Cégep Ste-Foy
Programmation objet 1– 420-W20-SF

Travail Pratique 2

Préparé par
Karine Filiatreault et Pierre Poulin
Modifié par
Raphaël Paradis

1 Résumé

Pour ce travail pratique, nous allons concevoir un petit jeu de rôle très simplifié.

2 Conditions de réalisation

Valeur de la note finale	Type	Remise
15 %	En équipe	19 mars avant minuit

3 Spécifications

3.1 Classes à coder

Vous aurez **au minimum** les quatre classes suivantes à produire pour ce travail (en plus des classes de tests pertinentes). Le diagramme de classe **n'est pas fourni** pour ce travail. Je vous recommande donc fortement de réfléchir d'abord par vous-même à votre architecture (et à faire un diagramme sommaire), puis d'implémenter votre solution par la suite.

La classe **`Personnage`**

Description des attributs :

- **nom :** Le nom du personnage.
- **classe :** La classe du personnage : une énumération (voir plus bas pour les détails).
- **arme :** Le type d'arme que le personnage utilise : une énumération (voir plus bas).
- **nbPotions :** Le nombre de potions magiques que le personnage possède. Le personnage n'a pas de potion au départ.
- **sorts :** La liste de sorts que le personnage peut utiliser (voir la description de la classe **Sort** plus bas). À noter que seul le mage peut avoir des sorts et que tout autre type de personnage aura une liste de sort *null*.
- **degatsDernierCombat :** Un dictionnaire contenant le numéro du tour en clé et le différentiel de dégâts pour ce tour en valeur. Le différentiel est calculé ainsi : dégâts infligés – dégâts reçus. Ce dictionnaire est réinitialisé entre chaque nouveau combat.

De plus, la classe personnage doit contenir les statistiques de ce dernier. **C'est à vous de trouver une solution efficace en suivant les principes de programmation vu en classe.** Voici la description des attributs :

- **ptsVie :** Son nombre de points de vie.
- **ptsVieMax :** Son nombre de points de vie maximum.
- **ptsAttaque :** Son nombre de points de force (influençant ses attaques).
- **ptsDefense :** Son nombre de points d'armure (influençant sa résistance aux attaques).
- **ptsExperience :** Le nombre de points d'expérience du personnage. Un personnage commence toujours avec **zéro point** d'expérience.

Le constructeur d'un personnage recevra les paramètres suivants : nom, classe, sorts et arme. Le reste doit tout de même être initialisé.

Chaque personnage doit avoir une classe (pas de lien avec la POO 😉) parmi les suivantes :

Classes	Nb. Points de vie	Nb. Points d'attaque	Nb. Points d'armure
Archer	3-18	2-16	1-6
Mage	2-12	3-18	1-4
Guerrier	3-30	1-10	2-12
Assassin	2-16	4-16	1-8
Moine	4-24	3-12	1-6

Donc lorsqu'on choisit une classe lors de la création de notre personnage, on a automatiquement des points de vie aléatoires alloués (entre les valeurs minimum et maximum indiquées pour chaque classe, comme si vous rouliez un dé).

De plus, selon le type d'arme choisi, les points d'armure sont modifiés. :

No. de l'arme	Arme	Points d'armure supplémentaires
0	Aucune (mains nues)	0
1	Épée et bouclier	5
2	Épée à deux mains	2
3	Arc et flèches	1

Ces points ne devraient pas être additionné à l'attribut ptsDefense des statistiques. Je vous recommande de les garder ailleurs (par exemple, en créant un autre attribut dans les statistiques)

➔ Le magicien ne peut pas avoir d'arme !!

Description de **quelques** méthodes publiques - il sera probablement nécessaire de *découper votre code en ajoutant des méthodes privées*.

- public bool EstMort()
  Retourne vrai si le personnage n'a plus de points de vie.
- public void Attaquer(Personnage ennemi)
  Permet au personnage courant d'essayer de faire subir des dégâts (c'est-à dire de faire baisser sa vie) au personnage reçu en paramètre.

Premièrement, pour déterminer si le personnage atteint la cible lors de l'attaque, on *lance* un dé à vingt faces. Si le résultat est supérieur aux points d'armure de l'ennemi, le personnage atteint la cible. Sinon, rien d'autre ne se passe.

Secondement, pour déterminer les dégâts, on calcule la force réelle d'une attaque en additionnant les points d'attaque du personnage aux points de dommage de son arme. Notez que, puisqu'un magicien n'a pas d'armes, on additionne plutôt les points de dommage d'un de ses sorts choisi aléatoirement dans sa liste de sorts.

Voici la description des armes :

No. de l'arme	Arme	Points de dommage
0	Aucune (mains nues)	1
1	Épée et bouclier	3
2	Épée à deux mains	5
3	Arc et flèches	4

N'oubliez pas d'affecter le dictionnaire **degatsDernierCombat** à chaque attaque!

- **public void BoirePotion()**
  Décrémente le nombre de potions magiques de 1, augmente le nombre de points de vie de 2-8 points si la potion a été bue. Le personnage ne peut jamais avoir plus de points de vie que lors de sa création initiale.

D'autres méthodes publiques seront nécessaires pour mener le combat. Le principe de base pour un combat est le suivant :

Le nombre de points de dommage subis correspond à la force de l'attaque reçue diminué par le nombre de points d'armure de base (donc sans compter celle procurée par l'arme) du personnage attaqué. S'il y a dommage (> 0), celui-ci est soustrait du nombre de points de vie du personnage. Les points de vie ne peuvent pas aller dans le négatif.

La classe **`Sort`**

Description des attributs :

- **nom :** Le nom du sort : une énumération.
- **ptsDegatsMin :** Le nombre de points de dégâts minimum du sort.
- **ptsDegatsMax :** Le nombre de points de dégâts maximum du sort.

Le constructeur d'un sort prendra seulement le nom du sort en paramètre et devrait donner les bonnes valeurs de dégâts selon celui-ci. Note : vous devrez trouver une solution inventive pour suivre de bons principes de programmation. N'hésitez pas à venir me voir si vous lisez ceci!

Sorts	Nb. dégâts
Boule de feu	2-16
Missile magique	3-12
Foudre	1-20

Description de la méthode publique :

- public int ObtenirDegats()
  Retourne le nombre de dégâts aléatoires générés par le sort entre le nombre de points minimum et maximum (inclusivement) du sort.

La classe **`GestionJeu`**

Description des attributs :

- joueur : le joueur doit être créé au début du jeu avant de faire le premier niveau.
- ennemi : l'ennemi changera à chaque niveau visité par le personnage joueur. Voir le tableau des niveaux plus bas 😊
- noTableau : le niveau auquel le joueur est rendu. Commence à zéro.
- noTour : le numéro de tour du combat courant. Commence à un.

Le constructeur prendra le joueur et l'ennemi en paramètres.

Description des méthodes publiques :

- public bool EngagerCombat()
  Permet à deux personnages de combattre. Retourne vrai si le joueur a gagné le combat. Sinon, retourne faux.

  Un combat est constitué de plusieurs tours. Pour chaque tour, on commence par lancer un dé (6 faces) afin de déterminer lequel des deux personnages attaque le premier. Si le dé est pair, alors le joueur est le premier à attaquer et l'inverse sinon. À noter que les deux personnages vont s'entre-attaquer à chaque tour.

  Les personnages s'attaquent à tour de rôle jusqu'à la mort d'un des deux.

- public void RecueillirRecompense()
  Affiche au joueur la récompense à laquelle il a droit s'il a gagné un combat. La récompense est déterminée de façon aléatoire pour un nombre entre 1 et 100. Voici les possibilités et leurs effets :

Nombre min (incluse)	Nombre max (incluse)	Message affiché	Effet
1	30	Vous avez trouvé une potion magique!	Augmente de 1 le nombre de potions magiques.
31	60	Vous gagnez de l'expérience supplémentaire!	Augmente de 25 le nombre de points d'expérience.
61	100	Désolé, vous ne gagnez rien.	Aucun

La classe **`Program`**

La classe **`Program`** doit instancier tout ce qui nécessaire pour permettre de créer une « fiche de personnage » et de tenter de parcourir les 4 niveaux du jeu. Vous devrez poser des questions à l'utilisateur et vous servir des réponses pour créer les objets appropriés. Le programme ne doit jamais planter même lorsque l'utilisateur ne répond pas correctement aux questions posées (ex. on demande un choix entre 1 et 3 et l'utilisateur entre -7 ou 5) mais plutôt afficher un message d'erreur et reposer la même question jusqu'à ce que la réponse soit acceptable.

Quelques spécifications supplémentaires pour la création d'un magicien :

- Le magicien doit avoir au moins un sort.
- Il ne devrait pas avoir de doublons dans sa liste.
- Vous pouvez sauter le choix de l'arme puisque ce dernier ne peut pas en posséder.

** Vous devez faire au minimum quatre combat. Un tableau avec des idées vous est fourni plus bas **

Le **`Program`** est le seul endroit dans le code où vous pouvez faire de l'affichage et de la saisie à la console.

Je vous invite à vous poser la question suivante : puisque la classe **`Program`** et **`GestionJeu`** sont assez intimement liées, qui va gérer quoi? Posez-vous la question selon l'angle de la responsabilité unique : si Program est la seule classe où vous pouvez faire de l'affichage et de la gestion d'entrées, est-ce qu'elle devrait être responsable d'autres choses? Prenez le temps d'y réfléchir et n'hésitez pas à créer de nouvelles classes si vous le jugez nécessaire.

Vous devriez créer plusieurs fonctions statiques ici afin de bien découper votre code. Les méthodes de saisie suivantes devraient être un minimum et retourneront seulement une valeur valide :

- `public static int DemanderNombreEntreMinEtMax(int min, int max)`
- `public static string DemanderChaineNonVide(string question)`
- `public static int ChoisirOption(string[] options, bool avecZero)`[1]

[1] Des détails seront fournis en classe par le professeur

3.2 Exemples :

Exemple – Création du personnage :

```
Nom du personnage:
Gandalf
Quelle classe choisissez-vous pour votre pesonnage?
1- Archer
2- Mage
3- Guerrier
4- Assassin
5- Moine
Entrez une valeur entre 1 et 5 :
2
Quelle arme voulez-vous utiliser?
1- Aucune (mains nues)
2- Épée et bouclier
3- Épée à 2 mains
4- Arc
Entrez une valeur entre 1 et 4 :
1
Quel sort voulez-vous ajouter?
0 - Quitter
1- Boule de feu
2- Missile magique
3- Foudre
Entrez une valeur entre 0 et 3 :
1
Quel sort voulez-vous ajouter?
0 - Quitter
1- Boule de feu
2- Missile magique
3- Foudre
Entrez une valeur entre 0 et 3 :
3
Quel sort voulez-vous ajouter?
0 - Quitter
1- Boule de feu
2- Missile magique
3- Foudre
Entrez une valeur entre 0 et 3 :
0
*****************************************
********** Personnage créé ***************
*****************************************
Nom: Gandalf
Arme: Aucune(mains nues)
Classe: Mage
        Pts Vie: 3      PtsArmure: 7    Pts Force: 3    Pts Exp: 0
Sorts:  Boule de feu     Dégats = 1-6
        Foudre           Dégats = 0-0
```

Exemple - Gestion des erreurs de saisie :

```
Nom du personnage:

Ce champ est obligatoire.
Nom du personnage:
Boromir
Quelle classe choisissez-vous pour votre pesonnage?
1- Archer
2- Mage
3- Guerrier
4- Assassin
5- Moine
Entrez une valeur entre 1 et 5 :
7
Le chiffre doit être entre 1 et 5.
Entrez une valeur entre 1 et 5 :
e
Vous devez entrer un chiffre
Entrez une valeur entre 1 et 5 :
3
Quelle arme voulez-vous utiliser?
1- Aucune (mains nues)
2- Épée et bouclier
3- Épée à 2 mains
4- Arc
Entrez une valeur entre 1 et 4 :
2
****************************************
*********** Personnage créé **************
****************************************
Nom: Boromir
Arme: Épée et bouclier
Classe: Guerrier
       Pts Vie: 1      PtsArmure: 2    Pts Force: 2    Pts Exp: 0
```

3.3 SECTION SUPPLÉMENTAIRE - Précisions sur le jeu – Niveaux du jeu

Pour réussir le jeu, le joueur doit réussir 4 niveaux. À chaque niveau, le joueur devra combattre un adversaire différent. Voici une liste des niveaux et leurs adversaires (vous pouvez également créer votre propre aventure) – NOTEZ QUE LES ENNEMIS SONT AUSSI DES PERSONNAGES :

No	Description	Adversaire
1	Le premier donjon dans lequel vous entrez est rempli de squelettes et de restes humains…	Nom : Squelettes Classe : Squelettes Points de vie : 15 Points de force : 3 Points d'armure : 3 Arme : Aucune
2	Après la lutte contre les hordes de morts-vivants, la lumière se fait de plus en plus rare… La suite du donjon contient toutes sortes de créatures inquiétantes…	Nom : Goblin Classe : Goblin Points de vie : 8 Points de force : 2 Points d'armure : 2 Arme : Épée et bouclier

3	Les gobelins et les morts-vivants n'étaient pas de taille! Tout à coup, les gigantesques portes de la dernière salles'ouvrent pour révéler un énorme lézard ailé provenant de l'enfer : Un dragon! Le défi ultime.	Nom : Faermoore Classe : Dragon Points de vie : 25 Points de force : 5 Points d'armure : 5 Arme : Aucune Sort : Boule de feu
4	Avec Faermoore tué, les murs du donjon s'effondrent… Un dernier survivant de la destruction du donjon se lance à votre poursuite…	Nom : Troll Classe : Troll Points de vie : 20 Points de force : 6 Points d'armure : 4 Arme : Épée à deux mains

Selon le tableau où le joueur est rendu :

- Afficher la description du tableau
- Afficher le nom de l'adversaire à combattre
- Engager le combat
- Afficher chaque tour du combat

Après chaque combat gagné :

- Les différentiels de dégâts pour chaque tour sont affichés.
- Les points d'expérience du personnage sont augmentés de 50 points.
- À tous les 100 points d'expérience accumulés, l'attaque du personnage est augmentée de 1 point.
- Le personnage peut recueillir une récompense.

4 Tests unitaires

Vous devez produire les tests unitaires pour :

- Personnage
  - Création du personnage
  - Gestion de la vie du personnage (respect des contraintes de la vie)
  - Gestion de l'expérience du personnage (augmentation de la force est respectée)
- Sort
  - Création du sort

5 Évaluation

Vous trouverez dans un fichier sur LÉA (donné ultérieurement) la grille d'évaluation utilisée pour la correction de ce travail pratique.

6 Modalités de remise

Remettez votre projet sur LÉA, dans la section Travaux, à l'intérieur d'une archive *Zip*. Supprimez **tous les fichiers et dossiers temporaires**. N'oubliez pas de supprimer le dossier TestsResults. Faites une seule remise par équipe.