Hacer pasar los tests completando el metodo principal de Penal.

Los primeros tests es agregar 2 asserts, para indicar que tiro y atajada estan en rango.

Para todo el resto, ceberia ser cambiar valores de las varaibles temporables dentro de los ifs, para contemplar cada caso. Por los del angulo, se puede hacer el fueAlAngulo (reemplazar luego el whileTrue por un includes:).

Para los ifs, empezar desde afuera hacia adentro.

Primero modelando las estrategia de tiro, a colocar y a matar. Moviendo todo, y pasando como colaborador externo el penal para luego poder enviarle menjaes tipo registrarXXX para registrar la info necesaria. Dado que son varios los cambios, no sirve retornar nada, sino que hay que enviar mensajes que modifiquen el penal. Varias cosas q eran temporales van a pasar a ser variables de instancia, como los 2 closures que determinan si fue gol o no (el de la fuerza suficiente y el de coordinadas correctas). Conviene que la estrategia colabore con el delantero para poder luego obtener la fuerza (no es necesario recibir desde Penal la fuerza). El mensaje final en el metodo principal seria “delantero patearPelota: pelota contraArquero: arquero conDireccion direccionTiro durante: self”.

Luego seguir con los ifs de las pelotas.

Finalmente, los ifs de la estrategia de atajada (que colabora con el arquero).

El esquema final de colaboraciones deberia ser Penal -> Delantero -> EstrategiaDeTiro -> Pelota -> Arquero -> EstrategiaDeAtajada -> Pena

Hay que prestar atencion a quien conviene que haga que cosa y ver que colaboradores son necesario en cada caso para minimizar complejidad. En la solucion que subí hasta ahora, posiblemente haya responsabilidades no del todo optimas. La pelota por ejemplo no se si es ideal que registre info en los penales, y no mejor dejar todo eso para el paso final del arquero.

Hay bastantes abstracciones que se pueden hacer para quede mucho mas claro el metodo principal y otros metodos.