Cégep de Saint-Hyacinthe
Département d'informatique

Programmation orientée objet (420-2DP-HY)

TP4 – Combats dans l'arène

Préparé par
Martin Lalancette
bureau B-2335

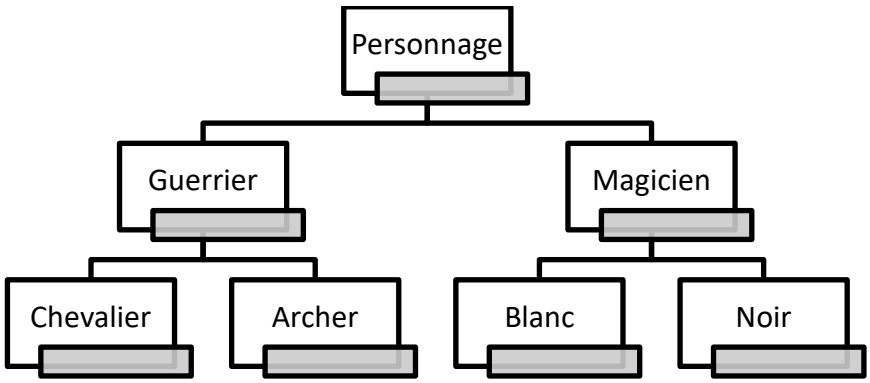
DESCRIPTION

But :	Conception d'un programme en C# de type formulaire comportant l'utilisation des notions vues en classe.
Objectifs :	• Générer un diagramme de classe • Pratiquer les notions antérieures (Révision) • **Bibliothèque** • **Notions d'héritage et polymorphisme** • **LINQ** • **Générer la documentation** • **Respecter les normes de programmation**
Durée :	12 h
Pondération :	15 pts (**en équipe de deux obligatoire**)
Remise :	À la fin de la semaine 15.
Contenu général :	• Remettre vos documents d'analyse et votre solution contenant le projet dans un document .ZIP via LÉA.
Notes	• Conserver une copie de sécurité. Il est de **votre responsabilité** de conserver une copie de sécurité dans l'éventualité où la lecture des données serait impossible. Cette copie doit **être disponible sur demande**.

Combats dans l'arène **– 100 %**

Vous devez former une équipe de deux personnes et en informer votre enseignant via MIO. Dans LÉA, il y aura un fichier EXCEL contenant le lien DevOPS – GIT de votre équipe que vous devrez utiliser pour effectuer les transactions GIT. Dans ce travail, vous devez créer une

bibliothèque de classe, un **projet WPF** et gérer la documentation. Le but de cette application est de simuler des combats médiévaux entre plusieurs personnages (8 minimum). Voici tout d'abord la hiérarchie de classe (simplifiée) des personnages qui peuvent se retrouver dans l'arène :



Dans un premier temps, vous devez coder ces classes dans une **bibliothèque de classes**. Seules les dernières classes sont **concrètes**, les autres sont **abstraites**. Les caractéristiques suivantes doivent être bien situées dans ces classes :

- Nom, Niveau, Expérience, Classe d'armure, Points de vie, Bonus d'attaque, Dommage maximal, Sorts et potions (magiciens seulement), Nom de son démon (magicien noir seulement), Nombre de flèches (???),
- Victoires, Défaites, Nombre d'attaques
- Attaquer(), autres méthodes que vous jugez nécessaires.
- D'autres constantes, champs, propriétés ou méthodes que vous jugez pertinents.

Créez une classe **Arène**, **dans la bibliothèque de classes**, qui contient le code nécessaire aux fonctions suivantes :

- Création d'un groupe de 8 personnages différents (ou plus si vous désirez, en nombre pair), dont le type (Chevalier, Archer, Magicien (blanc ou noir)) est déterminé aléatoirement. Au début, tous les personnages ont le niveau à 1 et l'expérience à 0.
- Les 2 premiers personnages se combattent : le 1er attaque et le 2e se défend, puis c'est l'inverse, jusqu'au premier personnage mis KO (Points de vie <= 0). Ensuite, les 2 personnages suivants combattent. **Un personnage obtient de l'expérience lors d'une victoire seulement : Niveau de l'adversaire X 50 pts**. 0 point pour une défaite.
- Les combats font rage jusqu'au moment où, à la fin d'un tour, un des personnages a atteint le niveau 10. À chaque tour, l'ordre des personnages doit être changé au hasard et les points de vie réinitialisés à leur valeur de départ.
- Une attaque : Lancer un d20, si le chiffre obtenu + bonus d'attaque est supérieur à la classe d'armure de l'adversaire, celui-ci est touché sinon manqué. Les dommages sont calculés en fonction du **dé de dommage** de l'attaquant. Déduire le chiffre obtenu des points de vie de l'adversaire.
- Augmentation du niveau (tous les personnages) :

Expérience	Niveau	Expérience	Niveau
0	1	14 000	6
300	2	23 000	7
900	3	34 000	8
2 700	4	48 000	9
6 500	5	64 000	10

Tableau des valeurs de base par classe :

Caractéristique	Chevalier	Archer	Magicien N	Magicien B
Niveau partant	1	1	1	1
Expérience partante	0	0	0	0
Classe d'armure	11	9	8	10
Points de vie de départ	20 (Augmente de 10% par niveau)	17 (Augmente de 13% par niveau)	9 (Augmente de 12% par niveau)	11 (Augmente de 10% par niveau)
Dommages maximums	1d8	1d8	1d10	2d4

Tableau des bonus à l'attaque à additionner au chiffre du d20**:

Attaquant / Défenseur		Magiciens		Guerriers	
		Blanc	Noir	Chevalier	Archer
Magiciens	Blanc	0	+2	+2	+2
	Noir	+1	0	+2	+2
Guerriers	Chevalier	+1	+2	0	+3
	Archer	+1	+2	+2	0

Interface(s) visuelle(s) dans le projet WPF:

- **Concernant l'affichage dans l'interface visuelle, vous devez utiliser votre créativité. Vous devez utiliser des minuteries (*timer* ~~*ou thread*~~) pour gérer les temps et les affichages.**
- Le formulaire principal doit contenir minimalement :
  - Une liste (ListView) contenant les 8 (minimum) combattants. **Trier les joueurs par niveau et par points en ordre descendant par défaut**.
  - Afficher les statistiques au fur et à mesure des combats.
  - Deux modes d'exécution possibles (un bouton **Combattre** et une case à cocher nommée **Tout**)

- Si **Tout n'est pas coché** et que je clique sur **Combattre**, les statistiques sont affichées après le tour (1 tour = que les 8 combattants se sont battus jusqu'à une défaite ou une victoire). Si je clique à nouveau, les combats du deuxième tour ont lieu et les statistiques sont rafraichies au terme de ce tour. Ainsi de suite jusqu'à ce qu'une personne atteigne le niveau 10.
- Si Tout **est coché** et que je clique sur **Combattre**, tous les tours sont exécutés jusqu'à ce qu'un personnage atteigne le niveau 10. Par la suite, les statistiques sont affichées.
- Ajouter un bouton **Sauvegarder** qui permet d'enregistrer les statistiques dans fichier JSON.
- Ajouter un bouton **Charger** qui permet de récupérer les statistiques du fichier JSON.
- Ajouter un bouton **Réinitialiser** qui permet de remettre les statistiques de combat à zéro afin de pouvoir recommencer les combats du début en tout temps.

Générer la documentation automatisée avec DefaultDocumentation:

- **Seulement la bibliothèque de classes sera documentée**.
- Bien documenter les membres de la bibliothèque de classes avec les commentaires XML vus en classe.
- Ajouter la composante (présentée en classe) qui permet d'extraire les informations XML pour produire la documentation sous forme **Markdown** automatique.
- Publier le tout dans DevOps GIT de ce projet.
- **Concevoir le page README.MD de la page d'accueil du projet**.

Barème d'évaluation
Respect des normes de programmation et opérations habituelles dans DevOps (GIT) ..... /1.0
Modélisation des classes de personnages selon ce qui est demandé + diagramme de classes . /2.0
Programmation des méthodes Attaquer et autres méthodes + bonus d'attaque ..... /2.0
Programmation de la classe Arène et ses fonctionnalités ..... /2.5
Programmation de l'interface visuelle globale (Créativité !!!)..... /2.5
Programmation de l'interface – Bouton Combattre et options..... /2.0
Programmation de l'interface – Boutons Sauvegarder, Charger JSON et Réinitialiser ..... /1.5
Génération de la documentation (Wiki – projet et code) ..... /1.5
Note totale* ..... /15.0

* Tout travail plagié en partie ou en totalité se verra attribuer une note totale de 0 %.

PDEA #1: Lors d'activités d'évaluation sommative en classe ou hors classe (documentation, rapport de laboratoire, rapport de stage, examen), une **pénalité maximale de 10 %** peut être retranchée de la note finale de ladite évaluation (le barème étant de **0,5%/erreur incluant les fautes répétitives**). Pour les commentaires dans les

programmes informatiques, le barème est de **0,5% par faute**. Pour les interfaces utilisateur, le barème est de **1% par faute**.

PDEA #4 : Toute évaluation sommative remise après la date d'échéance fixée se voit attribuer la note zéro pour les étudiants **de 2e à 6e session**. Afin de faciliter l'accueil et l'intégration des nouveaux étudiants de **1re session**, cette règle s'appliquera de la façon suivante : 30 % de pénalité pour une 1re offense (à condition que la remise soit faite dans les 24 heures suivant la date de remise officielle), 100% de pénalité pour les offenses subséquentes.