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 basé sur le jeu de rôle (RPG) que nous avons commencé lors des précédentes sessions. Il s'agit ici d'une version légèrement modifiée, mais qui reprend néanmoins beaucoup de concepts des précédents TPs. Nous avons une classe `GameCharacter` qui represente un personnage générique. Cette classe devra être *spécialisée* en sous-classes `Warrior` et `Mage` (Nous n'aurons que deux sous-classes par souci de simplification, mais dans un jeu de rôle plus sophistiqué nous pourrions avoir d'autres classes tels que des voleurs (rogue), prêtres (priest) ou encore démonistes (warlock)).

Les classes (Warrior, Mage) sont des *classes* de personnages dans la terminologie des RPG. Ces classes introduisent la notion de profil de caractéristiques (cf. `CharProfile`) a.k.a. "stats" dans le jargon RPG. Pour simplifier, nous auront seulement les stats suivantes:

- intelligence (intellect): augmente les dégats magiques
- force (strength): augmente les dégats physiques
- endurance (stamina): augmente les points de vie max

Dans la classe `CharProfile` se trouvent aussi deux autres caractéristiques de nature un peu différentes:

- xp: représente l'expérience courante d'un personnage
- level: représente le niveau actuel d'un personnage

En effet, au fur et à mesure qu'il complète des quêtes, un personnage gagne de l'xp et eventuellement, fini par monter en niveau (levelUp). Le passage d'un niveau à un autre demande de plus en plus d'xp (progression quadratique). Lorsque un nouveau niveau est atteint le profil d'un personnage sera modifié définitivement.

Les personnages auront leur équipement de protection habituel (implémenté dans les TPs précédents, que nous vous fournissons). Les guerriers équiperont des cotte de mailles et des vestes en cuir résistante au feu alors que les mages porterons des robes de protection magique.

Enfin, pour gérer les types de dégats (physique, magique, etc.) autant que les résistances à ces derniers, nous utiliserons la classe `Damage` introduite dans les TPs précédents et que nous vous fournissons ici.

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 deux *keywords*

incorrects/incorrectement placés (mots reservés du langage Jave).

Remarque: en lisant les messages d'erreurs du compilateur, vous devriez être en mesure de trouver dans quelle classes se situent les problèmes.

Exercice 1

Compléter la classe `MageRobes` sur le même modèle que les classes `ChainMail` ou `FireProofLeatherVest`. Nous rappelons que ces classes *implémentent* toutes l'interface `Protection`.

Votre implémentation doit satisfaire la contrainte suivante:

- Une instance de la classe `MageRobes` ne protège un joureur qui la porte qu'*uniquement* contre des dégats de type magiques **et** électriques.

Exercice 2

Description

Les classes `Warrior` et `Mage` mettent en place un méchanisme d'héritage. Ces deux classes héritent de la classe `GameCharacter` (fournie) et la *spécialise*. En effet, un mage représente une *classe* (au sens RPG) particulière de personnage qui possède ses propres charactéristiques, différentes de celles d'un guerrier par exemple. Comme un mage *est* un personnage, il semble logique qu'il hérite des champs et méthodes de la classe `GameCharacter` (perosnnage "générique").

De plus, les classes `Warrior` et `Mage` implémentent l'interface `CharClass` qui impose la méthode `levelUp()`. Cette dernière prend en argument d'entrée un profil de personnage et en renvoie un autre en retour. En effet, nous allons utiliser pour cela un objet immutable dont la classe vous est fournie et complète: `CharProfile`. Cette dernière modélise le concept de profil de caractéristiques mentionnées en introductions a.k.a. "stats"). (Vous n'avez donc pas besoin de modifier `CharProfile`).

Dans cet exercice, nous allons mettre en place un mechanisme de "levelling" (montée en niveau) d'un personnage. Lorsque un personnage fini une quête, il reçoit des points d'xp et monte en niveau si son niveau d'xp atteint ou dépasse le seuil pour passer au niveau suivant. Toute xp supplémentaire (éventuelle) est perdue.

Remarque: Veuillez noter qu'une classe statique `LevelClass` vous est fournie. Cette dernière contient une constante (statique) maxLevel fixée

à 10. L'idée est que cette valeur va être fixe pour un jeu donné.

La progression d'un niveau vers le suivant se fait selon la formule:

$40x^2 + 360x$, ou x est le niveau *actuel*.

Méthode levelUp

Implémenter la méthode `CharProfile levelUp(CharProfile pr)` pour la class `Warrior` et `Mage`. Cette méthode sera appellée à chaque fois qu'une quête complétée donnera lieu à un passage au niveau suivant. La montée en niveau aura pour effet d'améliorer certaines des caractéristiques (stats) d'un personnage, selon sa *classe* (Warrior ou Mage). L'appel à la méthode `levelUp(...)` augmente le niveau d'un personnage de +1.

- Warrior: à tous les niveaux *multiples de 3* : la force (strength) est doublée et l'endurance (stamina) augmente de 1; sinon (si niveau non multiple de 3), la stamina augmente de 4 et la force de 1.
- Mage: à chaque niveau *pair* : l'intellect augmente de 2 et la stamina de 1, sinon (à chaque niveau *impair*) : l'intellect est doublé et les autres stats inchangées.

Remarque: les stats non mentionnée d'augmentent pas et restent à leur niveau de base (+0).

Méthode completeQuest

Completez la méthode `completeQuest(...)`.

- récupérer l'xp de la quête.
- mettre en place le méchanisme de levelling.
- mettre à jour le profil du personnage.

Remarque: Cette méthode commence par faire appel au comportement défini (fourni) dans la super-classe, à savoir:

- `int questXp = super.completeQuest(q)`