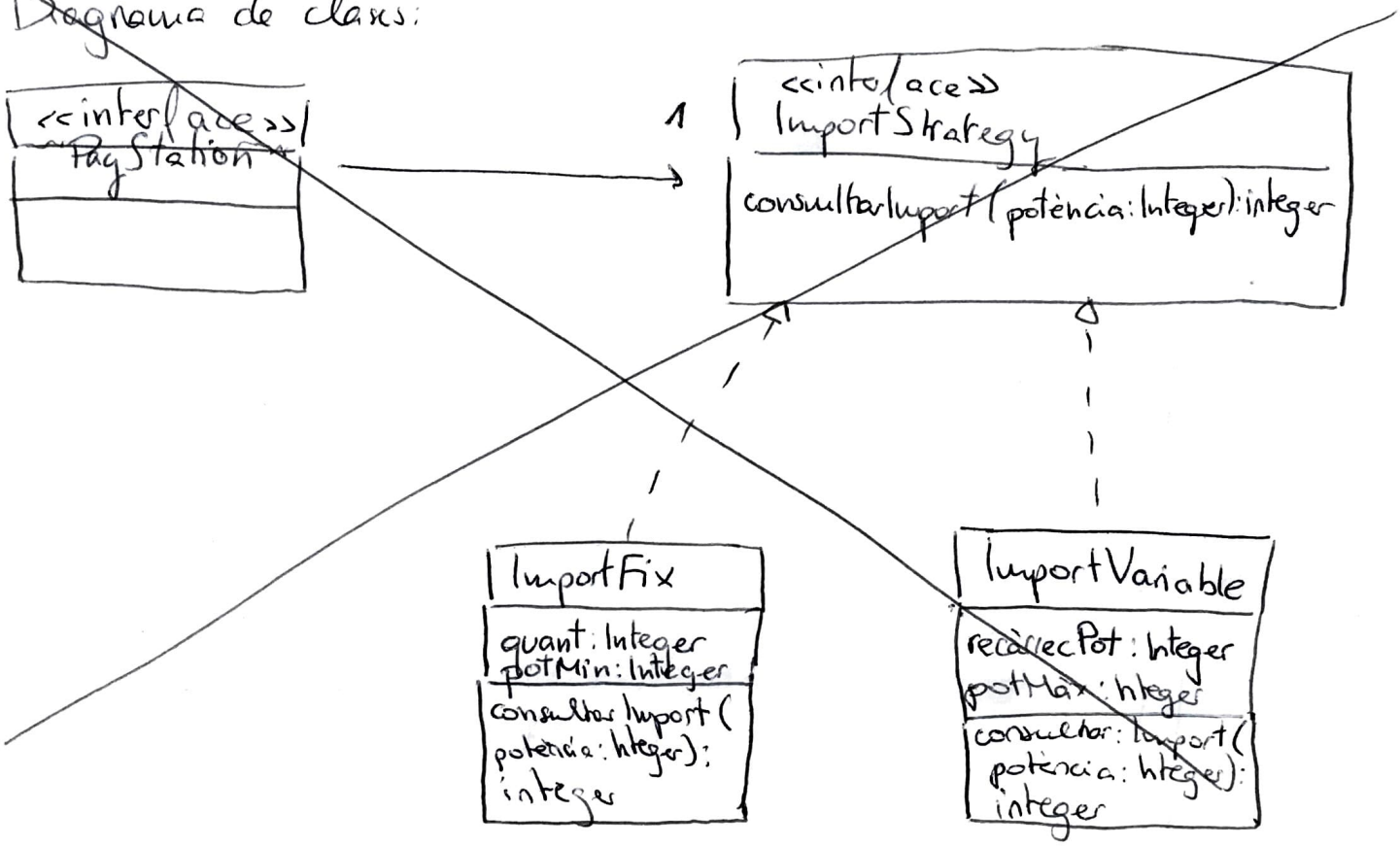
Utilitzarem una solució on aplicarem el patró estratègia.
El component variable de l'aplicació del import quedaria encapsulada en una interface i les classes que implementarien el càlcul concret que han de tenir els parquímetres.

Diagrama de classes:



Benefici: Es fàcil afegir un nou tipus d'import degut a que podem canviar-lo en temps d'execució.

(ImportFixe)

Iteració 1: Definim un nou testCase, TestImportFixe, per definir els tests relacionats amb el mètode del import fixe.

→ Definim el test per provar aquest ~~import fixe~~ import, creant una instància amb una quantitat de 10 i una potència mínima de 200. Apliquem l'import amb una potència de 100. Això retornarà 0 ja que la potència és inferior a la potència mínima.

→ Definim el codi per tal que el test anterior passi.

→ Definim el test per provar aquest import, creant una instància de la estratègia amb una quantitat de 10 i una potència mínima de 200. Apliquem a l'import amb una potència de 300. Això retornarà la quantitat.

→ Definim el codi per tal que el test anterior passi.

(ImportVariable)

Iteració 2: Definim un nou testCase, TestImportVariable, per definir els tests relacionats amb el mètode de l'import variable.

→ Definim el test per provar aquest import, creant una instància amb un recàrrec de potència de valor 10 i una potència màxima de 200. Apliquem l'import amb una potència de 300. Això provocarà el cas excepció, ja que potència serà > que potència màxima.

→ Definim el codi per tal que el test anterior passi.

→ Definim el test per provar aquest import, creant una instància de la estratègia amb un ~~recàrrec~~ recàrrec de potència de valor 10 i una potència màxima de 200. Apliquem l'import amb una potència de 50. Això retornarà el resultat de multiplicar el recàrrec de la potència per la potència

→ Definim el codi per tal que el test anterior passi.

Iteració 3:

→ Definir al testCase TestPayStation ja existent un test nou test

(Refactoring)

Iteració 3: L'objectiu d'aquesta iteració serà tenir els tests per a les operacions del parquímetre que treballin amb una estratègia simple (Stub), amb independència de les anteriors estratègies (import Fix i Variable). També proporcionarem els tests d'integració per garantir que quan tots els components treballen de forma conjunta el parquímetre funciona com esperem.

→ Definir una estratègia StubStrategy simple per provar que el codi de PayStation funciona. Ho definim a la carpeta de test. Aquesta estratègia no estarà dins del codi de producció

→ Definim el test a TestPayStation creant una instància de la estratègia. Aquest test retornarà el mateix valor de potència que haurem introduit.

→ Definim el codi per tal que el test anterior passi.

→ Definir un testCase TestIntegration per provar que els diferents imports funcionen bé i com s'espera.