Decorator et Proxy

Ce Td commence un série de TD dans lesquels vous allez implémentez les éléments d'un jeu autour des combats moyennageux. **Important** : vous trouverez un jeu de classes initial qui permettra de démarrer l'implémentation aisément. Ces classent se trouvent dans le répertoire :
Pour cette première version, il s' agit de représenter des soldats qui disposent de matériel d'armement et d'un certain nombre de points de vie.

`http://dept-info.labri.fr/~beurton/Enseignement/ApprocheOjet/2021-2022/src/`

- Il y a au moins deux catégories d'armement: les fantassins et les cavaliers.
- Le matériel d'armement est au moins de deux types : les boucliers et les épées.
- Un soldat peut frapper un coup, methode `int hit()` dont le résultat est la force de ce coup qui dépend de l'armement et de la catégorie du soldat. Ici pour simplifier, on supposera qu'un soldat frappe (`hit()`) ce qu' il doit frapper et dont la méthode n'a pas d'argument (paramètre).
- Un soldat peut parer un coup d'une certaine force (`boolean wardOff(int strength)`) dont le résultat indique si le soldat reste en vie et dont l'efficacité dépend de l'armement et de la catégorie du soldat.

A partir de ces éléments, vous pourrez créer des cavaliers et des fantassins avec épée, bouclier .... et les faire combattre comme des chiffoniers jusqu'à ce qu'ils soient mis hors course (i.e. mort !).

Exercice 1

Un utilisant le pattern Decorator :

1. Elaborer une architecture correspondant au domaine ci-dessus. Evidemment celle-ci doit permettre de créer un soldat armé d'une composition du matériel disponible.
2. Implémenter cette architecture et réaliser des tests unitaires.
3. Modifier votre implementation de manière à pouvoir obtenir une trace de l'enchainement des méthodes `ward-off` and `hit`.
4. Considérons la contrainte suivante : un soldat ne peut avoir deux occurences du même armement, Est-ce que le pattern Decorator a pour rôle d'assurer cette contrainte ?

Exercice 2

1. Améliorer l' architecture obtenue ci-dessus en utilisant le pattern Proxy et en considérant une interface de Soldat qui contient des méthodes spécifiques d'ajout d'armement, i.e. `addShield` et `addSword`.
2. Faire en sorte de satisfaire à la contrainte donnée en 1.4.
3. Modifier votre architecture de manière à ce qu'il soit possible d'ajouter un autre type d'arme.

Exercice 3

Faire en sorte que les armes des soldats se dégradent lors des combats. En d'autres termes, ajouter des spécificités aux objets de décoration et cela de manière transparente pour tout le reste de l'architecture.