Ejercicio 5: Penales

Se nos pide diseñar una primera versión de un juego de fútbol que por ahora será sólo de tiros de **penal**.

El diseñador del juego nos explica que las reglas básicas del tiro de un penal son muy sencillas:

"… Cuando el árbitro pita el penal, el delantero patea al arco la pelota con cierta dirección (grilla de 3x3, siendo 0@0 izquierda-abajo, 1@1 centro-medio y 2@2 derecha-arriba, junto a todas las otras combinaciones válidas) y el arquero intenta atajarla lanzándose también en alguna dirección. Si las direcciones no coinciden (por ej. la pelota va a 2@0 y el arquero se quedó en el aburrido centro 1@1) es gol. Si en cambio las coordenadas son iguales, dependerá de si la fuerza del arquero es mayor o igual a la del delantero. Si lo es, el penal es atajado…"

Hasta aquí, todo maravilloso pensamos, *esto se termina más rápido que pasar los tests del Ej. de Stack con ifs*… Pero enseguida nos agregan:

"… Igual, ahora que lo pienso mejor, quizás ese caso básico no ocurra casi nunca, porque lo cierto es que dependiendo de la forma en que patea el delantero, la pelota que se use, y cómo se lance el arquero, las cosas pueden complicarse un poquito, así como para darle gracia…"

A nosotros no nos hizo tanta, pero seguimos escuchando la aclaración:

"… Por ej. en principio los delanteros podrían patear los penales de 2 maneras, ***"a colocar"*** *o* ***"a matar"****. Si remata "a colocar" el jugador sólo tendrá disponible el 50% de su fuerza base, mientras que el que tira "a matar", dispondrá de toda su fuerza. Sin embargo, tirar "a colocar" garantiza que la pelota siempre vaya donde el delantero quiere, mientras que el remate "a matar" puede que a veces no llegue a la trayectoria pretendida, pues no todos somos Messi… ah claro, y también depende de la pelota que se use, no es lo mismo patear con la* ***Jabulani*** *del mundial de Sudáfrica 2010 que era rapidísima y le suma fuerza al delantero si la usa para "matar" que la* ***playera*** *que usamos la última vez que fuimos a Costa Azul que siempre salía volando y apenas si llegaba al arco…"*

-*"Pará.. vas muy rápido… dame un Francia para pensarlo un poco…",* planteamos.
Sin embargo el diseñador continuó inmutable como una fecha, ignorando nuestra perplejidad ante tanta verborragia interminable:

"… El resultado evidentemente también dependerá de cómo se lance el arquero. Si decide elegir una dirección antes de que el delantero patee, y elige bien, tendrá por ej. beneficios en su fuerza, ya que llegará cómodo. Si en cambio espera al remate para lanzarse, y justo le tiraron a matar con la Jabulani, seguro no tiene chance a reaccionar y será gol. Además estos mismos arqueros que esperan por el remate en vez de elegir una dirección antes, son medio tramposos y se adelantan de la línea del arco para lograr beneficios extra y compensar que se lanzan tarde. Cuando les tiran "a matar" sale bien porque las pelotas salen disparadas rápido, pero "a colocar" y encima con la playera, que tarda en llegar al arco, queda expuesto, el VAR lo detecta y es punto para el delantero…"

*"Igual no te preocupes, **por ahora** sólo hace falta que me modeles una ronda de penales de un único tiro y esos dos tipos de pelotas. Encima ya te escribí los tests con todos los casos explicados de forma detallada… Ponete a mirar todo… lo que sí, necesito algo corriendo para antes de las 21:55hs, que hay que mostrárselo a los inversionistas… Ah y en Smalltalk, nada de Unity…"*

Trabajo a Realizar

Deben abrir el archivo de tests que les dejó el diseñador y construir un modelo que los pase (o al menos todos los que puedan!). Dicho modelo debe organizar el conocimiento utilizando clasificación y seguir las heurísticas de diseño vistas hasta ahora en el curso. El modelo resultante deberá representar la realidad del dominio del problema de forma correcta, ser declarativo, no tener código repetido, no utilizar ifs cuando puede ser reemplazado por polimorfismo, la asignación de responsabilidades de forma adecuada, buenos nombres de sus variables, colaboradores, mensajes y categorías , la creación de objetos válidos y completos desde su creación, la no violación del encapsulamiento de los objetos, etc…

Recomendaciones:

1. **Pase los tests** lo más pronto posible y en orden con un modelo simple operacional utilizando ifs.
2. Luego reemplace los ifs por polimorfismo en los casos que considere necesario. No olvide tener presente responsabilidades y esencia de los objetos resultantes.
3. Quite el código repetido que le haya quedado, y mejore la declaratividad de sus métodos. Recuerde que cuenta con un ambiente que posee **refactors automatizados** muy útiles para este paso, como el *rename* o el *extract method.*

Aclaraciones:

- **No se pueden modificar los tests** del diseñador. Luego, no hace falta que le quite código repetido. El trabajo a realizar está en el modelo, no en los tests.
- Mantenga el castellano que se eligió para escribir los tests.

Ayudas:

- Los tests hacen uso de **Points** para describir coordenadas en la dirección tanto del tiro como del arquero. Un Point se puede crear haciendo: *coordenada := 0@3*. Luego se puede enviar el mensaje ***x*** a la coordenada para obtener la coordenada horizontal (0) o el mensaje ***y*** para la vertical (3). No es necesario modelar que las coordenadas sean números enteros porque los tests no lo contemplan.
- Considere utilizar el mensaje **between** para ahorrarse condiciones. Por ej. (5 between: 3 and: 12) retorna true, y (-1@1 between 0@0 and: 2@2) retorna false.