TRAVAUX PRATIQUES
MMORPG

Vous allez dans ce TP découvrir d'autres notions de programmation Python telles que l'héritage. Il est important à chaque fois de préparer le schéma UML de votre classe avant d'entamer toute écriture de code.

1. Le contexte

le but de ce TP va être de créer et gérer un joueur de Massive Multiplayer Online Role Playing Game (MMORPG) et ses personnages dans le jeu.

2. la classe Personnage

Vous allez créer une classe personnage représentant un avatar du joueur. Cette classe possédera les attributs suivants :

- un pseudo,
- un niveau,
- un nombre de points de vie,
- une initiative

Cette Classe possédera aussi deux constructeurs. Le premier permet de créer un personnage de niveau 1 ayant un nombre de points de vie et une initiative à 1 aussi. Il faudra préciser le pseudo lors de la création. Le second permettra de créer un personnage avec un niveau donné, le nombre de points de vie et l'initiative étant égal au niveau. Cette Classe devra aussi posséder les méthodes suivantes :

- une méthode attaque d'un autre personnage. Cette méthode testera l'initiative de chaque personnage. Celui qui a la plus grande initiative frappe le premier et l'autre personnage perd un nombre de point de vie égal au niveau de l'attaquant. S'il n'est pas mort (0 Pv ou moins), le second personnage occasionne à son tour des dégâts de la même façon que le premier. Si les deux personnages possèdent la même initiative, ils tapent en même temps et s'occasionnent réciproquement des dégâts.
- une méthode combat. Elle permet de mener toutes les attaques nécessaires à la mort d'un des deux assaillants. Cette méthode utilisera un boucle
- la méthode soigner qui permet de restaurer les points de vie d'un personnage jusqu'à son niveau.

Vous testerez cette classe dans un programme principal mettant en oeuvre deux personnage menant un combat.

3. La notion d'héritage

Nous allons à présent distinguer deux type de personnages différents, les Guerriers et les Mages.

3.1. **Le Guerrier.** Un Guerrier est un personnage. Il possède les mêmes attributs que la classe personnage. Vous créerez les mêmes constructeurs que pour le personnage avec les règles de calculs suivantes :

- pv = niv x 8 +4
- init = niv x 4 +6

Remarque : Un Guerrier étant un personnage, il est nécessaire dans les constructeurs de Guerrier d'instancier un personnage servant de support. Cela se fait en utilisant le mot clé ***super().__init__(arguments...)*** *qui fait appel à l'un des constructeurs de la classe Personnage en fonction des arguments passés entre parenthèses. On peux aussi utiliser la méthode explicite* ***Personnage.__init__(self, arguments...)***

3.2. **Le Mage.** Un Mage est un Personnage possédant en plus des attributs de Personnage un attribut spécifique, la Mana. Créer les mêmes constructeurs que pour la classe Personnage avec les règles de calcul suivantes :

- pv = niv x 5 +10
- init = niv x 6 +4
- mana = niv x 5

3.3. **utilisation des classes Guerrier et Mage.** Dans le programme principal, créer un guerrier et un mage et faite les combattre. Comment cela peut il être possible ?

3.4. **modification de méthode.** Dans les classes Personnage, Guerrier et Mage, vous allez créer une méthode **degats** qui retourne le nombre de dégâts occasionnés par chaque attaque des personnages avec les règles suivantes :

- Personnage : dégâts = niveau
- Guerrier : dégâts = niveau x 2
- Mage : dégâts = niveau + 3 tant qu'il a du mana et niveau quand il n'en a plus. chaque attaque coute 4 de mana

Modifier en conséquence la méthode attaque de Personnage pour faire appel à la méthode dégâts lors de la résolution de l'attaque. Modifier la méthode Combat afin d'afficher après chaque attaque le nombre de points de vie des combattants. Que constater vous si vous faites combattre les différents types de personnage ?

4. La classe joueur

Un joueur possède un nom, et un ensemble de personnages stocké dans une liste. Un constructeur permettra de donner le nom du joueur et le nombre maximum de personnages du joueur. La Classe Joueur possédera les méthodes suivantes :

- une méthode permettant d'ajouter un personnage à la liste des personnages du joueur. le personnage sera ajouté s'il le maximum n'est pas atteint
- trois méthodes permettant d'accéder à l'un des personnages du joueur en précisant respectivement le numéro du personnage, le pseudo du personnage, ou en fournissant un personnage. Pour cette dernière méthode il faudra implémenter dans **Personnage** la méthode d'égalité **__eq__**, méthode appelée lorsque que vous faite la comparaison de 2 éléments en utilisant l'opérateur ==
- trois méthodes permettant d'éliminer un personnage de la liste en précisant respectivement le numéro du personnage, le pseudo du personnage, ou un personnage

Tester aussi cette classe dans le programme principal.