Trabajo Práctico 1 — Smalltalk

[7507/9502] Algoritmos y Programación III
Segundo cuatrimestre de 2022

Alumno:	DUCHEN, Leonardo
Número de padrón:	104885
Email:	lduchen@fi.uba.ar

Índice

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 penales en Pharo utilizando los conceptos del paradigma de la orientación a objetos vistos hasta ahora en el curso.

2. Supuestos

El partido es solo una cadena de texto, y no se usa para nada más. El único requisito es que no esté vacío (ver más abajo en Excepciones).

No sé cuenta si alguien metió un gol, o si hay un marcador de estos. Es solo para elegir a un pateador de penales en un equipo determinado.

Los nombres de los jugadores pueden estar repetidos (y vacíos), ya que se buscará al que tenga mejores estadísticas.

En caso de empate al querer elegir un pateador, ya sea porque tienen las mismas estadísticas, se elegirá al primero que fue registrado.

En caso de querer elegir un pateador de un equipo no registrado, se elegirá al primero que fue registrado, sin importar del equipo que sea.

Se puede registrar, y pueden patear, jugadores de más de dos equipos. Contrario a todas las reglas de fútbol actuales, si se quiere, se puede hacer.

3. Modelo de dominio

El modelo de dominio se basa alrededor de la clase AlgoPenales, que es la encargada de registrar los jugadores de los equipos y elegir quien va a patear el penal.

AlgoPenales creará un jugador Titular o Suplente, dependiendo lo que se quiera registrar, y se encargará de delegarle mensajes a este jugador. Estos jugadores se guardan en una colección de jugadores.

Un Jugador tendrá sus propias características y podrá calcularlas para poder compararlas al momento de patear con otro jugador más, si es necesario. Es importante la distinción entre un Titular y un Suplente, ya que los dos tienen distintas maneras de calcular sus estadísticas.

Mencionado lo anterior, la experiencia y ánimo de un Jugador, ya sea Titular o Suplente, también influye en los cálculos de las estadísticas.

Por último, al momento de elegir quien patea el penal, AlgoPenales busca en la colección de jugadores y devuelve el de mejores estadísticas.

4. Diagramas de clase

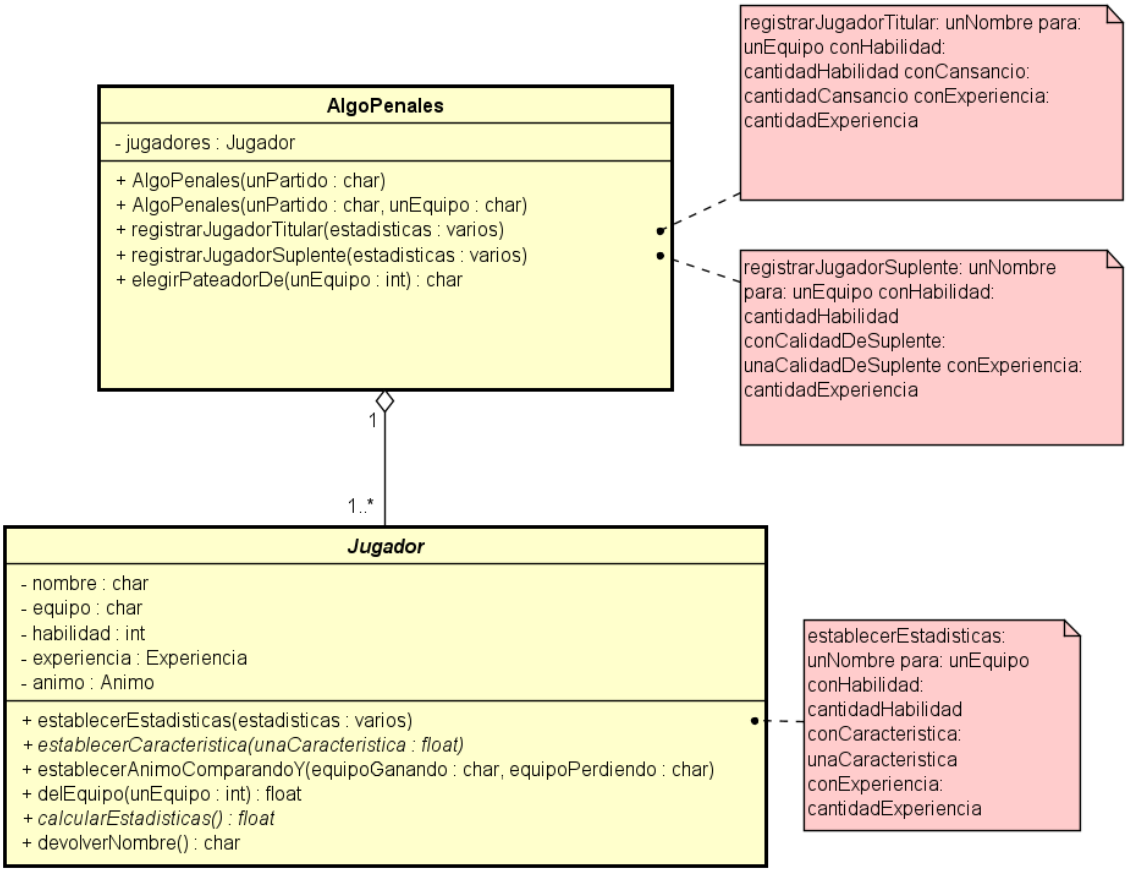

Figura 1: Diagrama del AlgoPenales.

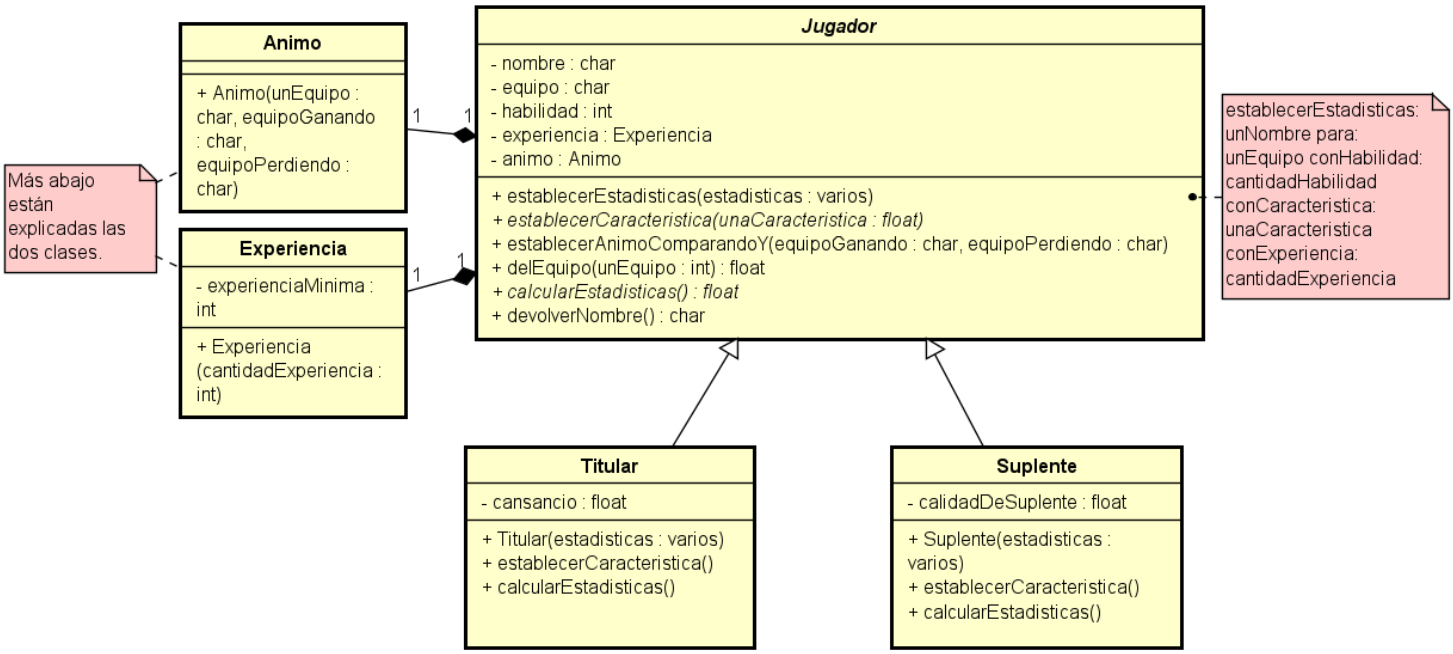

Figura 2: Diagrama del Jugador.

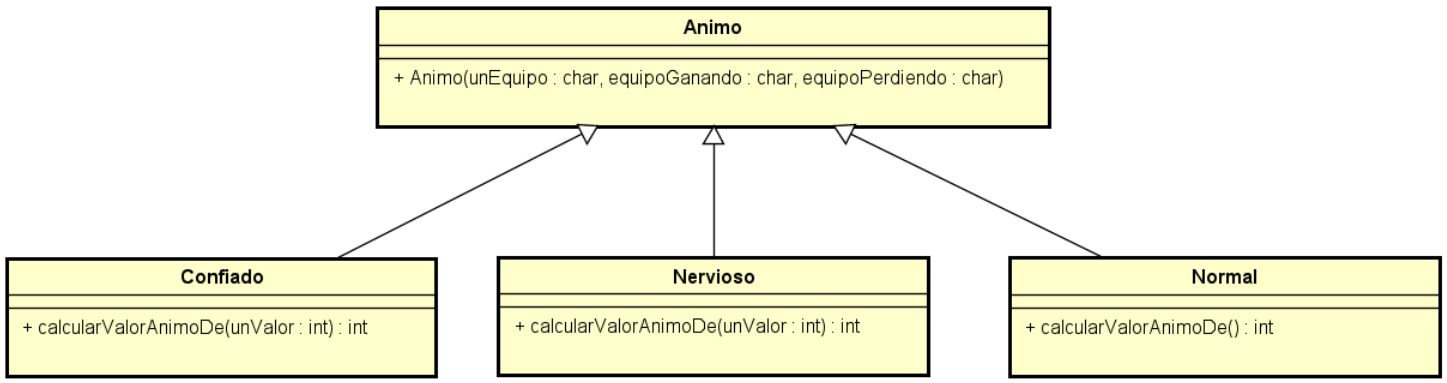

Figura 3: Diagrama del Animo.

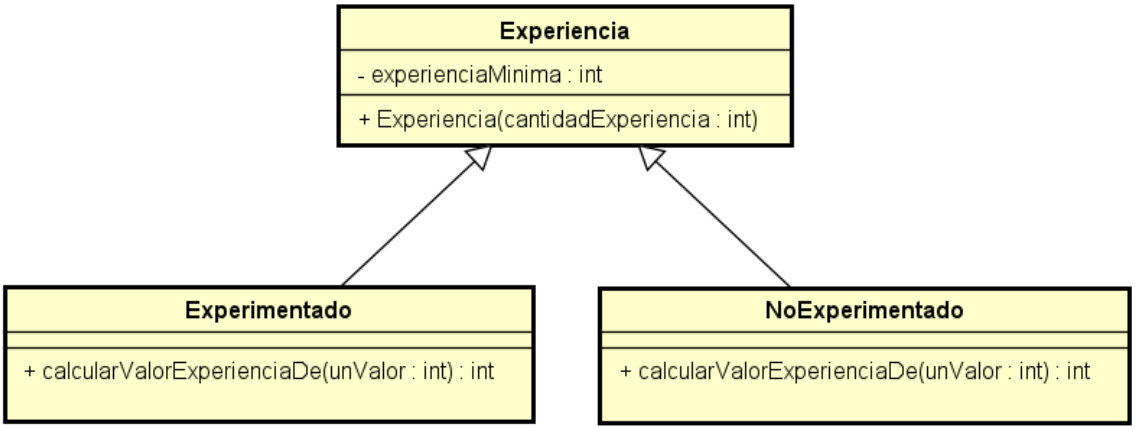

Figura 4: Diagrama de la Experiencia.

5. Detalles de implementación

Delegué varios métodos de la clase AlgoPenales a la clase de Jugador (y esta delega a otras clases). Traté de apuntar a que AlgoPenales tuviera pocas responsabilidades, siendo la única cuando se elige un jugador para patear.

El método en AlgoPenales detecta las mejores estadísticas al calcularlas, mandándole un mensaje al Jugador, y ahí la clase tiene que ver si es del equipo o no. Si es del equipo, procede a calcular normalmente, pero si no lo es, devuelve un -1. Esto es así porque el método comparará a todos los jugadores, y los del equipo indicado tendrán sus valores reales, mientras que los de otro equipo tendrán el valor negativo siendo siempre de menor estadística (no se puede tener valores negativos a través del cálculo de estadísticas porque el mínimo es 0 o tirará una excepción, así que al poner un -1 me aseguro de que así sea siempre).

Todo esto se hace con un “if”, ya que lo preferí antes que tener que crear una clase para usar polimorfismo, donde tengo que delegar varias veces y varias clases se van a conocer entre ellas, con nombres poco descriptivos (¿Clase Equipo y clase NoEsEquipo? No huele bien…) y, probablemente, riesgo de una clase anémica.

Jugador es una clase madre, con sus hijas Titular y Suplente. Estas dos heredan todos los métodos de Jugador, la cual es abstracta ya que tiene un método abstracto (valga la redundancia) para que se encarguen las clases hijas de responderlo.

Uno de los métodos es establecer las estadísticas del jugador, que fue delegado de AlgoPenales, y me pareció mejor que la clase Jugador lo haga así para no violar el encapsulamiento con getters y setters.

Cada clase hija, Titular o Suplente, tiene su propia característica que es establecida como método abstracto desde la clase madre. Cansancio en el caso de Titular, y la calidad de suplente en el caso de Suplente.

Apliqué polimorfismo, con herencia, para crear uno de los objetos de Animo si es necesario Confiado, Nervioso o Normal, si va ganando, si va perdiendo o si va empatando su equipo respectivamente. Al querer saber si el puntaje del ánimo se suma en el cálculo de las estadísticas, uno de los objetos creados calcula el valor y lo devuelve, siendo 1000 si es Confiado y -1000 si es Nervioso. Normal solo devuelve el valor. Hice lo mismo con Experiencia, creando las clases Experimentado, que también devuelve el valor como Normal, y NoExperimentado, que divide por dos el cálculo, si el jugador tiene un número menor a tres (si es igual, significa que tiene experiencia).

Normal y Experimentado pueden resultar clases anémicas, ya que solo devuelven el valor de las estadísticas. Se supone que Normal debería sumar cero y Experimentado dividir por uno, pero al hacer eso sobreviven mutantes.

Usé “ifs” para crear estos objetos, ya que la otra opción era tener los “ifs” en el momento del cálculo y es preferible que estén de forma que, si en el futuro quiero agregar una nueva clase, solo tengo que ir a la clase Animo o Experiencia, y agregar una nueva condición.

6. Excepciones

PartidoVacio: Si al crear AlgoPenales se pasa por parámetro un partido vacío, saltará esta excepción.

EstadisticaNegativa: Cuando se registran jugadores, ya sea titulares o suplentes, saltará esta excepción si algunas de las estadísticas pasadas por parámetro son menores a 0.

JugadorInexistente: Al querer elegir un pateador de un equipo, saltará esta excepción si la colección de jugadores está vacía, ya sea porque no se registraron jugadores titulares o suplentes.

7. Diagramas de secuencia

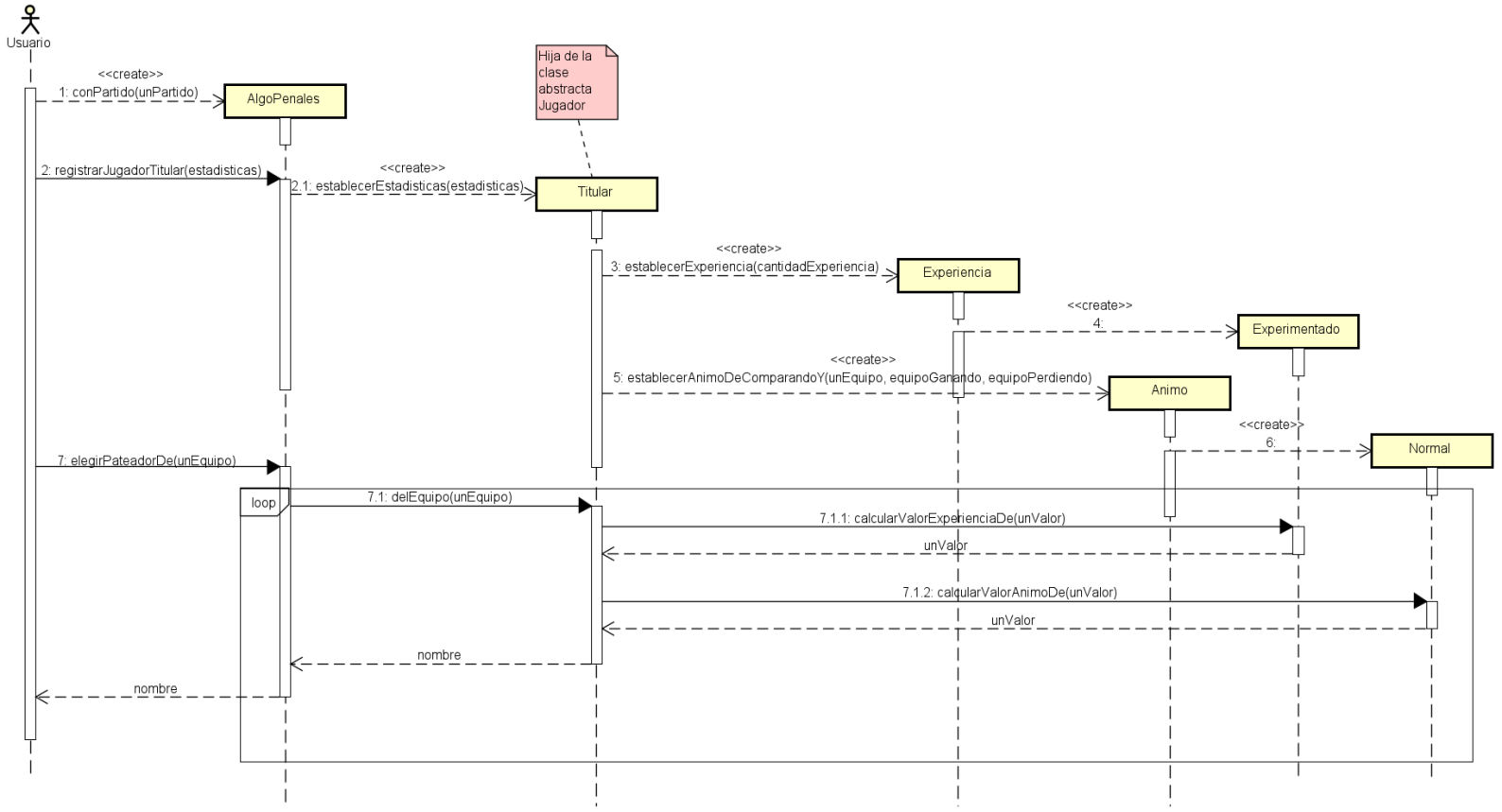

Figura 5: Test01 de la Cátedra.

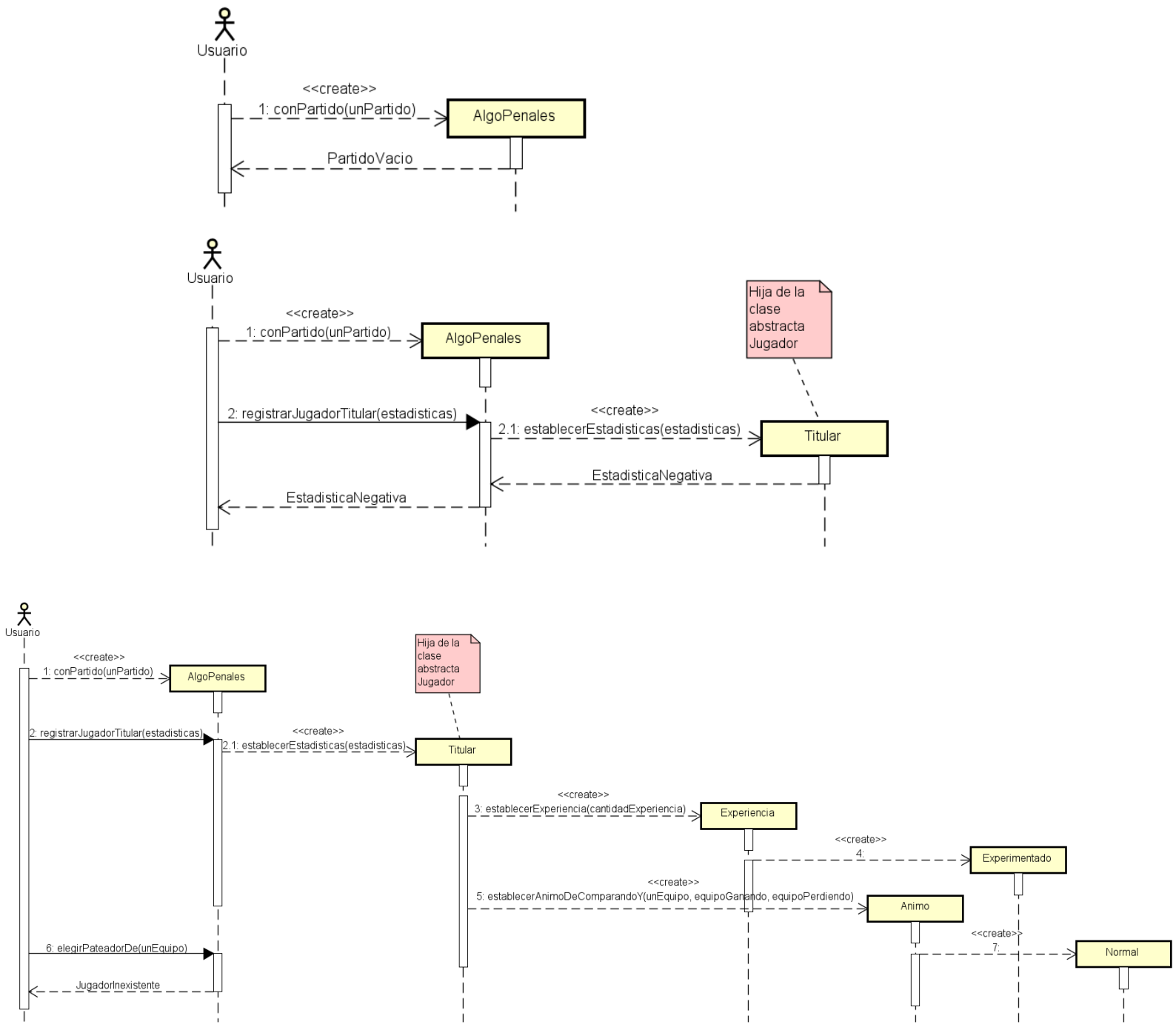

Figura 6: Alternativas del Test01 de la Cátedra con excepciones.

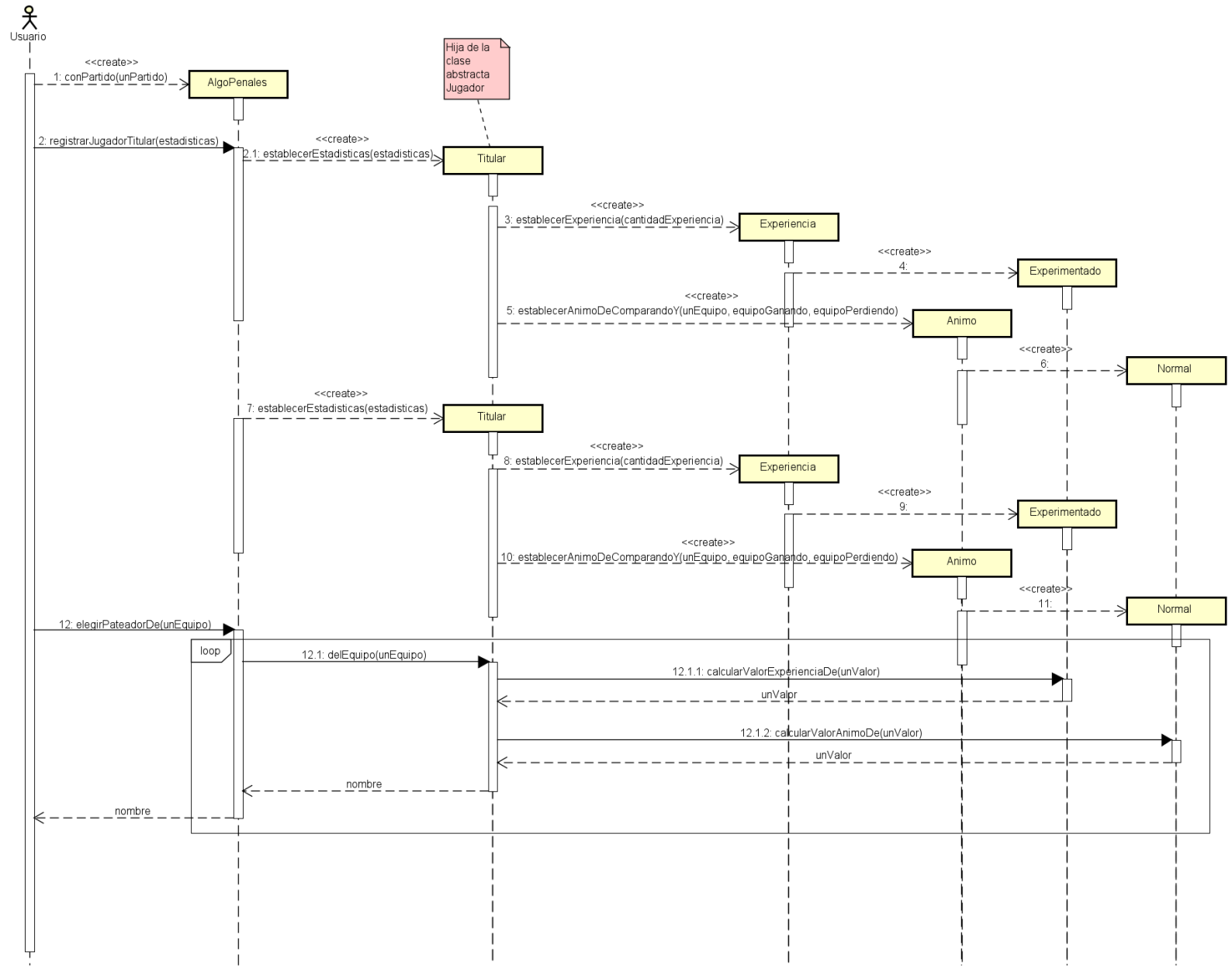

Figura 7: Test03 de la Cátedra.