Mini Jeu

Objectifs

Nous allons mettre en pratique ce que nous avons appris en programmation orientée objet.

- Encapsulation, création de classes, de propriétés
- Héritage, polymorphisme

Le but de cette activité est de créer un mini jeu où nous allons pouvoir utiliser un dé pour vaincre des monstres. Dans la première version du programme, tout va être automatique et pseudo-aléatoire[1].

Rendu

Un dossier nommé « Mini Jeu – NOM Prénom » ou une archive de type zip « Mini Jeu – NOM Prénom.zip » contenant.

- Diagramme de classes.
- Dictionnaire des données (optionnel).
- Code source (fonctionnel).

Instructions

- L'IHM, en plus d'afficher l'état et les actions, doit permettre de définir son personnage (nom et genre).
- Le héros (ou l'héroïne) doit arriver au sommet du donjon.
- A la fin on doit afficher combien il a vaincu de monstres standard et élite.
- Le héros démarre avec 100 points de vie, un monstre standard avec 1D6[2] points de vie, un monstre élite avec 2D6[3] points de vie.
- Si dans la salle il y a un monstre aléatoire (standard ou élite), le héros attaque le monstre ; puis si le monstre a survécu il attaque à son tour le héros et ceci jusqu'à ce que mort d'un des deux surviennent.
- Une attaque du héros sur un monstre consiste en un jet de dé pour chacun des protagonistes. Si le dé du héros est supérieur ou égal au dé du monstre, alors celui-ci est touché. Sinon, le monstre esquive l'attaque, rien ne se passe et c'est au tour du monstre d'attaquer.
- L'attaque du monstre standard sur le héros est similaire, mais à la différence que le jet du monstre doit être strictement supérieur au jet du héros.
- Lorsque le héros va subir des dégâts, il lève son bouclier représenté par le jet d'un nouveau dé. Si le résultat de ce dernier est inférieur ou égal à 2 (donc 2 chances sur 6), alors le héros à réussit à parer. Le bouclier de base permet d'enlever 2 points des dégâts reçus (le joueur Player recevra un nombre de point de dégâts compris entre 0 et 5).

---

[1] Pour générer un nombre pseudo-aléatoire, il faut instancier un objet **Random**.

[2] 1D6 signifie l'utilisation d'un nombre pseudo-aléatoire compris entre 1 et 6 inclus qui représente un jet d'un dé à 6 faces.

[3] Le nombre de 'dés' virtuel qui vont être jetés sont identifié par le premier nombre (**2**D6). Le second chiffre (2D**6**) correspond, quand à lui, au **nombre de face** du dé. Pour 2D6, le nombre pseudo-aléatoire sera compris entre 2 et 12 inclus.

- Les dégâts sur les monstres ou le héros sont calculés en faisant la différence entre les deux dés. L'arme de base portée par le héros n'ajoute pas de points de dégât.
- L'attaque d'un monstre élite est la même que celle du monstre facile, sauf qu'il à 16 % de chance d'utiliser une attaque magique. L'attaque magique est imparable. Lorsqu'elle a lieu, on réalise un jet de dé et s'il est différent de 6 alors le héros perçoit des dommages équivalent à la valeur du dé multiplié par une valeur forfaitaire, disons 5.
- Les Items seront portés par le héros.

Quelques éléments techniques.

La classe (**Player**) qui représente notre héros/héroïne :

- Possède une propriété en lecture seule qui contient les points de vies ; ceux-ci sont initialisés dans le constructeur.
- Possède une propriété en lecture seule permettant de savoir si le joueur est vivant (**IsAlive**), et fait un test sur le nombre de points de vie.
- Possède une méthode **Attack**, prenant en paramètre un ennemi.
- Possède une méthode **Damage** qui prend en paramètre un entier avec la valeur des dégâts subits

La classe de monstre standard (**Monster**), représentant les ennemis :

- Possède une propriété en lecture seule qui contient les points de vies (stockée dans la variable **Health**) ; ceux-ci sont initialisés dans le constructeur.
- Possède une propriété en lecture seule permettant de savoir si le monstre est vivant.
- Possède une méthode **Damage** qui prend en paramètre un entier avec la valeur des dégâts subis.
- Possède une méthode **Attack**

La classe des monstres d'élite (**MonsterElite**) possède :

- Hérite des fonctionnalités d'un monstre de base et en ajoutera.
- Possède une propriété qui modifie le comportement de la méthode **Attack.**

La classe de dé (**Dice**) :

- Permet de faire un tirage pseudo-aléatoire, comme un dé classique.
- Possède une méthode **Roll** qui renvoie un entier

La classe des objets trouvés par le joueur (**Item**)

- Possède un nom (**Name**).

La classe du donjon (**Dungeon**)

- Contient plusieurs étages (**Floor**) défini aléatoirement.
- Chaque étage possède 0-1 monstre (standard ou élite) et chaque monstre possède 0-1 item.

Pour chacun des tours, il faudra afficher l'action qui est en train de se dérouler quelques exemples :

- "Prêts pour le combat ? C'est parti !"
- "Bravo !!! Vous avez tué 8 monstres faciles et 10 monstres difficiles."

- "Snif, vous êtes mort..."
- "Vous arrivez à l’étage numéro 2 où se trouve un monstre d’élite !"
- "Vous subissez 5 points de dégât, il ne vous reste plus que 45 points de vie."
- "Le monstre standard subit 3 points de dégât, il en meurt."
- "Appuyer sur 0 pour continuer l’exploration du donjon."

Pour aller plus loin, vous pouvez ajouter.

Un boss de fin (**MonsterBoss**) avec les règles suivantes :

- Le boss de fin possède 100 points de vie.
- Tous les 3 tours, le boss de fin peut éviter l'attaque du joueur grâce à son bouclier magique.
- Le joueur peut éviter l'attaque du boss de fin grâce à son bouclier (que l'on a déjà créé dans cette activité !).
- Il possède une propriété **IsAlive** en lecture seule qui retournera vrai si le nombre de points de vie est supérieur à 0.

Pour aller encore plus loin, vous pouvez ajouter

- Des potions de soins (parmi les **Item**).
- Des armes
- Toutes autres idées

Annexes

- Le code source devra pouvoir être compilé dans le site https://www.onlinegdb.com