FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 1 — Smalltalk

[7507/9502] Algoritmos y Programación III Curso
X Segundo cuatrimestre de 2024

Alumno:	FULLANA, Atuel
Número de padrón:	110247
Email:	afullana@fi.uba.ar

Índice

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 aplicación de un sistema de atletas de una olimpiada en Pharo utilizando los conceptos del paradigma de la orientación a objetos vistos hasta ahora en el curso.

2. Supuestos

Estos son los supuestos que surgieron en base a las pruebas dadas por la catedra:

- Un atleta es capaz de saber cual es su peso máximo.
- La edad, altura y peso no pueden ser negativos, en caso de que sean negativos lanzaran un error especifico.
- El criterio no puede ser distinto a Debil o Fuerte.
- La AlgoOlimpiada es capaz de saber cual es el atleta registrado más fuerte o débil.
- No se puede obtener el atleta más débil o mas fuerte de una AlgoOlimpiada sin atletas registrados.
- Cuando se comprar con una cierta precisión, hay un margen de error menor a la precisión.

3. Diagramas de clase

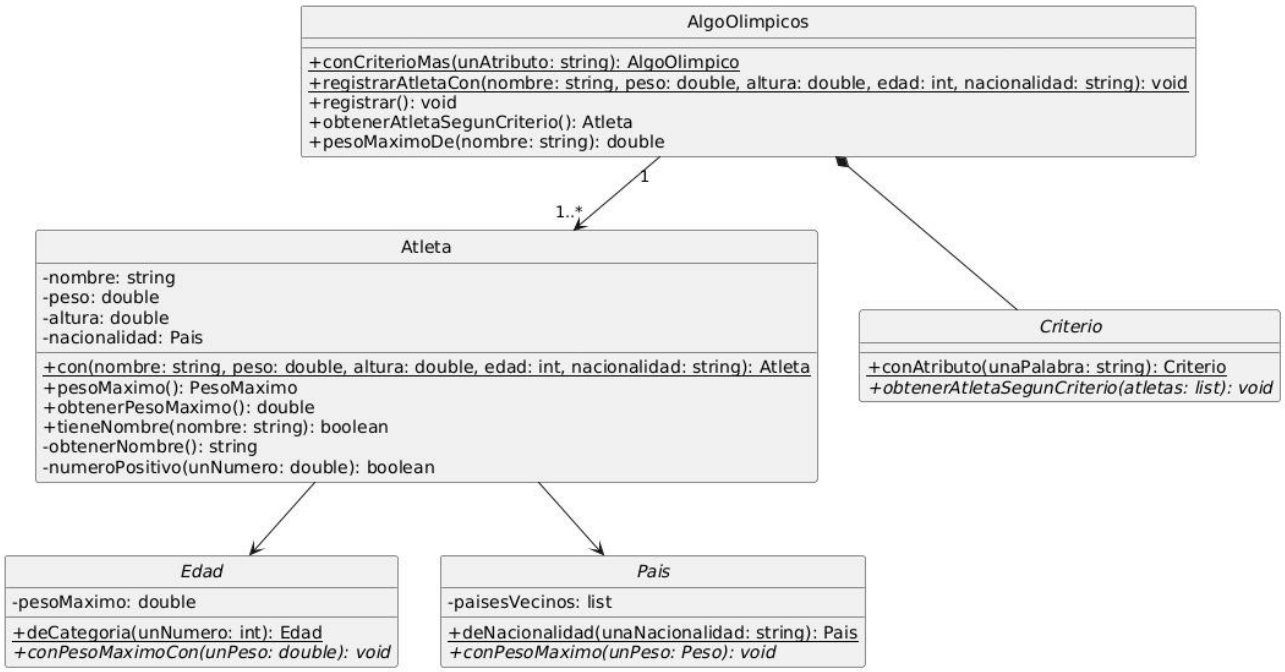

Figura 1: Diagrama de clases abstractas.

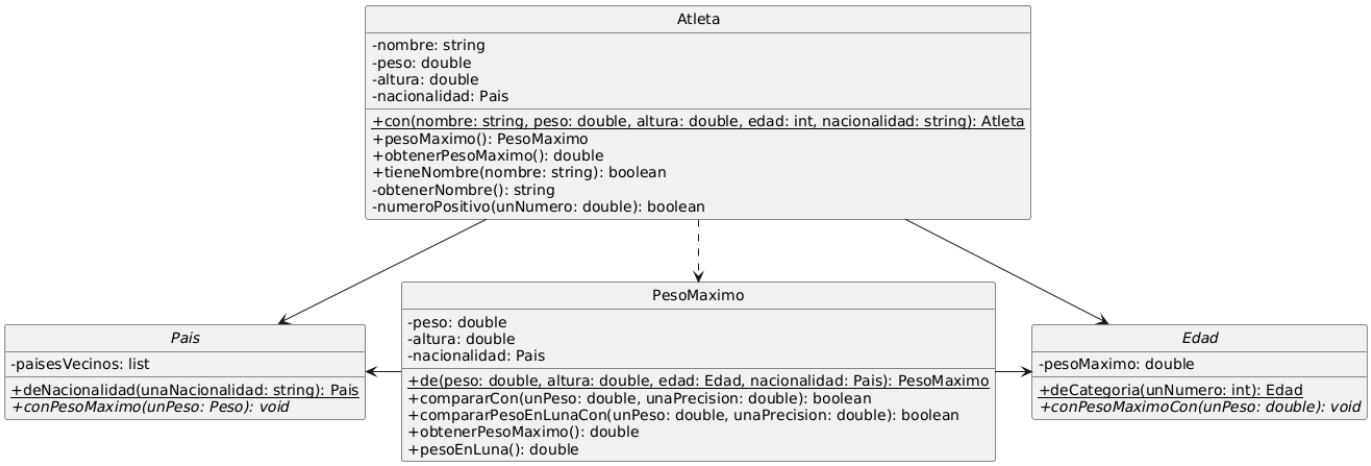

Figura 2: Diagrama de la clase Atleta

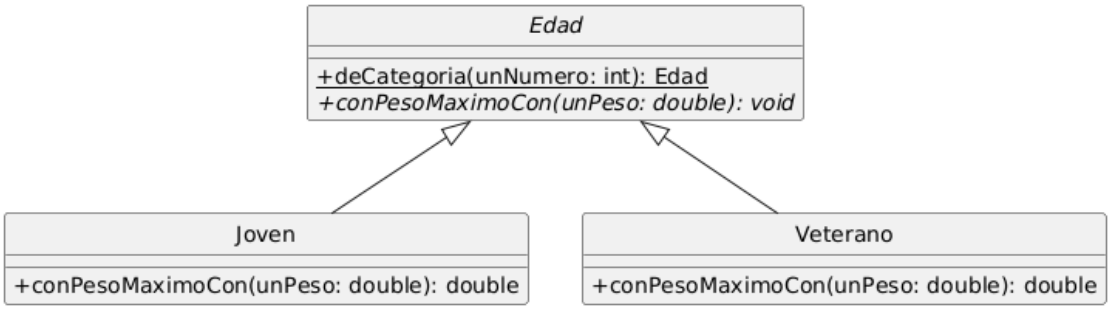

Figura 3: Diagrama de la clase edad.

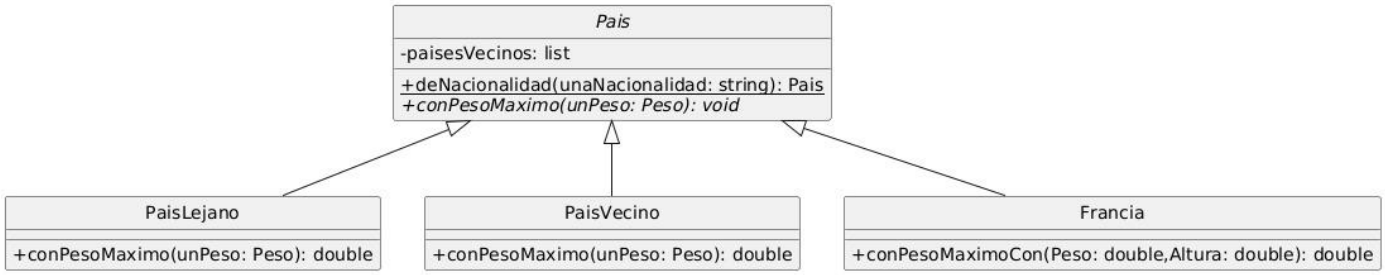

Figura 4: Diagrama de la clase país.

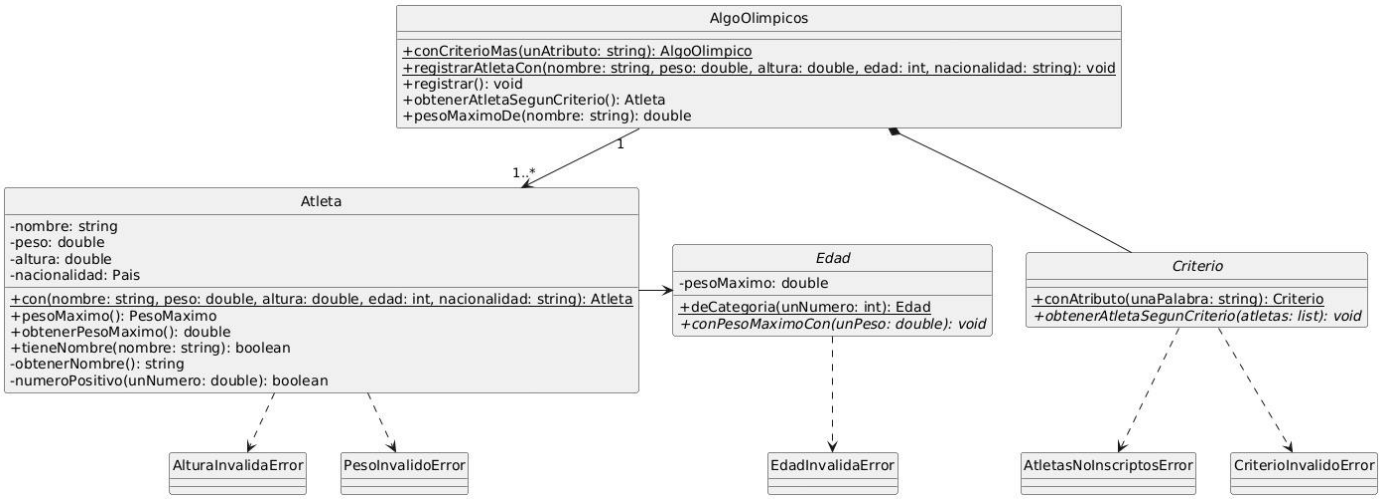

Figura 5: Diagrama de excepciones.

4. Detalles de implementación

4.1. Pilares de la POO.

Al principio el código fue creado siguiendo el Desarrollo Guiado por Pruebas (TDD), con las pruebas que nos proporcionó la catedra.

Implementando las clases y los métodos con el menor esfuerzo para que pasen las pruebas hasta que pasen las primeras pruebas.

Luego en la refactorización, se empezaron a aplicar los pilares de la POO.

Un claro ejemplo de la herencia es la refactorización realizada en la clase Criterio, donde se detectó código repetido, entonces se creó una clase abstracta para poder delegar las responsabilidades, usando el concepto Tell, Don't Ask, donde las clases no solicitan datos para procesarlos, sino que delegan la lógica a los objetos mismos.

Además, gracias al encapsulamiento, se ocultan los detalles de implementación de los métodos, manteniendo la información accesible solo a través de los métodos.

Una de las mayores ventajas de esto es que el código es escalable, es decir, si en un futuro se intenta agregar otro tipo de criterio, con una simple implementación y crear una nueva clase ya se podría añadir otro criterio, sin modificar o modificando lo menos posible, el código ya realizado.

Al momento de crear la solución del código se usó la abstracción, ya que solamente se crearon los métodos necesarios para resolver el problema, no se implementaron cosas que no fueron necesarias o que no se utilizan para solucionar dicho problema.

Tanto en las clases hijas de criterio, país y edad se utilizó polimorfismo, donde las clases hijas entienden el mismo mensaje, pero se comportan de manera distinta.

4.2. Implementaciones.

- Solamente hay 2 criterios válidos, Fuerte y Debil.
- La edad, altura y peso no pueden ser valores negativos.
- La clase Pais tiene un atributo paisesVecinos, que es una lista que se crea con el initialize y se le agregan todos los países cercanos a Francia.
- La clase criterio tiene recibe una lista de Atletas, y las clases hijas la recorren hasta encontrar el Atleta deseado
- En las pruebas unitarias se puede ver como la clase Atletas se puede crear sin necesidad de llamar al método registrarCon de la clase AlgoOlimpicos, pero en las pruebas dadas por la catedra solamente se creaban cuando se registran.

5. Excepciones

- Excepción AtletasNoInscriptosError, esta excepción se lanza cuando se intenta obtener un atleta de una AlgoOlimpiada vacía, es decir, cuando no se registró ningún atleta en dicha AlgoOlimpiada. El motivo de su razón de ser es que no se puede obtener al atleta mas fuerte de una olimpiada sin atletas, tiene que registrarse un atleta antes.
- Excepción CriterioInvalidoError, esta exacepción se lanza cuando se intenta establecer un criterio con un string distinto a Fuerte o Debil. El motivo de su razón de ser es que el string será el responsable de crear el criterio fuerte o débil de la AlgoOlimpiada.

La razón de ser de las 3 próximas excepciones es que los datos altura, edad y peso se usaran después para el cálculo del peso máximo y por un tema de implementación el peso máximo no puede ser negativo.

- Excepción AlturaInvalidaError, esta excepción se lanza cuando se intenta crear un atleta con una altura negativa.
- Excepción EdadInvalidaError, esta excepción se lanza cuando se intenta crear un atleta con una edad negativa.
- Excepción PesoInvalidoError, esta excepción se lanza cuando se intenta crear un atleta con un peso negativa.

6. Diagramas de secuencia

En el primer diagrama se muestran las acciones que hace el programa cuando se ejecuta el ***test01UnAtletaConCiertoPesoYEdadPuedeLevantarCiertosKilos***

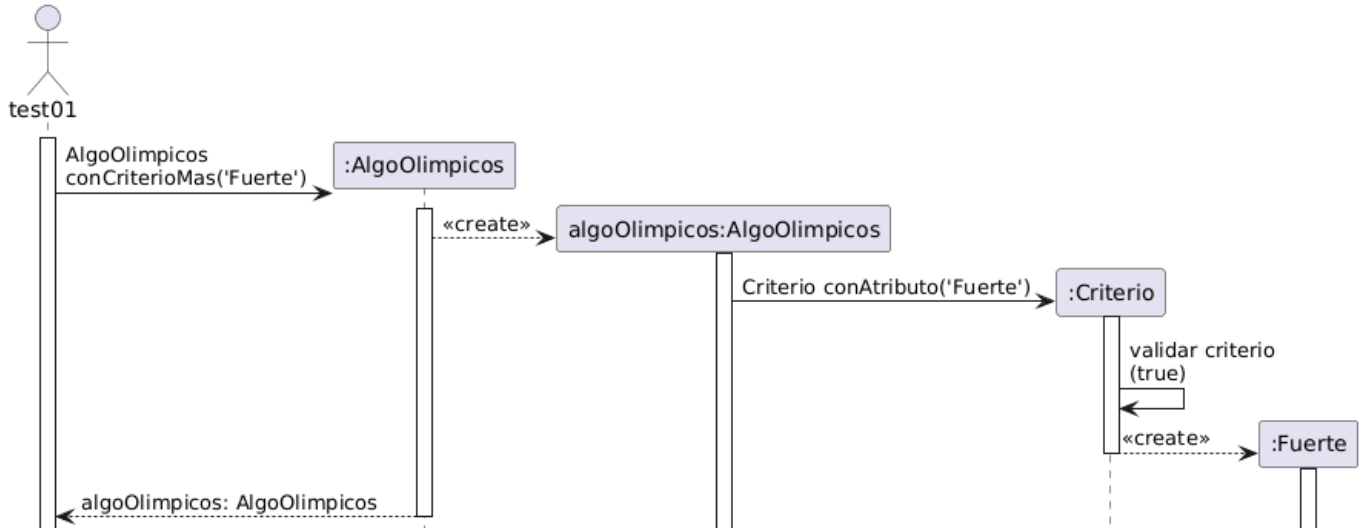

Figura 1: muestra cómo se crea la clase algoOlimpicos con el criterio más fuerte

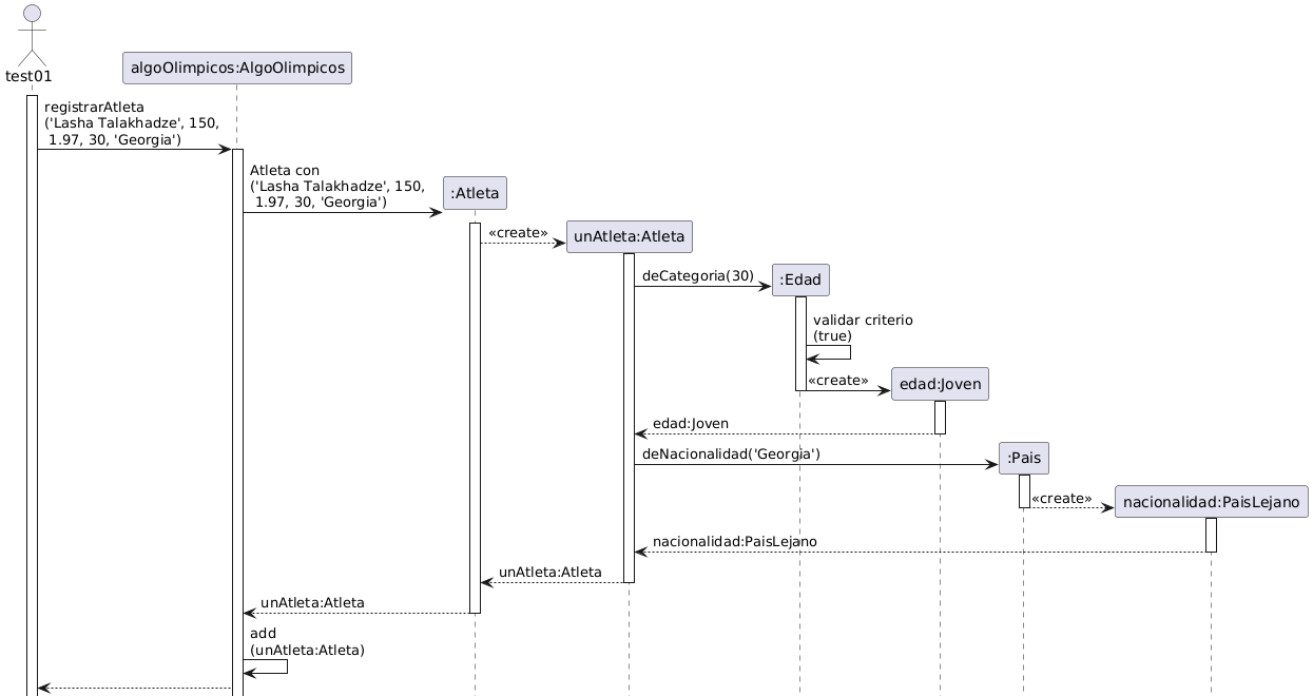

Figura 2: muestra cómo se registra cierto atleta.

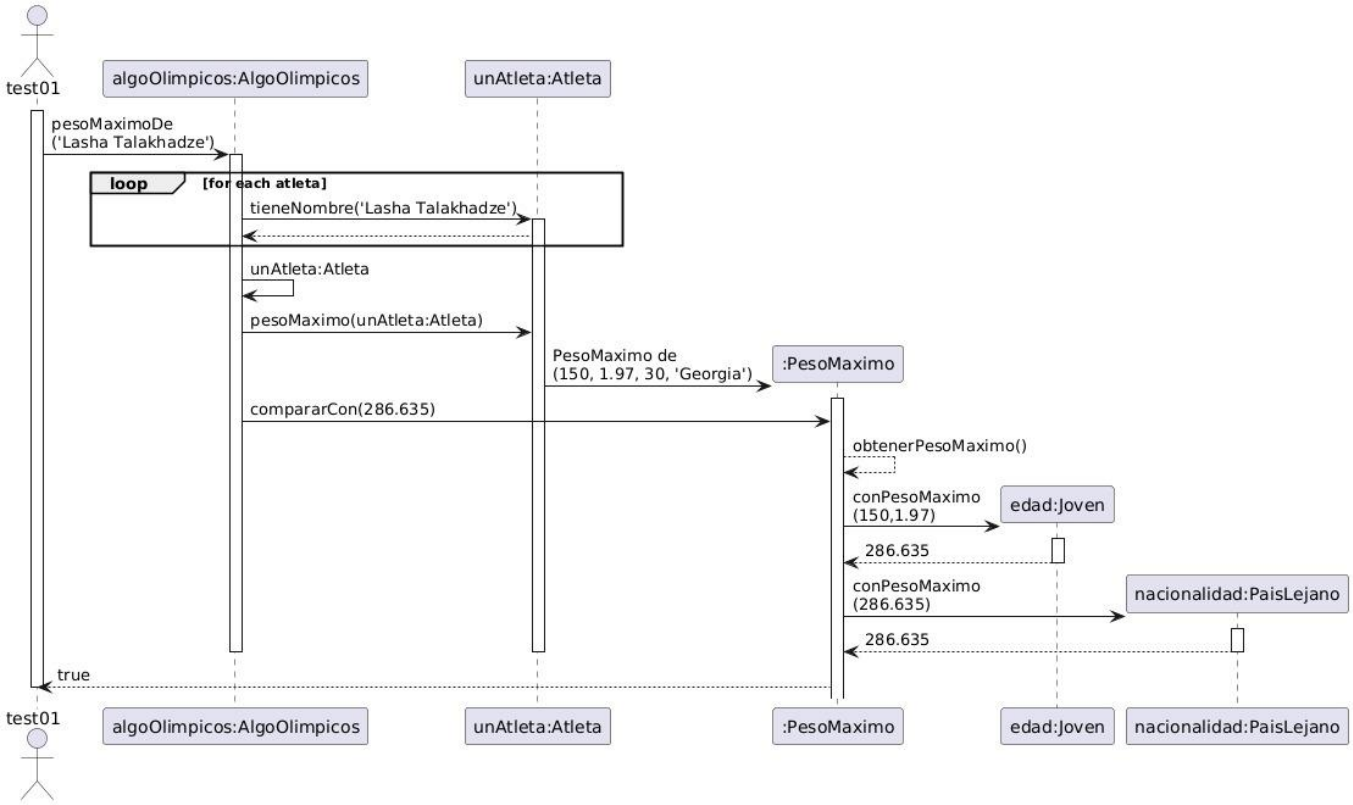

Figura 3: muestra como hace el programa para pasar el ***test01UnAtletaConCiertoPesoYEdadPuedeLevantarCiertosKilos***.

En este caso del loop, en el test01 es un caso feliz, a continuación, se muestran los dos casos, el feliz y el no feliz.

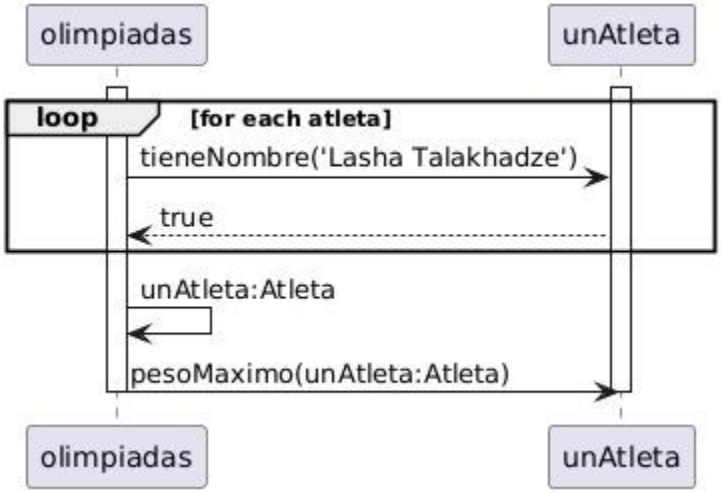

Figura 4: caso feliz para encontrar el atleta deseado

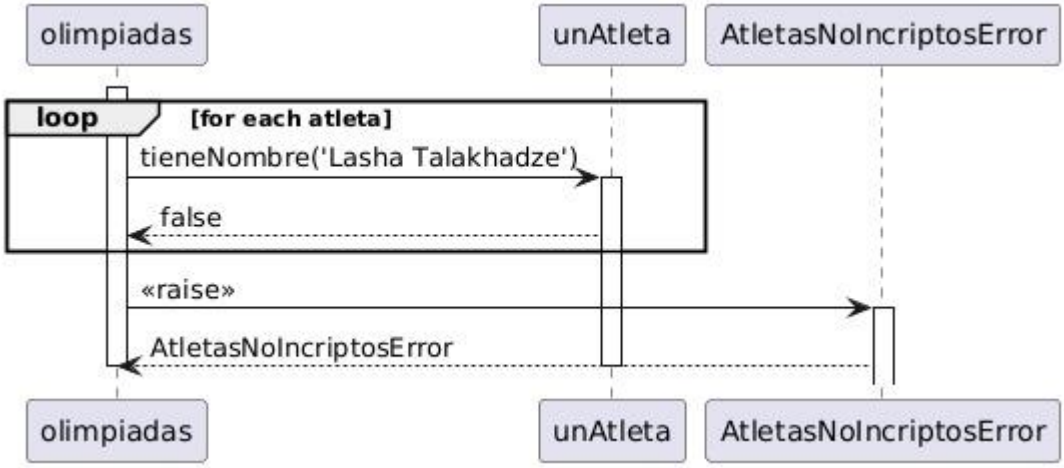

Figura 5: caso **no** feliz para encontrar al atleta deseado, lanza error.

A continuación, se muestra un ejemplo de uso en el cual un usuario le pide a un administrador que le diga cual es el atleta más fuerte/débil.

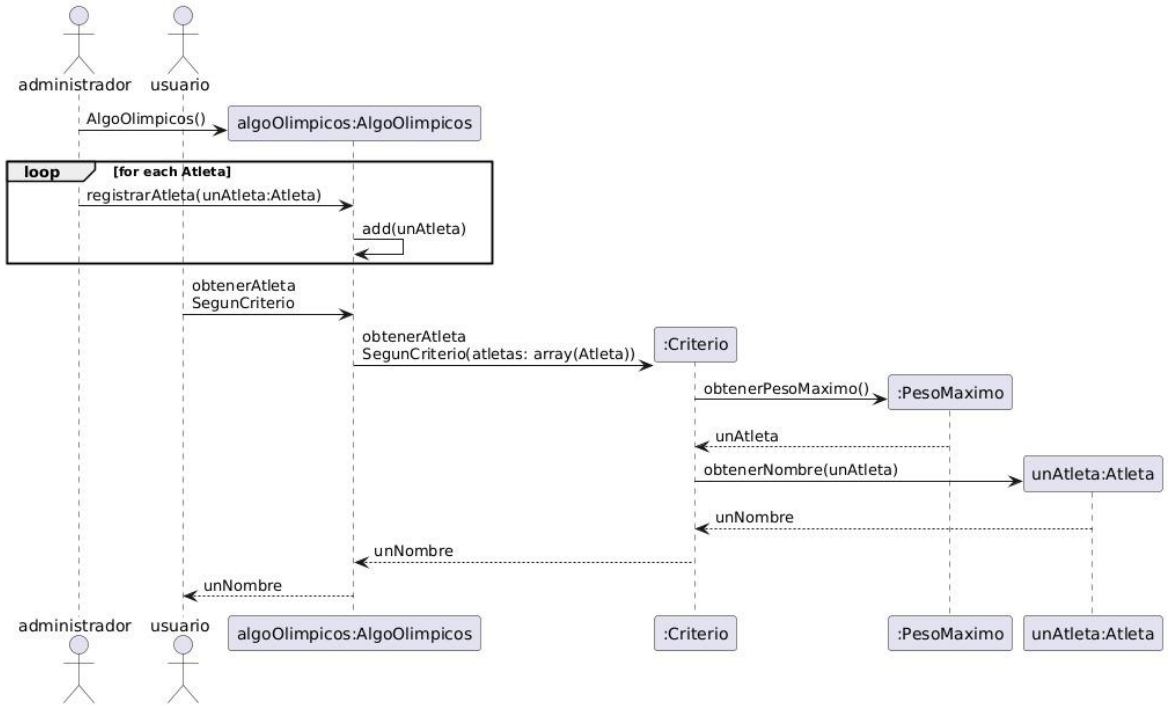

Figura 6: muestra como un usuario le pide al administrador que le diga cual es el atleta más fuerte/débil registrado en el algoOlimpicos.