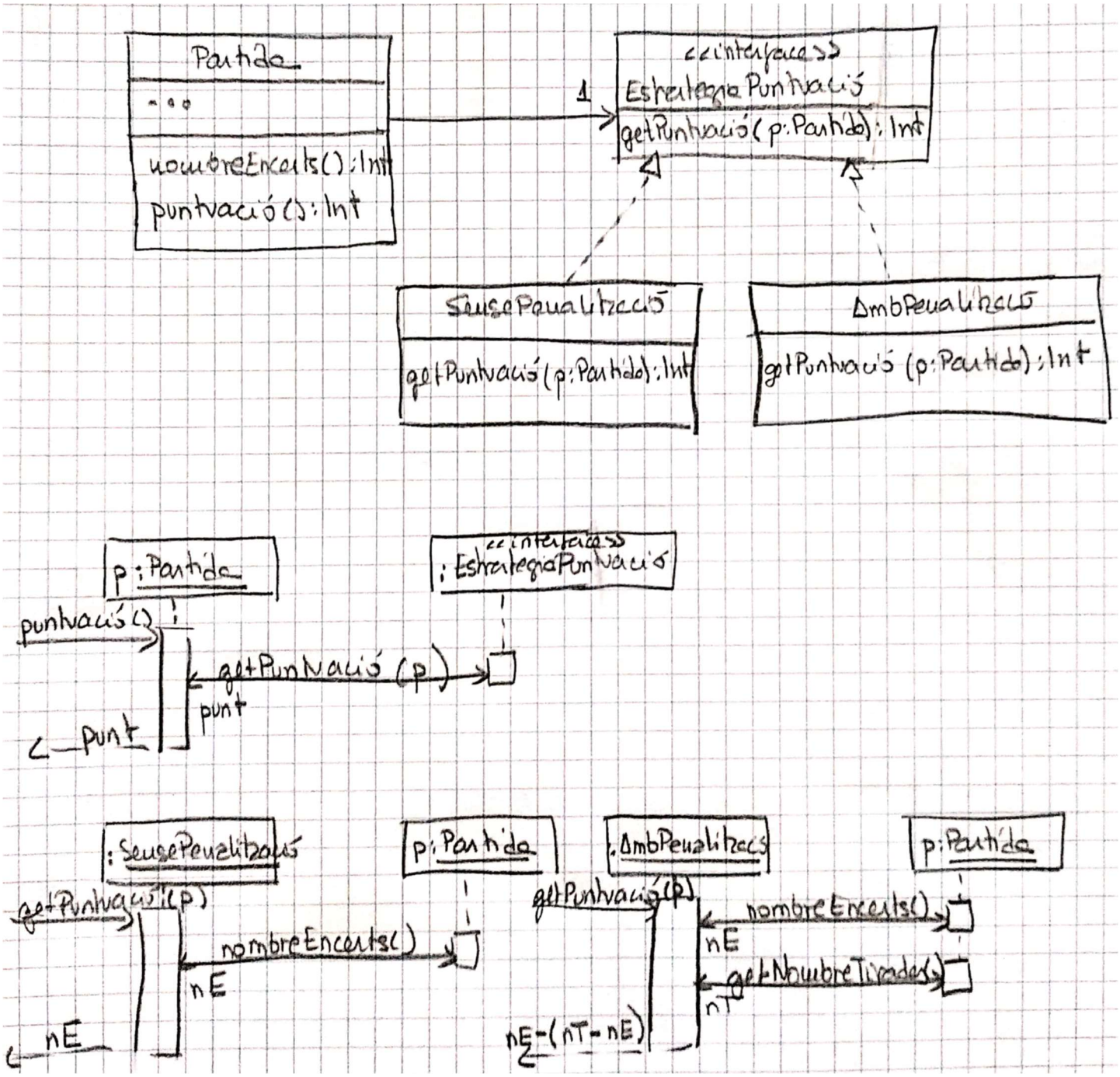
Segon Control AS – Grup 10 **31-05-2021.**

1. Cal modificar la línia de codi número 56. Aquesta línia és la que calcula la puntuació actual. Com hem d'afegir noves formes de calcular la puntuació caldrà modificar aquesta invocació.

2. Aplicaríem el patró estratègia repartint la responsabilitat del càlcul de la puntuació entre la partida i les classes que calcularan la puntuació (estratègies concretes). L'estratègia serà una interfície i les estratègies concretes implementaran la interfície. Un dels avantatges que obtindrem és que la solució compleix amb el principi d'obert-tancat. És a dir, les noves formes de calcular la puntuació suposaran afegir noves classes i operacions però no modificar les existents. A continuació es presenta el diagrama de classes i els diagrames de seqüència corresponents a l'operació de càlcul de la puntuació.



3.

- Iteració 1 (Refactoring):
    - Refactoritzar la Partida per substituir la línia this.nombreEncerts() per la invocació de l'operació getPuntuació sobre un objecte de tipus EstratègiaPuntuació. Els tests no passaran ja que no tenim definit ni l'objecte de tipus EstratègiaPuntuació ni l'operació getPuntuació. L'entorn de desenvolupament ens guiarà per crear l'atribut de tipus EstratègiaPuntuació i la interface EstratègiaPuntuació.
    - Crear la classe SensePenalització que implementi la interface EstratègiaPuntuació i moure el codi de càlcul que hi havia a la Partida a l'operació getPuntuació de SensePenalització. Els tests encara no passaran.
    - Refactoritzar l'operació de setUp del test case per tal de que en crear l'objecte Partida se li passi com a paràmetre un nou objecte de tipus SensePenalització. Els tests continuen sense passar. L'entorn de desenvolupament ens guiarà per crear l'operació constructora de Partida. Haurem de definir el codi de l'operació constructora (el paràmetre que li passarem serà una EstratègiaPuntuació). Els test han de passar. Ara el nostre codi fa el mateix que feia abans del refactoring però amb el patró Strategy aplicat.
- Iteració 2 (Amb penalització): Definim un **nou** test case, TestAmbPenalització per definir el test relacionat amb el mètode de càlcul amb penalització.
    - Definim correctament l'operació setUp() per tal de que en crear l'objecte Partida usi com a paràmetre l'estratègia amb penalització. L'entorn de desenvolupament ens guiarà per crear la classe AmbPenalització que implementa la interface EstratègiaPuntuació.
    - Definim el test per provar la puntuació amb penalització (un test és suficient).
    - Definim el codi de l'operació getPuntuació per tal de que el test anterior passi.
- Iteració 5 (Refactoring): Hem refactoritzat el codi i hem introduït els tests i el codi corresponents a la nova estratègia per calcular la nova puntuació. Ara ens queda refactoritzar els tests. Farem el següent:
    - Refactor de test case TestAmbPenalització i declarem una instancia de AmbPenalització directament en comptes de la instància de la partida i a l'operació setUp() donem d'alta la instància de l'estratègia amb penalització. El resultat serà més curt i fàcil de llegir.
- Iteració 6 (Refactoring): A la iteració anterior hem aconseguit tenir el test de l'estratègia AmbPenalització amb independència de la partida. L'objectiu d'aquesta iteració serà tenir el test de l'estratègia SensePenalització amb independència de la partida i els tests per a les operacions de la partida que treballin amb una estratègia simple (Stub), és a dir, amb independència de les dues estratègies. Finalment també proporcionarem els tests d'integració per garantir que quan tots els components treballen de forma conjunta el parquímetre funciona com esperem.
    - Definim el test case TestSensePenalització i definim un test nou (o utilitzem el que teníem per la partida amb l'estratègia sense penalització) per a l'estratègia SensePenalització. Aquest test case ha de ser anàlog al de TestAmbPenalització. Ara tenim els tests de les classes que implementen l'estratègia separats dels de la partida.
    - Definim una estratègia SimpleStrategy simple (un stub retornant una puntuació de 1) per provar que el codi de la partida funciona. La implementació de l'estratègia s'ha de definir en la carpeta de test ja que el codi d'aquesta estratègia no estarà dins del codi de producció.

Hem de modificar els tests que teníem per a l'estratègia sense penalització per tal de que ara retornin com a puntuació un 1.

- Definir un test case TestIntegration per provar que la partida funciona tal i com s'espera amb les dues estratègies.