Section pratique

Allez chercher le projet GoblinQuest sur Lea ou à https://github.com/felixsoum/GoblinQuest
et modifiez le projet pour répondre aux questions.

Question 6. [10 points]

a) Dans la classe **`Goblin`**, implémentez la méthode `IsAlive()` pour qu'elle retourne `true` si les points de vies (`hp`) du goblin sont supérieur à `0`.

b) Créez une nouvelle classe **`GoblinTank`** qui dérive de **`Goblin`**. Suite à son instanciation, le **`GoblinTank`** doit avoir le double du nombre de points de vie (`hp`) défini dans **`Goblin`**. Ne pas utiliser un nombre littéral (`200`), puisque le nombre peut varier dans la classe **`Goblin`**.

c) Créez une nouvelle classe **GoblinAssassin** qui dérive de **Goblin**. Suite à son instanciation, le **GoblinAssassin** doit avoir le triple des points de dégâts (`damage`) défini dans **Goblin**. Ne pas utiliser un nombre littéral (`30`), puisque le nombre peut varier dans la classe **Goblin**.

Question 7. [10 points]

Ajoutez un projet de test à la solution et créez les tests nécessaires pour confirmer le bon fonctionnement de votre code suite au questions 6a, 6b et 6c. Puisqu'il y a 3 questions, vous devez faire 3 tests indépendants.

Question 8. [10 points]

Implémentez les 3 méthodes qui sont `static` dans la classe **`Weapon`** en utilisant des requêtes LINQ. Il y a des méthodes dans le `Main()` du **`Program`** pour vous aider à tester vos méthodes.

a) Cette méthode doit trier les armes en ordre décroissant de coût (plus cher au moins cher):

```
public static List<Weapon> SortByCostDescending(List<Weapon> weapons)
```

Résultat voulu:

```
Greatsword
Greataxe
Sword
Axe
Dagger
Baton
Croissant
```

b) Cette méthode doit filtrer les armes pour retenir seulement les armes qui sont de la rareté Epic:

```
public static List<Weapon> FilterEpicOnly(List<Weapon> weapons)
```

Résultat voulu:

```
Greatsword
Greataxe
```

c) Cette méthode doit filtrer les armes pour retenir seulement les armes gratuit (coût de 0):

```
public static List<Weapon> FilterFreeOnly(List<Weapon> weapons)
```

Résultat voulu:

```
Baton
Croissant
```

Question 9. [10 points]

Implémentez la méthode `static Battle()` dans la classe **`Program`** et la méthode `Attack()` dans la classe **`Goblin`**. `Battle()` doit simuler le combat entre deux goblins en suivant ces exigences:

- Le goblin qui débute le combat est choisi aléatoirement (utilisez `Random`). Il y a 50% de chance que ce soit le premier et 50% de chance que ce soit le deuxième.
- Tour à tour les combattants s'attaque en invoquant la méthode `Attack()` dans la classe **`Goblin`**. Le combat s'arrête dès qu'un goblin meurt.
- Les goblins infligent des points de dégâts supplémentaire dépendamment des armes équipées:
    - Aucun bonus si aucune arme
    - `Rarity.Common`: +1 damage
    - `Rarity.Rare`: +2 damage
    - `Rarity.Epic`: +3 damage

Question 10. [10 points]

Implémentez le patron DAO dans le jeu pour sauvegarder les données des goblins contenu dans une liste. La sauvegarde se fait dans un fichier texte (plain text). Le DAO doit aussi pouvoir aller chercher ces données pour recréer la liste de tous les goblins sauvegardés. Assurez vous d'attendre ces exigences suivantes:

- Méthode pour sauvegarder les données, avec la signature:
  `public static void InsertAll(List<Goblin> goblins)`
- La sauvegarde doit retenir toutes les valeures actuelles des champs des goblins (un goblin pourrait être endommagé ou mort), ainsi que le type actuel du goblin (soit **`Goblin`**, **`GoblinTanker`** ou **`GoblinAssassin`**) et l'arme équipée par le goblin. Alors toutes les données des armes équipées doivent être sauvegardées aussi.
- Méthode pour charger la sauvegarde avec la signature:
  `public static List<Goblin> GetAll()`
- Cette méthode doit recréer parfaitement l'états de tous les goblins sauvegardés y compris leur armes et leur sous-type s'ils sont soit un **`GoblinTanker`** ou un **`GoblinAssassin`**.
- Le fichier de sauvegarde doit se retrouver dans un endroit relatif et ne doit pas dépendre de la machine.