Travaux Dirigés
Langage Orienté Objet / Java

Partie 1 – Classes et packages

The Learning Souls Game

Le but de la série de travaux dirigés que nous allons réaliser sur plusieurs séances est de créer étape par étape un jeu vidéo de type ***Role Playing Game (RPG)*** inspiré de la fameuse série des ***Dark Souls***. Ce type de jeux oppose un (ou des) héro(s) à divers monstres dans des combats.
Les combats sont régis par les actions de chacun et basés sur des séries d'attributs tels que le nombre de points de vie (***life***), la force (***stamina***), les armes équipées (***weapon***), l'armure portée (***armor***), etc. Les actions consomment en général de la force (il n'est par exemple plus possible de donner un coup quand on n'a plus de force). Les coups reçus retirent des points de vie. Le nombre de points de vie retiré à chaque coup dépend à la fois de la puissance de l'arme utilisée, de la protection, et aussi d'une part de chance correspondant à la précision du coup porté dans la frénésie du combat.
Un combat gagné permettra en général de récupérer du matériel (***loot***) comme des pièces d'armure, des armes, ou encore des potions servant à régénérer la force et/ou la vie. Ce matériel sera stocké dans un inventaire lui-même régi par diverses règles que nous verrons en temps voulu.
Le but du héro est de survivre en terrassant tous les monstres.

NB : de manière à bien intégrer les concepts de l'approche Objet, il nous arrivera souvent d'implémenter des fonctionnalités de façon triviale (et non optimale) pour ensuite les améliorer, et bien prendre conscience du gain apporté par une bonne conception orientée Objet.

1. Classe Hero

1.1. Créer la classe **Hero** avec ses différents attributs :

- **name** : une chaine de caractères contenant le nom d'un héro.
- **life** : un entier représentant le nombre de points de vie restants.
- **maxLife** : un entier représentant le nombre maximal de point de vies. Ce n'est pas une constante car elle pourrait évoluer -> dans les RPG la vie maximale est souvent liée au niveau d'entrainement du personnage.
- **stamina** : un entier représentant la force restante.
- **maxStamina** : un entier représentant la force maximale. Ce n'est pas une constante car elle pourrait évoluer -> dans les RPG la force maximale est souvent liée au niveau d'entrainement du personnage.

1.2. Créez les getters & setters pour l'ensemble de ces attributs en réfléchissant bien à leur visibilité (qu'on fixera au minimum des besoins).

1.3. Créez 2 constructeurs

- Un constructeur à 1 paramètre **name** de type String qui permet de créer une instance de **Hero** portant le nom passé en paramètre.
- Un constructeur par défaut (sans paramètre) qui donne automatiquement le nom « ***Gregooninator*** » (ou un autre de votre choix, mais celui-là est mieux…) à l'instance créée.

NB : un héro est instancié avec une vie à 100 et une force à 50.

1.4. Créez une méthode **printStats** qui permet d'afficher les statistiques d'un **Hero** dans la console sous la forme (exacte) :

```
[ Hero ]  Gregooninator   LIFE: 100  STAMINA: 50
```

NB : en Java, le « **\t »** insère une tabulation.

1.5. Modifiez la méthode **printStats** en faisant en sorte qu'elle utilise une surcharge de la méthode **toString()**.

2. Classe LearningSoulsGame

La classe **LearningSoulsGame** correspond au « moteur » de notre jeu : c'est elle qui initiera la création des héros et des monstres, et gérera leurs affrontements.

2.1. Créez la classe **LearningSoulsGame** et implémentez la méthode **main** en faisant en sorte qu'elle crée une (ou des) instance(s) de **Hero**, et lui demande d'afficher ses statistiques dans la console.

3. Classe Hero

3.1. Implémentez le méthode **isAlive** qui renvoie un booléen, et qui permet de savoir si le héro est (encore) vivant (**true**) ou mort (**false**).

3.2. Modifiez la méthode **toSring()** en utilisant **isAlive** de manière à ce que **printStats** affiche maintenant un message de la forme :

```
[ Hero ]  Gregooninator   LIFE: 100  STAMINA: 50     (ALIVE)
```

4. Classe Monster

La classe **Monster** est destinée à fournir les mécanismes de base communs à tous les (types de) monstres que nous allons créer. Ces mécanismes sont en réalité assez similaires à ceux de la classe **Hero**.

4.1. Créez la classe **Monster** en faisant un copier/coller du code contenu dans **Hero**

NB1 : il s'agit d'un exercice, et nous verrons bientôt que la méthode du copier/coller n'est PAS DU TOUT OPTIMALE !!!
NB2 : un monstre est instancié avec une vie à 10 et une stamina à 10.

4.2. On veut pouvoir créer beaucoup de monstres de manière très rapide en leur donnant à chacun un nom différent, mais attribué automatiquement. Nous allons donc faire en sorte que le constructeur par défaut de **Monster** nomme ses instances `Monster_1`, `Monster_2`, … `Monster_N`, N correspondant au nombre d'instances de monstres qui on été créées.
   - Créez une variable de classe **INSTANCES_COUNT** de type entier qui contiendra le nombre d'instances créées à partie de **Monster**.
   - Modifiez le ou les constructeurs pour que ce compteur soit incrémenté à chaque instanciation. NB : normalement, seul le constructeur à 1 paramètre devrait être modifié…
   - Modifiez le constructeur par défaut pour qu'il nomme correctement les instances de **Monster** en utilisant **INSTANCES_COUNT**.

5. Classe LearningSoulsGame

5.1. Modifiez le **main** et créez plusieurs monstres en utilisant les différents constructeurs. Vérifiez le résultat en affichant les statistiques des monstres qui viennent d'être créés. Exemple :

```
[ Hero ]    Gregooninator     LIFE: 100   STAMINA: 50 (ALIVE)
[ Monster ] Studentatort      LIFE: 10    STAMINA: 10 (ALIVE)
[ Monster ] Monster_2         LIFE: 10    STAMINA: 10 (ALIVE)
[ Monster ] Monster_3         LIFE: 10    STAMINA: 10 (ALIVE)
```

6. Package lsg

Jusqu'à présent, nous avons mis toutes nos classes dans le même package (qui ne possède pas de nom, c'est donc le package par défaut). Dans un projet de grande envergure comme le nôtre, il est nécessaire de faire preuve de plus d'organisation.

6.1. Créez un package nommé **lsg** et mettez-y l'ensemble des classes créées.
6.2. Dans **LearningSoulsGame**, testez l'appel (direct) à **isAlive** en changeant sa visibilité (**private, friendly, protected, public**) au sein de la classe **Hero.**

Quelle est la visibilité optimale pour **isAlive** dans cette configuration ? Pourquoi ?

7. Packages characters

7.1. Pour une encore meilleure organisation, créez un sous-package **lsg.characters** et mettez –y les classes **Hero** et **Monster.**
7.2. Laissez **LearningSoulsGame** dans **lsg**.
7.3. Dans **LearningSoulsGame**, testez l'appel (direct) à **isAlive** en changeant sa visibilité (**private, friendly, protected, public**) au sein de la classe **Hero.**

Quelle est la visibilité optimale pour **isAlive** dans cette configuration ? Pourquoi ?