Rapport

Par Nicolas Delmas et Félix BEAUDOIN

Préface :

Une entreprise nous a contacté afin de rajouter à leur code déjà existant, la possibilité de traiter les nouveaux produits “Conjured” qui vont commercialiser.

Présentation de leur système :

- Tous les éléments ont une valeur `sellIn` qui désigne le nombre de jours restant pour vendre l'article.
- Tous les articles ont une valeur `quality` qui dénote combien l'article est précieux.
- A la fin de chaque journée, notre système diminue ces deux valeurs pour chaque produit.
- Une fois que la date de péremption est passée, la qualité se dégrade deux fois plus rapidement.
- La qualité (`quality`) d'un produit ne peut jamais être négative.
- "Aged Brie" augmente sa qualité (`quality`) plus le temps passe.
- La qualité d'un produit n'est jamais de plus de 50.
- "Sulfuras", étant un objet légendaire, n'a pas de date de péremption et ne perd jamais en qualité (`quality`)
- "Backstage passes", comme le "Aged Brie", augmente sa qualité (`quality`) plus le temps passe (`sellIn`) ; La qualité augmente de 2 quand il reste 10 jours ou moins et de 3 quand il reste 5 jours ou moins, mais la qualité tombe à 0 après le concert.
- Un produit ne peut jamais voir sa qualité augmenter au-dessus de 50, cependant "Sulfuras" est un objet légendaire et comme tel sa qualité est de 80 et il ne change jamais.

Cahier des charges :

- Mise à jour du système : les éléments "Conjured" voient leur qualité se dégrader de deux fois plus vite que les objets normaux.

Mise en place des Kata :

Nous avons tout d’abord commencé par créer tous les tests nécessaire à vérifié le fonctionnement du programme dont certain n'était pas preciser dans la doc tel que :

- La qualité du brie augmente de 2 par update une fois périmé.
- Sulfuras n’est pas obligé d’avoir 80 de qualité.
- La qualité de base d’un produit peut être au dessus de 50 mais sa qualité ne pourra jamais augmenté.

Refactoring :

Nous avons déclaré des variables private static afin de ne pas devoir modifier le code si l'entreprise a un changement sur les exigences (*private static int*) ou les noms des produits qui répondent au attente des produits déjà disponible (*private static String*).

Dans la fonction updateQuality(), nous avons utilisé une liste car selon nous, c'était la méthode la plus simple afin de rajouter des objets spéciaux sans trop modifier la visibilité et la clarté du code. Nous avons utilisé une successions de if pour permettre aux futurs développeurs de comprendre rapidement ce que la fonction fait sans devoir lire le code ligne par ligne (accentué la simplicité de compréhension du projet).

Nous avons créé des méthodes spécifiques à chaque objet afin de rendre le code plus visible et plus simple à réutiliser. Nous avons ensuite appelé ces méthodes dans les if de notre méthode updateQuality(). Nous avons créé une méthode intervalle qui prend en compte les changements de qualité en fonction du vieillissement (utilisé dans la fonction updateBackStage() pour le moment) qui peut être réutilisé pour des futurs changements.