TP évalué

Introduction

Consignes générales

- Le tp doit être compilable avec la commande `gradle build`.
- Pour avoir la note maximale il faut que votre programme passe tous les tests unitaires (pré-définis, ne les modifiez pas!).
- Ceci est un travail *individuel*.
- Vous avez les 2h de la session à disposition.
- A la fin de la session: déposer votre TP (zippé) sur Moodle.

Remarque importante: ne modifiez **pas** les tests, il seront utilisés pour évaluer votre travail.

Instructions

- Commencez par télécharger le squellette fourni sur Moodle.
- Un script *build.gradle* est fourni ainsi que l'arborescence (respectant les conventions de Gradle) et un certain nombre de classes.
- A vous de compléter le squelette en écrivant les fonctions demandées dans les exercices.

Description

En préambule, nous attirons votre attention sur le fait que nous avons conscience que ce TP est long et que nous allons en tenir compte. En outre, la note maximale peut être atteinte même si certains tests ne passent pas (si tous passent la note maximale est garantie). Veuillez noter que le TP (squelette), qui vous est fourni comme point de départ, est *presque* compilable, mais ne compile *pas* et c'est à vous de trouver pourquoi.

Le TP est à nouveau basé sur le jeu de rôle (RPG) que nous avons commencé lors des précédentes sessions. Nous avons intégré certains concepts comme les caractéristiques (profil) limitées à intellect, strength et stamina. Il y a comme d'habitude une classe `Player` qui modélise un joueur. Ce dernier peut porter une armure de classe `Armor`, mais la notion de "protection stack" vue dans d'autres TPs n'est pas présente et l'armure absorbe simplement des dégats en fonction des points d'armure (et de son type: `MAGICAL` ou `PHYSICAL`).

Un joueur peut maintenant être équipé d'un (unique) objet magique. Ce dernier va améliorer temporairement les stats (intellect et/ou strength et/ou stamina) du joueur losque il est équipé. Ces stats "bonus" sont retirées du profil du joueur lorsque l'objet est déséquipé.

Le jeu se place dans le contexte de combats dans une arène virtuelle (quelque chose de similaire à un combat de gladiateurs). L'idée est qu'un joueur pourrait équiper un objet pris dans un inventaire d'arène (arena inventory) puis aller combattre dans l'arène. Nous n'allons *pas* modéliser tous les concepts liés à un tel scenario. Cependant, un combat simple, "une passe d'arme" va être testée, grâce à une méthode `attack()` et `wound()`. Il n'y aura pas de méthode de soins dans ce modèle très simplifié.

Exercice 0

Ce n'est pas vraiment un exercice, mais il vous est demandé de trouver ce qui empêche la compilation et de le corriger. Il s'agit de *keywords* (mots reservés du langage Java) liés aux types génériques bornés. Un message d'erreur de compilation assez clair devrait vous mettre sur la piste après avoir essayé de builder le squelette avec la commande `gradle clean build run` ou simplement `gradle build`.

Exercice 1

Compléter le constructeur de la classe Armor de telle sorte qu'une exception soit levée si on essaye d'instancier une armure avec plus que 100 pts d'armure. Il s'agit d'une exception `IllegalArgumentException` qui est une Runtime-Exception "unchecked". Il est *très important* que le message de l'exception soit *exactement* (à l'espace près) celui-ci:

- "Illegal armorVal. Max possible armor value: 100."

En effet certains test(s) sont basé(s) là dessus.

Indice: il suffit de lever (ou lancer) l'exception, pas besoin de la gérer.

Exercice 2

Toujours dans la class `Armor`, compléter la méthode `absorb(...)`. Cette dernière implémente l'absorption des dégats en fonction des points d'armures (champ privé `armorVal`). Le calcul est très simple: selon le type de l'armure (MAGICAL ou PHYSICAL), l'armure va absorber seulement des dégats correspondants à son type. La méthode absorb doit renovyer un objet Damage avec soit les dégats magique soit les dégats physiques réduits selon la formule:

- degats*(1-(armorVal/100))

de manière à réduire les dégats d'un pourcentage correspondant aux points d'armure.

Exercice 3

Compléter la méthode `equip(...)` de la classe `MagicalObject`. Cette méthode permette à un objet magique de "s'équiper" sur un joueur (il peut être plus pratique (par rapport aux génériques) de se placer du point de vue de l'objet pour ce TP, même si bien évidemment c'est la même chose que de dire qu'un joueur équipe l'objet.) Le méchanisme est comme suit:

- un objet magique est *retiré* de l'inventaire d'arène.
- cet objet est *placé* dans un slot (unique) du joueur et modifie temporairement son profil (tant qu'il porte l'objet).
- un booléen est renvoyé par la méthode pour singaler si l'opération a réussi ou pas.

Le code de la méthode unEquip(), i.e., déséquiper, vous est fourni pour vous aider.

Indice: pensez à la mnémonique PECS.

Exercice 4

Complétez les méthodes `attack()` et `wound()` de la classe player pour permettre l'implémentation de combats simples. Les joueurs sont instanciés avec un profil de caractésiques, des points de vie et une classe RPG (PlayerClass)

qui ne peut être que mage ou guerrier pour simplifier. Pour ce modèle simplifié de combats, nous allons considérer que les points de dégats infligés par un guerrier (warrior) sont égaux à ses points de force (strenght) et pour le mage ils sont égaux à ses points d'intellect. Par exemple un player de classe warrior ayant 5 points de strenght verra sa méthode attack() renvoyer un objet Damage avec la valeur 5 dans son champ `physical` et 0 dans son champ `magical`.

La méthode `wound()` va se charger de retirer des points de vie au personnage. Celle ci se basera sur l'armure du personnage et utilisera la méthode absorbe de l'armure pour réduire les dégats selon leur type et selon le type d'armure.

Remarque: nous vons avons proposé des enum (énumérations) pour les types MAGICAL, PHYSICAL et pour les classes RPG WARRIOR, MAGE. Si vous ne voulez pas utiliser d'enums, vous pouvez simplement modifier les classes DmgType et PlayerClass et utiliser des constantes numériques comme 0 et 1. Par exemple, pour `DmgType` on aurait :

```
public class DmgType{
    public static final int PHYSICAL = 0;
    public static final int MAGICAL = 1;
}
```

Cependant, pour la clarté du code et la lisibité, nous recommendons les énums qui semblent bien adaptés, dans ce cas.