreadorZ`, la cual conocerá distintas clases las cuales juntas, podrán registrar niveles de pelea de luchadores. `AlgoRastreadorZ` tiene 2 posibles Criterios, Fuerte o Débil. Sabiendo que los 2 pueden hacer lo mismo y cumplen con la condición es un Criterio podemos aplicar 2 de los pilares de la programación orientada a objetos, (Herencia y polimorfismo), esto lo haremos creando una interfaz llamada `Criterio`. Las 2 hijas, Fuerte y Débil, deberán tener ambas un mensaje: `buscarPeleadorEnListaDeRastreos(rastreos: list)`. Esto nos permitirá eliminar condicionales, por polimorfismo, lo cual es bueno ya que si, luego queremos cambiar la implementación de alguna clase, será más fácil, solo tendremos que cambiar el código una vez.

Esto mismo pasa con las Clases Transformación cuyas hijas son (`Base`, `KaioKen`, `MonoGigante`), y con `Modelo`, cuyas hijas son (`ModeloNuevo`, `ModeloViejo`)

Pero es muy importante chequear que se cumpla la condición ".Es un", ya que la herencia es una asociación muy fuerte.

## 4. Diagramas de clase

Los Diagramas de Clases, son parte del diseño UML, (Unified Modeling Language), estos modelos se utilizan simplemente para poder comunicar de una forma clara y simple el diseño de un sistema a implementar o ya implementado. Es un modelo estático del sistema a construir o de una parte del mismo. En él se muestran clases y relaciones entre las mismas.

En la Figura 1, podemos ver como la Clase `AlgoRastreadorZ` tiene un atributo `rastreos`, señalado por la flecha 1 ->\*. Y tiene un atributo `criterio`. También vemos que `AlgoRastreadorZ` crea la clase `Transformación` y `modelo`. Pero `Modelo` y `Criterio` son solamente conocidas por `AlgoRastreadorZ`, es decir, al eliminarse `AlgoRastreadorZ`, dejará de existir `Criterio` y `Modelo`.

En la Figura 2,3 y 4 podemos ver distintas interfaces, o clases madres, que dan herencia a otras subclases. Cada una de estas tiene un método de para crear a una clase hija, en caso de que el parámetro no cumpla con las precondiciones, se lanzará una Excepción.

Figura 1: Diagrama General.

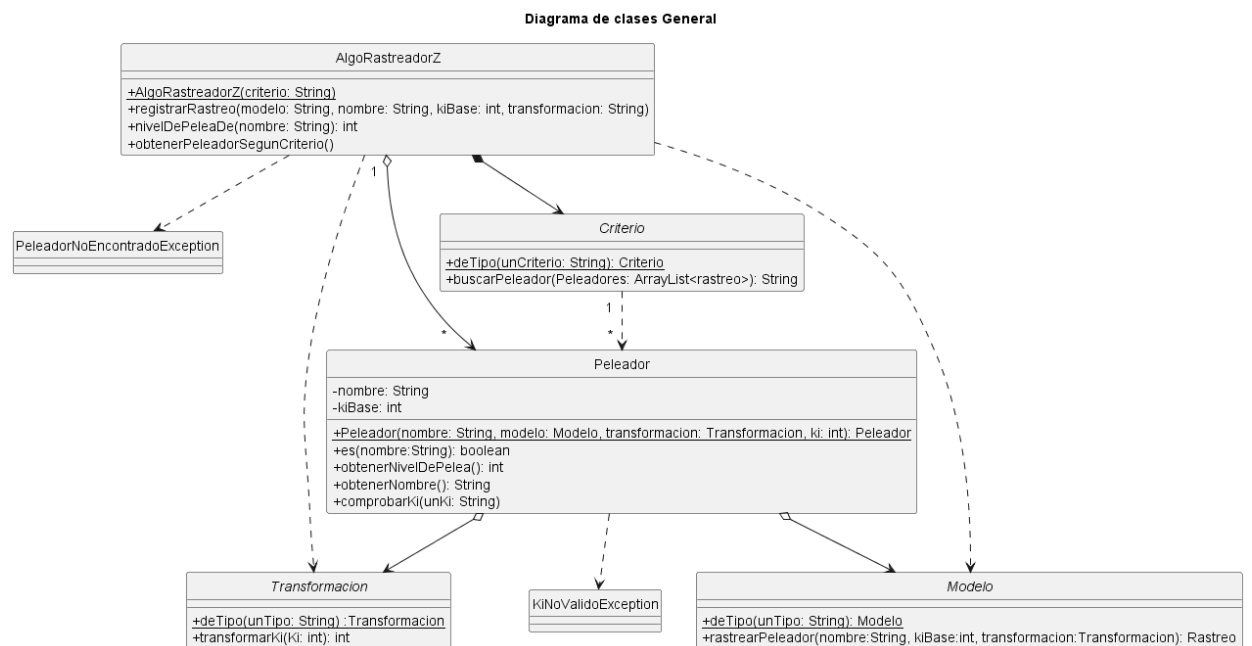


Figura 2: Diagrama de la Interface Criterio.

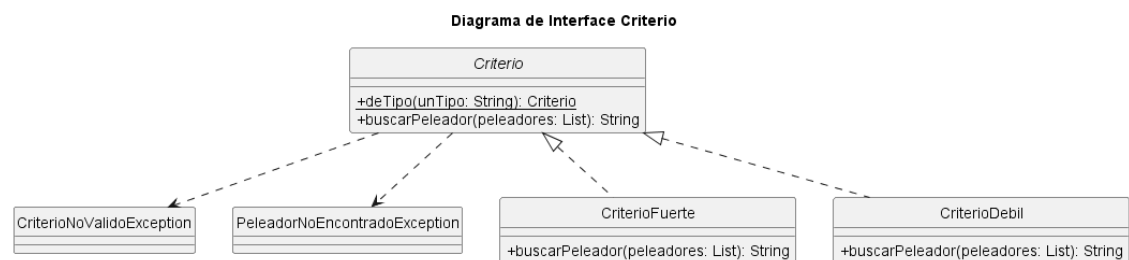


Figura 3: Diagrama de la Interface Modelo.

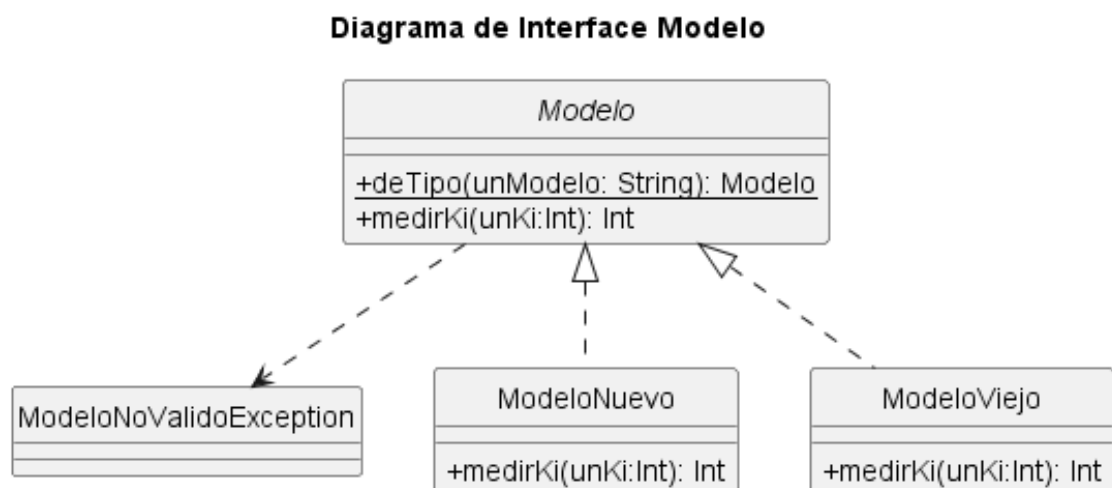
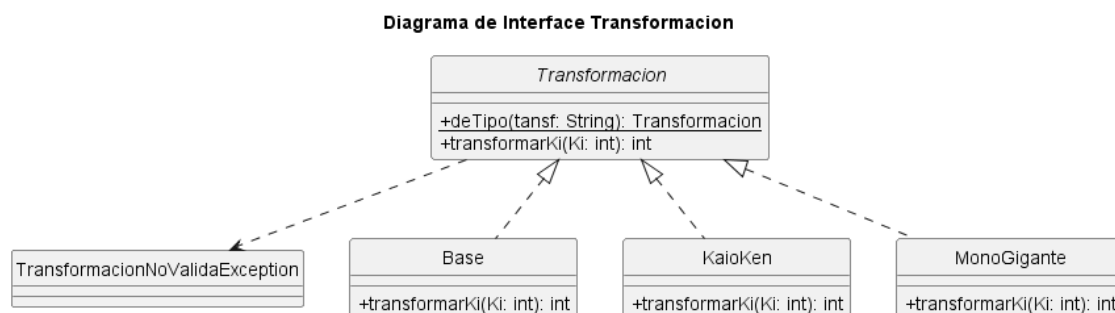


Figura 4: Diagrama de la Interface Transformacion.



## 5. Detalles de implementación

### 5.1. ModeloViejo

La clase ModeloViejo, tiene un metodo llamado: rastrearPeleadorConNombre: unNombre Ki-Base: unKi yTransformacion: unaTransformacion". El cual dado una transformacion, se transforma el ki pasado por parametro. Como este Visor no puede leer mas de 9000 de ki, si el ki base o el transformado es mayor a 9000, este se quedara con 9000, y luego creara un registro, del rastreo

```

| ki rastreo |
ki := unaTransformacion transformarKi: unKi.
ki := ki min: 9000.
rastreo := Rastreo conNombre: unNombre yKi: ki.
^ rastreo
  
```

### 5.2. CriterioFuerte

La clase CriterioFuerte, tiene un metodo llamado: "buscarPeleadorEnListaDeRastros".<sup>el</sup> cual dada una lista, determinara cual es el peleador, mas fuerte. En caso de No haber peleadores, se lanzara la Excepcion PeleadorNoRegistrado.

```

buscarPeleadorEnListaDePeleadores: historial
| pelearMinimo |

historial isEmpty: [
    PeleadorNoRegistradoException signal: 'No hay Rastros, rastrea antes de Buscar un peleador'
].
pelearMinimo := historial detectMax: [ :peleador |
    pelear obtenerNivelDePelea ].
^ pelearMinimo obtenerNombre.
  
```

## 6. Excepciones

### 6.1. CriterioNoValidoException

**Significado:** Cuando se reciba esta Excepcion como bien dice su nombre, significa que el usuario no ingreso un Criterio valido. Se mostrara una señal indicando: El Criterio no es valido.

**¿Donde se utiliza?** Esta se utiliza al crear el AlgoRastreadorZ conCriterioMas: el cual cuando quiere saber si su criterio es Fuerte o Debil, no podra decidir, por lo tanto se lanzara la excepcion.

## 6.2. ModeloNoValidoException

**Significado:** Como dice su nombre, es cuando el Usuario, no ingresa un Modelo valido, la Excepcion ira con una señal indicando: 'el Modelo no es valido'.

**¿Donde se utiliza?** Esta se lanza en la interface Modelo, cuando el usuario ingresa un modelo que no es ni "Viejo" ni "Nuevo".

## 6.3. TransformacionNoValidaException

**Significado:** Al igual que las otras 2 Excepciones, si recibis esta Excepcion es que la transformacion indicada no es valida. Se señalizara al usuario con un mensaje: La transformacion no es valida.

## 6.4. KiNoValidoException

**Significado:** Esta Excepcion significa, que el Ki otorgado por el usuario, no es Valido, es decir es menor o igual a 0.

**¿Donde se utiliza?** Esta tranformacion es lanzada cuando la clase Transformacion quiera transformar un ki y este no cumpla con las precondiciones.

## 6.5. PeleadorNoRegistradoException

**Significado:** Esta Excepcion, significa que el peleador que estamos buscando no existe, por lo tanto se lanzara la excepcion.

**¿Donde se utiliza?** Esta se utiliza cuando se quiere buscar un peleador con un criterio de busqueda, o cuando queremos el nivel de pelea de un Peleador no antes registrado.

# 7. Diagramas de secuencia

Un diagrama de secuencia muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo.

Figura 5: Diagrama de Secuencia Test 01 Parte 1

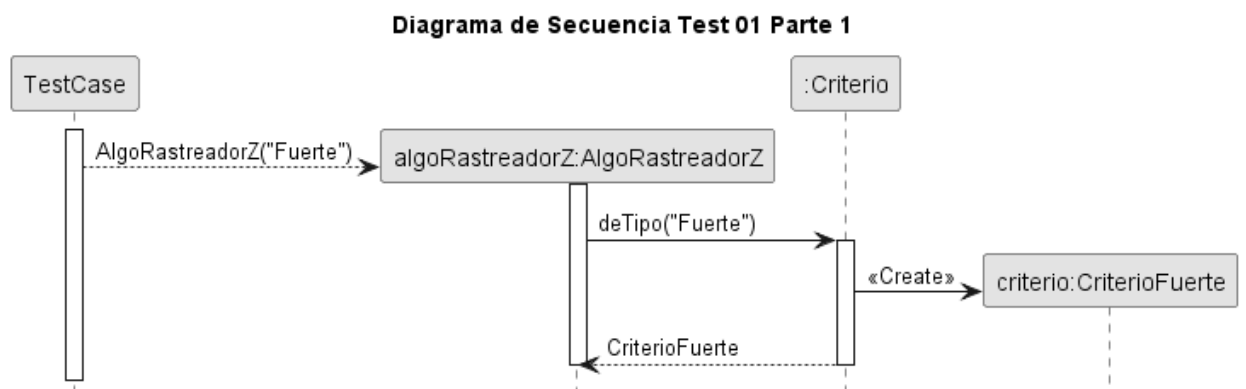


Figura 6: Diagrama de Secuencia Test 01 Parte 2

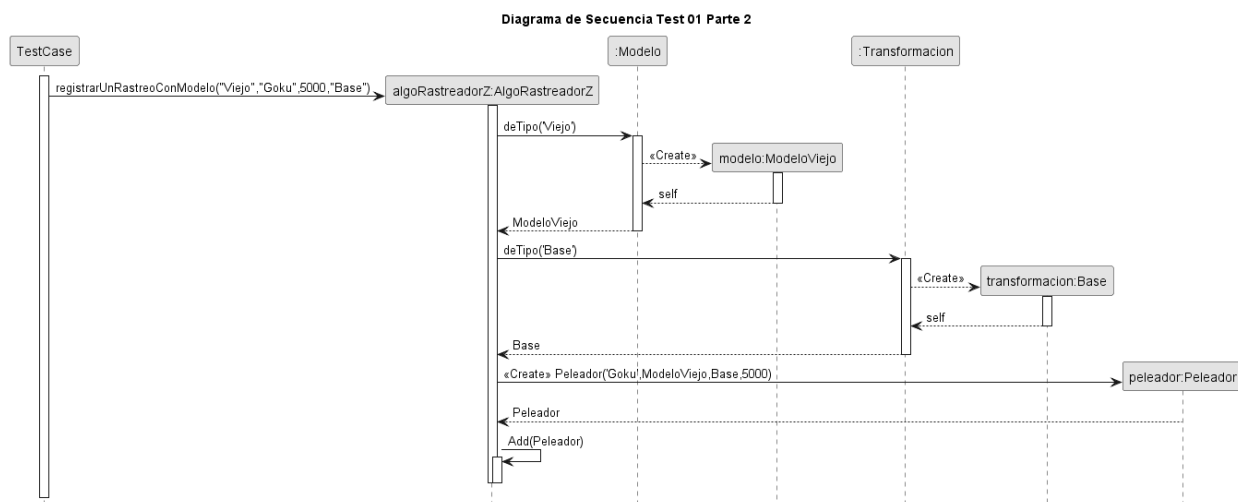
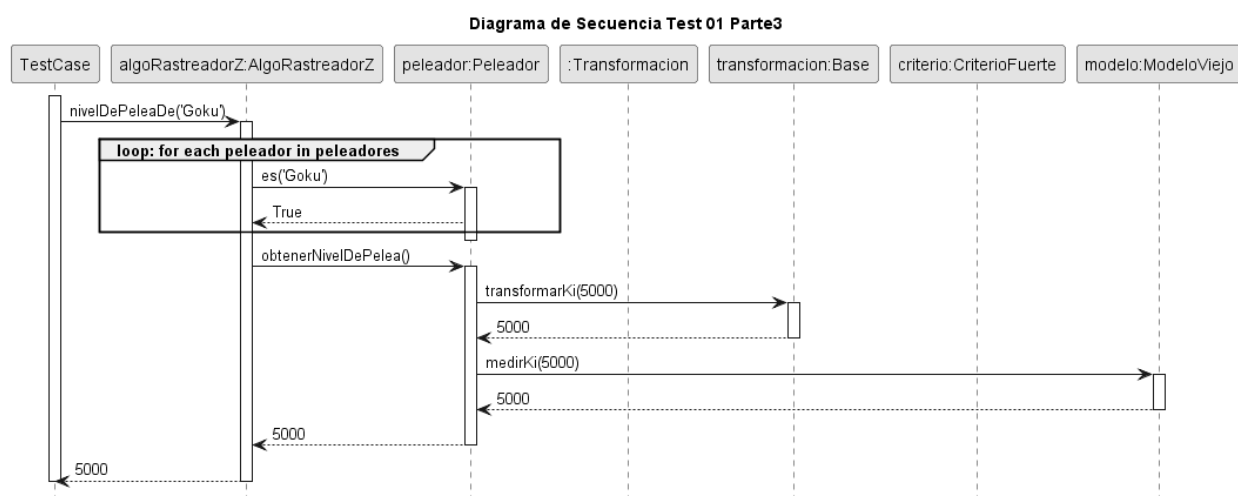


Figura 7: Diagrama de Secuencia Test 01 Parte 3



En este Diagrama de Secuencia, el usuario crea un AlgoRastreadorZ con un Criterio Fuerte. El cual la clase AlgoRastreadorZ delega a Criterio el cual tiene un metodo de clase el cual ve si el criterio ingresado es valido y en este caso crea un criterio Fuerte, el cual es guardado en un atributo de la clase AlgoRastreadorZ.

Luego el Actor le pide registrar un rastreo al Rastreador, el cual primero le pide a la clase Modelo, que le de un objeto de Tipo Viejo.

Aparte de eso el Rastreador, tambien necesita una transformacion, la cual mediante un metodo de clase de Transformacion se creara la clase necesitada, en este caso Base.

Una vez que tiene los objetos necesarios, el rastreador, crea un Peleador con un nombre, kiBase, Transformacion, y Modelo.

Cuando este sea creado, se agregara a la lista de Peleadores, de la clase AlgoRastreadorZ

Nuevamente el Actor le hace un pedido al rastreador, en este caso, el nivel de pelea del peleador 'Goku'. Para esto, el rastreador iterara por cada peleador en su lista de peleadores, y le preguntara si es 'Goku'. Cuando uno responda que si. El rastreador le pedira su nivel de pelea.

cuando al Peleador le pidan su nivel de Pelea, este le va a pedir a su transformacion que transforme su ki base, y luego le pedira a el modelo, que mida su ki. Y luego de estas operaciones, se le devolvera el ki a el rastreador, el cual a su vez le devolvera al Actor, finalizando la ejecucion del test.

Figura 8: Diagrama de Secuencia Test 08 Parte 1

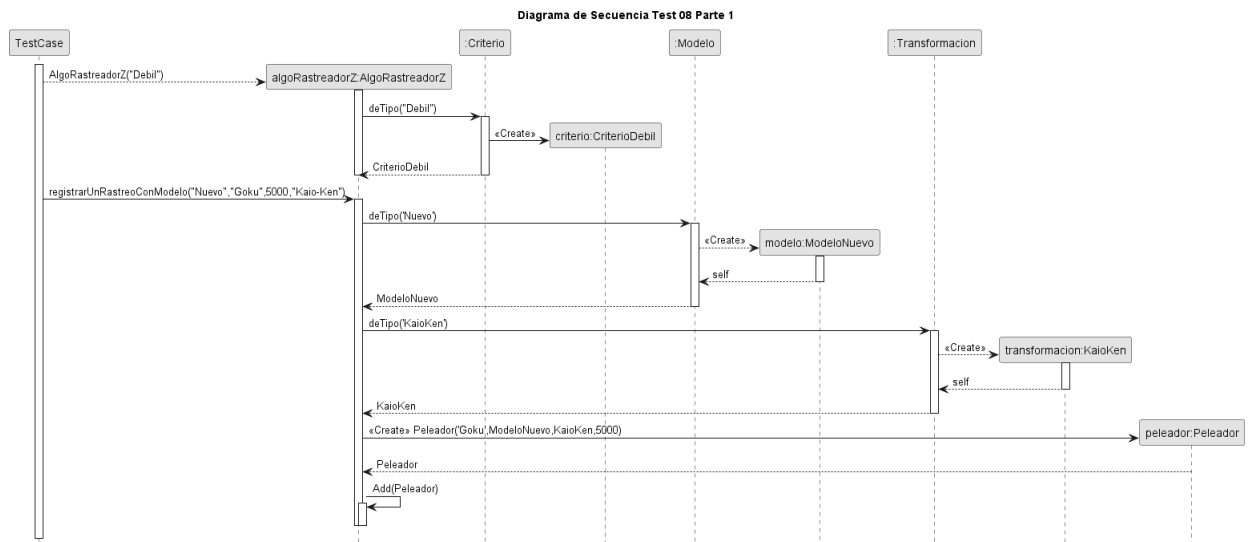


Figura 9: Diagrama de Secuencia Test 08 Parte 2

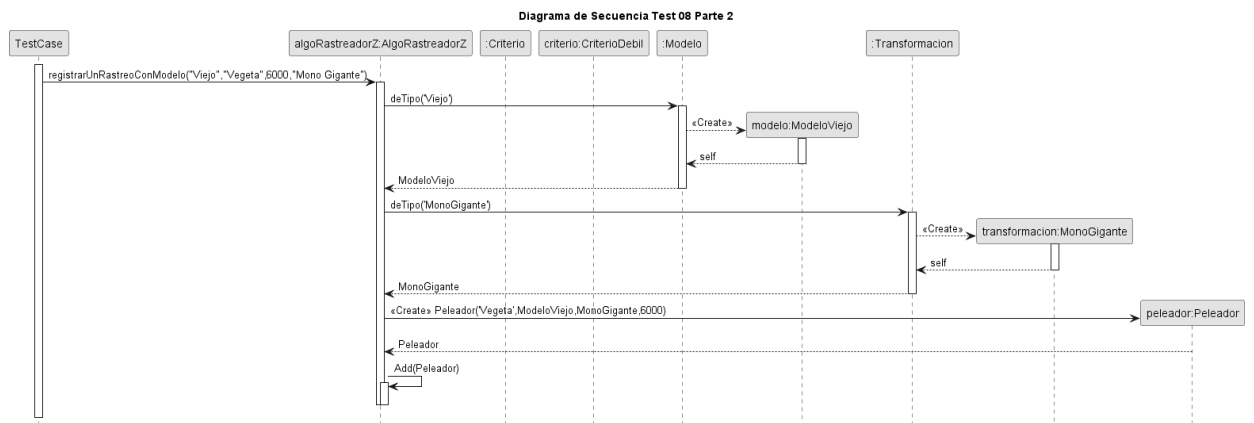


Figura 10: Diagrama de Secuencia Test 08 Parte 3



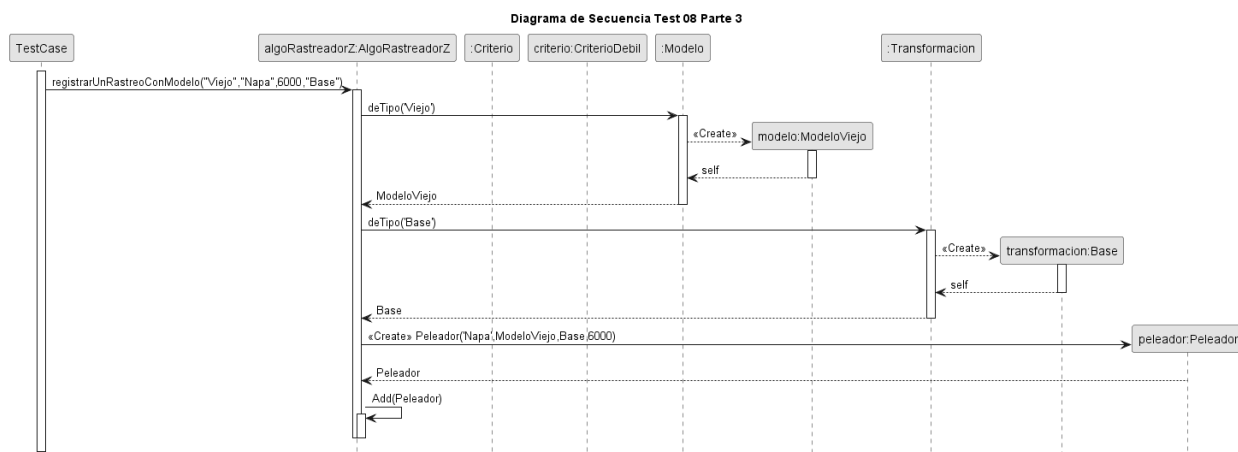
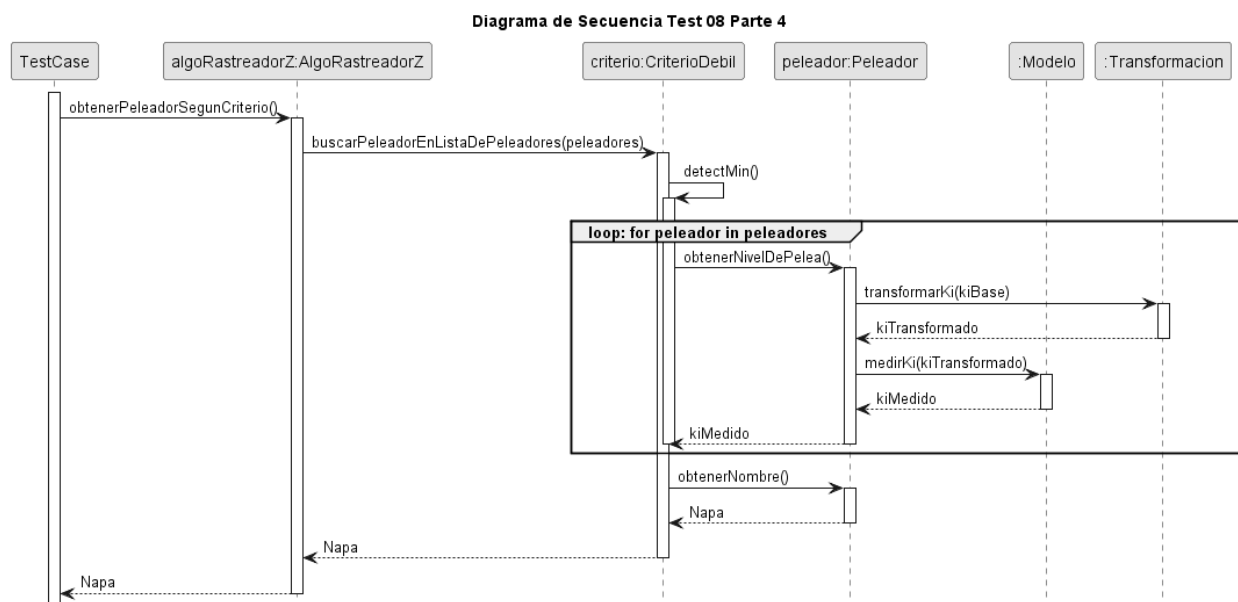


Figura 11: Diagrama de Secuencia Test 08 Parte 4



Este segundo Diagrama lo dividimos en varias partes así queda más claro la implementación del mismo.

La primera parte, es la más clara, el usuario crea un `AlgoRastreadorZ` con un `Criterio Fuerte`. El cual la clase `AlgoRastreador` delega a `Criterio` y en base a si es `Fuerte` o `Debil`, se crea el objeto deseado como vimos anteriormente en el Test 01.

La segunda parte consiste en realizar un rastreo, el proceso es el siguiente: `algoRastreadorZ`, le pide a `Modelo` y a `Transformacion` los objetos requeridos a través de métodos de Clase. Y luego se crea un `Peleador` con los objetos creados anteriormente.

En el caso 8 este proceso se repite en total 3 veces.

Finalmente, el Usuario, le pedirá al `algoRastreadorZ` `obtenerPeleadorSegunCriterio`, el cual el `Rastreador`, delegará a `Criterio`, pasándole una lista de registro de peleadores. El `criterio`, en este caso `Debil`, iterará la lista preguntándole al `peleador` su nivel de `Pelea`, el cual primero será transformado por la `Transformacion` ingresada anteriormente, y luego medida con el `Modelo` dado. Una vez hecho esto, tendrá el `peleador` con menor `ki`. Al cual se le pedirá su nombre y se lo devolverá al Usuario.