Programmation orientée objet (C++) – ING2-GSI

TP5 – 2 : Héritage multiple

Objectifs

- Comprendre l'héritage multiple
- Utiliser des collections hétérogènes

Jeu de cartes

Un jeune programmeur veut tenter d'appliquer ses nouvelles connaissances en C++ à la gestion d'un de ses hobbies, un jeu de cartes simulant des combats de magiciens.

Dans ce jeu, il existe trois types de cartes : les « terrains », les « créatures » et les « sortilèges ». Les terrains possèdent une couleur parmi cinq : blanc, bleu, noir, rouge et vert.

Les créatures possèdent un nom, un nombre de points de dégâts et un nombre de points de vie.

Les sortilèges possèdent un nom et une explication (sous forme de texte).

De plus, chaque carte, indépendamment de son type, possède un coût. Le coût d'un terrain est nul.

Attention au piège : `cout` est déjà un mot réservé du langage (lorsqu'on utilise l'espace de noms `std`) !!

Dans un programme `magic.cpp`, proposer (et implémenter) une hiérarchie de classes permettant de représenter des cartes de différents types.

Chaque classe aura un constructeur permettant de spécifier les valeurs de ses attributs. De plus, chaque constructeur devra afficher le type de la carte.

Le programme doit utiliser la conception orientée objets et ne doit pas comporter de duplication de code.

Ajouter ensuite aux cartes une méthode `afficher` qui, pour toute carte, affiche son coût et la valeur de ses arguments spécifiques.

Créer de plus une classe pour représenter un jeu de cartes, c.-à-d. une collection de telles cartes.

Cette classe devra avoir une méthode `ajouter` permettant d'ajouter une carte au jeu (de sorte à pouvoir utiliser le polymorphisme sur les cartes par la suite). Il sera peut-être utile d'ajouter une méthode libérant le contenu du jeu de carte.

Il existe de plus certaines cartes particulières qui peuvent être à la fois un terrain et une créature. Proposer une nouvelle classe pour de telles cartes. La méthode `afficher` devra dans ce cas donner toutes les informations relatives à la carte (i.e. sa couleur et le nombre de points de dégâts et de points de vie).

Pour finir, constituer dans le `main()` un jeu contenant divers types de cartes et faites afficher le jeu grâce à une méthode `afficher` propre à cette classe.

Par exemple, le `main` pourrait ressembler à (nécessite peut-être des adaptations) :

```
Jeu mamain;
mamain.ajoute(new Terrain(BLEU));
mamain.ajoute(new Creature(6, "Golem", 4, 6));
mamain.ajoute(new Sortilege(1, "Croissance Gigantesque",
     "La créature ciblée gagne +3/+3 jusqu'à la fin du tour"));
mamain.ajoute(new CreatureTerrain(2, "Ondine", 1, 1, BLEU));
mamain.afficher();
```

et produirait un résultat ressemblant à :

```
On change de main
Un nouveau terrain.
Une nouvelle créature.
Un sortilège de plus.
Une nouvelle créature.
Un nouveau terrain.
Houla, une créature/terrain.
Là, j'ai en stock :
     + Un terrain bleu.
     + Une créature Golem 4/6 de coût 6
     + Un sortilège Croissance Gigantesque de coût 1
     + Une créature/terrain bleue Ondine 1/1 de coût 2
Je jette ma main.
```

NOTE : Les méthodes `afficher` peuvent bien sûr être remplacées par l'opérateur <<.