FACULTAD DE INGENIERIA
Universidad de Buenos Aires

# 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

**1. Introducción** **2**

**2. Supuestos** **2**

**3. Modelo de dominio** **2**

**4. Diagramas de clase** **2**

**5. Detalles de implementación** **4**
5.1. ModeloViejo . . . . . 4
5.2. CriterioFuerte . . . . . 4

**6. Excepciones** **4**
6.1. CriterioNoValidoException . . . . . 4
6.2. ModeloNoValidoException . . . . . 4
6.3. TransformacionNoValidaException . . . . . 4
6.4. KiNoValidoException . . . . . 5
6.5. PeleadorNoRegistradoException . . . . . 5

**7. Diagramas de secuencia** **5**

# 1. Introducción

El presente informe reune la documentación de la solución del primer trabajo práctico de la materia Algoritmos y Programación III que consiste en desarrollar una aplicacion de un sistema de una maquina. 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 creacion del sistema del rastreador tuvimos que con nuestro criterio suponer algunas cosas. La mas importante, es que el ki de un Peleador, no puede ser menor o igual a 0. Tambien pudimos suponer que los nombres de los peleadores, criterios o modelos a elegir otorgados por el usuario, tienen que ser excactamente igual al de los Tests, incluyendo mayusculas. Por ultimo, supusimos que si 2 peleadores son registrados con el mismo ki, el criterio elegira el primer peleador que se registro.

# 3. Modelo de dominio

El diseño general de trabajo consiste en una clase principal AlgoRastreadorZ, la cual conocera distintas clases las cuales juntas, podran registrar niveles de pelea de luchadores. AlgoRastreadorZ tiene 2 posbles Criterios, Fuerte o Debil. Sabiendo que los 2 pueden hacer lo mismo y cumplen con la condicion es un Criterio podemos aplicar 2 de los pilares de la programacion orientada a objetos, (Herencia y polimorfismo), esto lo haremos creando una interfaz llamada Criterio. Las 2 hijas, Fuerte y Debil, deberan tener ambas un mensaje: buscarPeleadorEnListaDeRastreos(rastreos: list). Esto nos permtira eleminar condicionales, por polimorfismo, lo cual es bueno ya que si, luego queremos cambiar la implementacion de alguna clase, sera mas facil, solo tendremos que cambiar el codigo una vez.

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

Pero es muy importante chequear que se cumpla la condicion ."Es un", ya que la herencia es una asociacion 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 eståtico del sistema a construir o de una parte del mismo. En él se muestran clases y relaciones entre Ias mismas.

En la Figura 1, podemos ver como la Clase AlgoRastreadorZ tiene un atributo rastreos, señalizado por la flecha 1 ->*. Y tiene un atributo criterio. Tambien vemos que AlgoRastreadorZ crea la clase Tranformacion y modelo. Pero Modelo y Criterio son solamente conocidas por AlgoRastreadorZ, es decir, al eliminarse AlgoRastreadorZ, dejara 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 metodo de para crear a una clase hija, en caso de que el parametro no cumpla con las precondiciones, se lanzara una Excepcion.

Figura 1: Diagrama General.

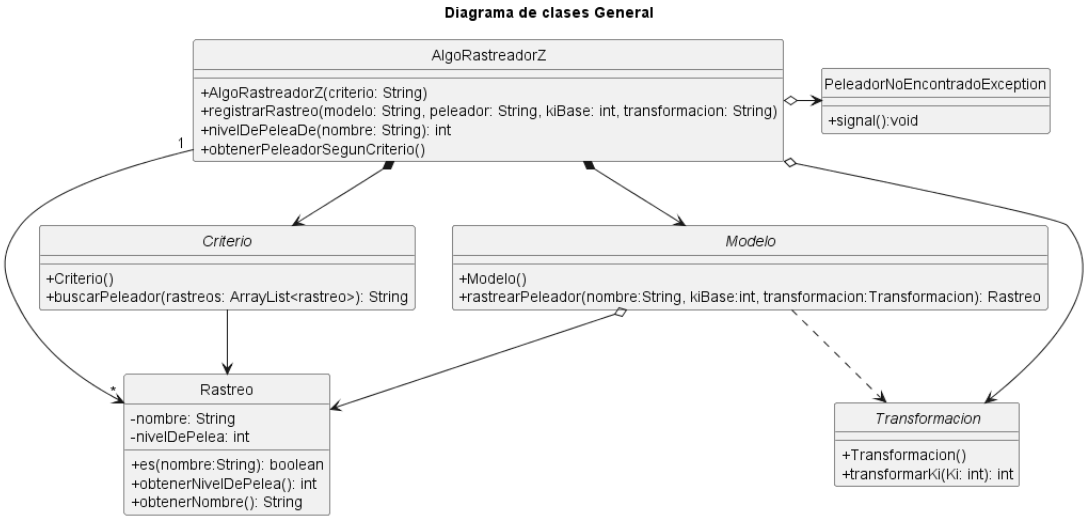



Figura 2: Diagrama de la Interface Criterio.

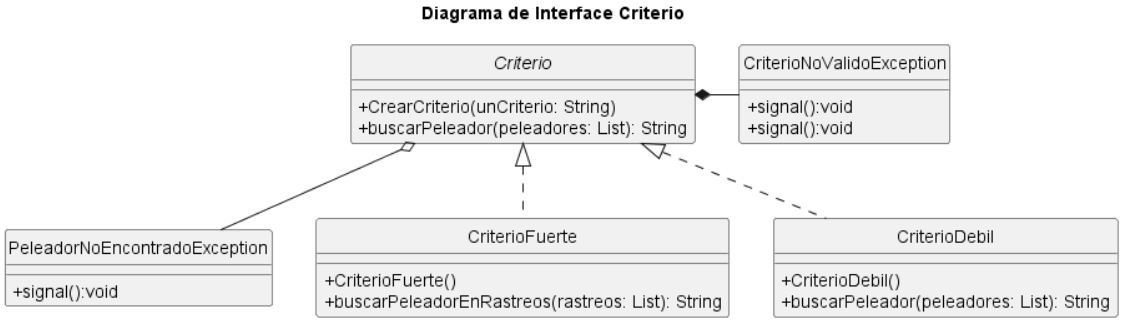



Figura 3: Diagrama de la Interface Modelo.

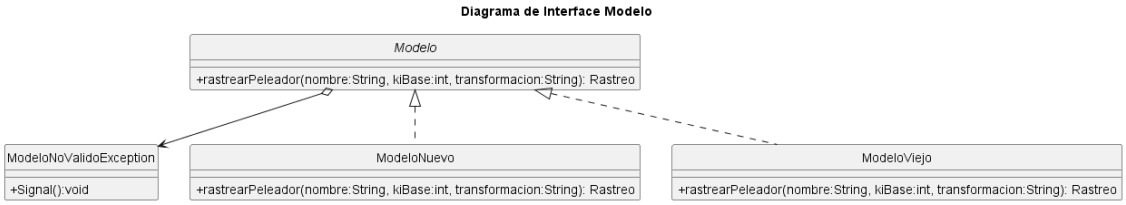
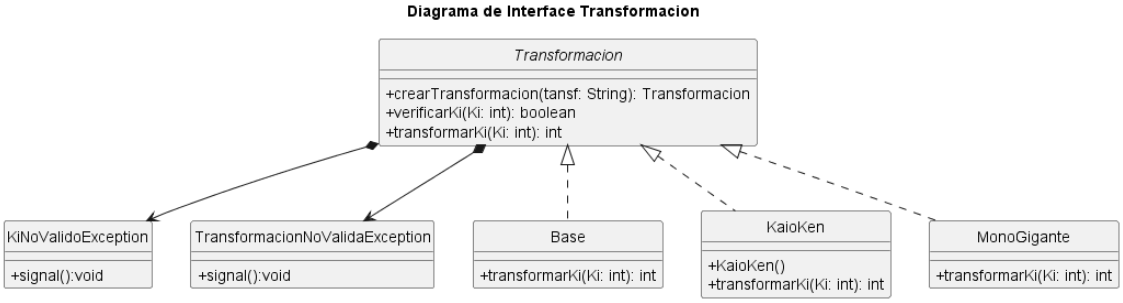



Figura 4: Diagrama de la Interface Transformacion.



# 5. Detalles de implementación

## 5.1. ModeloViejo

La clase ModeloViejo, tiene un metodo llamado: rastrearPeleadorConNombre: unNombre KiBase: 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: "buscarPeleadorEnListaDeRastreos.[el] cual dada una lista, determinara cual es el peleador, mas fuerte. En caso de No haber peleadores, se lanzara la Excepcion PeleadorNoRegistrado.

```
buscarPeleadorEnListaDeRastreos: aCollection
| peleadorMaximo |

peleadorMaximo := aCollection detectMax: [ :peleador|  peleador obtenerNivelDePelea ].
peleadorMaximo ifNil: [
PeleadorNoRegistradoException signal:
'No hay Rastreos, rastrea antes de Buscar un peleador' ].
^ peleadorMaximo 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 Çriterio"no es un criterio 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 "Modelo"no es un modelo 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 transfomracion "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.

En la Figura 5 podemos ver como el actor, en este caso TestCase, le va pidiendo cosas al objeto algoRastreadorZ, el cual este ira delegando a otras clases, para cumplir con los requisitos.

Figura 5: Diagrama de Secuencia Test 01

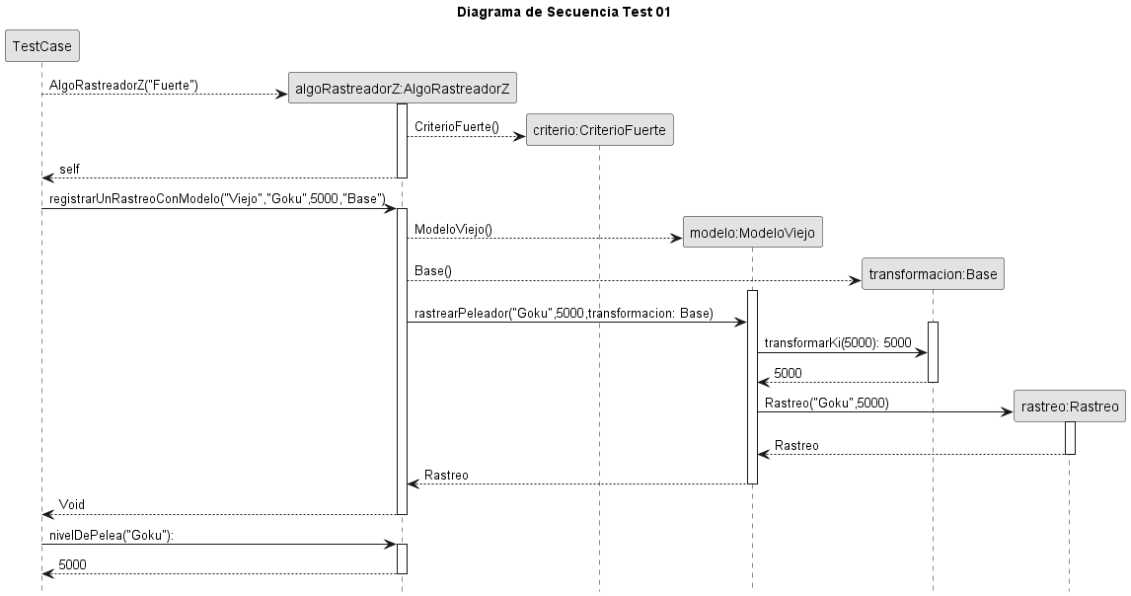



Figura 6: Diagrama de Secuencia Test 06 Parte 1

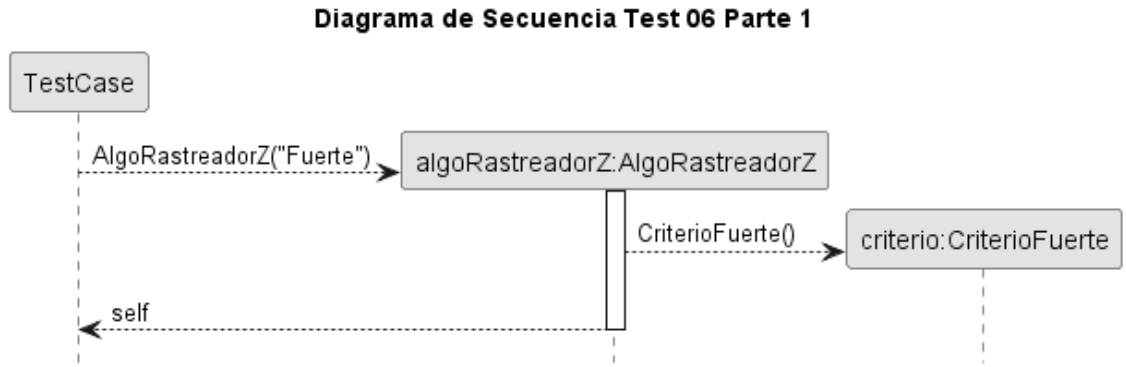



Figura 7: Diagrama de Secuencia Test 06 Parte 2

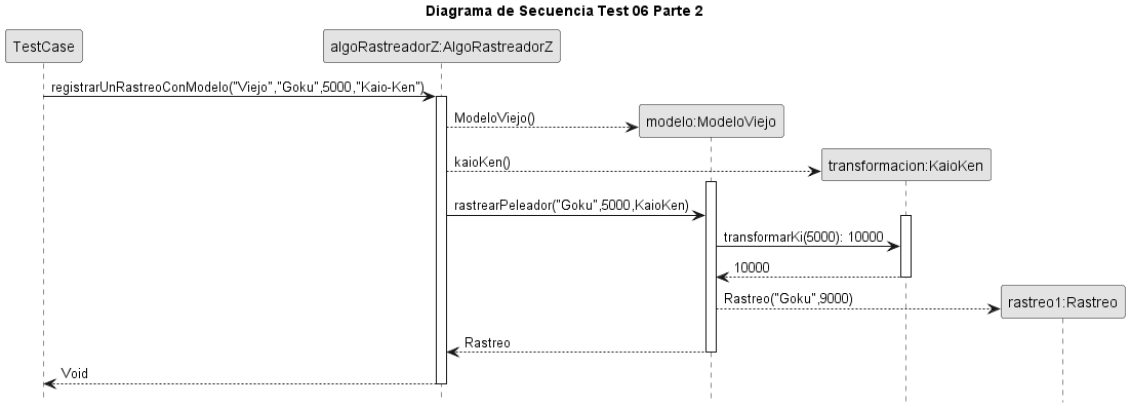



Figura 8: Diagrama de Secuencia Test 06 Parte 3

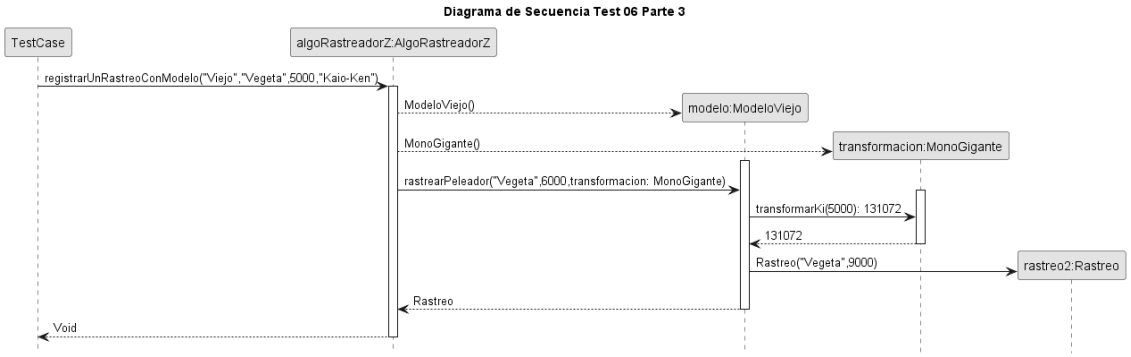
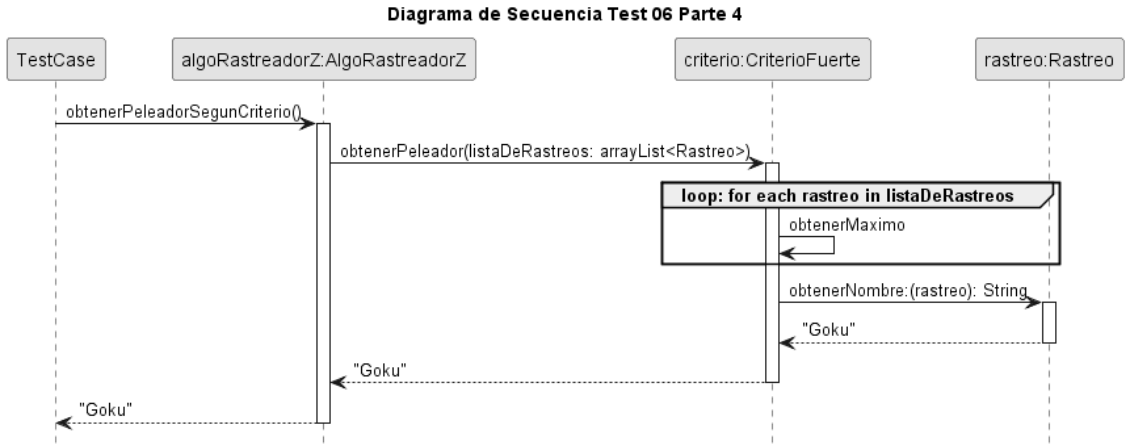



Figura 9: Diagrama de Secuencia Test 06 Parte 4




Este segundo Diagrama lo dividimos en varias partes asi queda mas claro la implementacion del mismo.

La primera parte, es la mas clara, el usuario crea un AlgoRastreadorZ con un Criterio Fuerte.

La segunda y tercer parte consiste en realizar un rastreo, algo similar a la figura 5 (caso 1), pero al ser un modelo viejo y el peleador tener mas de 9000 de ki, este creara un rastreo de ki 9000, y se guardara en la lista de rastreos los rastreos 1 y 2.

finalmente, el Usuario, le pedira al algoRastreadorZ obtenerPeleadorSegunCriterio, el cual el Rastreador, delegara a Criterio, pasandole una lista de registro de rastreos. El criterio, en este caso Fuerte, iterara la lista hasta encontrar el rastreo con maximo ki, una vez encontrado. obtendra el nombre del rastreo y se lo devolvera al Usuario.