INF1010

Programmation Orientée-Objet

Travail pratique #4
Polymorphisme

Objectifs :	Permettre à l'étudiant de se familiariser le concept de polymorphisme.
Remise du travail :	Mardi 1er Novembre 2016, 8h
Références :	Notes de cours sur Moodle & Chapitre 8 et 19du livre Big C++ 2e éd.
Documents à remettre :	La solution ainsi que les fichiers .cpp et .h complétés réunis sous la forme d'une archive au format .zip.
Directives :	Directives de remise des Travaux pratiques sur Moodle Les en-têtes (fichiers, fonctions) et les commentaires sont obligatoires. Les travaux dirigés s'effectuent obligatoirement en équipe de deux personnes faisant partie du même groupe. Veuillez suivre le guide de codage

Informations préalables

Compilation : le cas des inclusions (ou dépendances) circulaires. On appelle inclusion circulaire le fait que deux « points .h » s'incluent mutuellement. L'usage simple des gardes de compilation ne suffit plus à protéger le programmeur d'une erreur de compilation. En effet, les gardes de compilation (***#ifndef***, ***#define*** et ***#endif***) empêchent qu'un même fichier soit inclus plusieurs fois. Mais le problème des dépendances circulaires réside dans le fait que chacun des deux fichiers a besoin des déclarations présentes dans l'autre. En particulier, si on imagine deux classes ayant chacune une méthode prenant l'autre type de classe en argument, on a alors un cas de dépendances circulaires. Pour résoudre ce problème, on utilise des déclarations anticipées. C'est-à-dire que l'on prévient le compilateur de l'existence d'une classe via la déclaration *class maClasse;* on définira la seconde classe après ces directives (dans le même fichier). Et on fera l'opération inverse dans le fichier contenant la déclaration de *maClasse*.

Notez, toutefois, que les inclusions circulaires peuvent ralentir la compilation et doivent être utilisées avec parcimonie.

Enfin, certaines fonctionnalités du TP requièrent de l'aléatoire. Pour cela, veuillez utiliser les fonctions *rand* et *srand*. En particulier, soyez très prudent avec l'utilisation de *srand,* une mauvaise utilisation provoquera un résultat déterministe !

Travail à réaliser

Le travail présent a pour but d'ajouter de nouvelles fonctionnalités au jeu Polyland. Les mécaniques que nous allons ajouter sont inspirées d'un célèbre jeu, lui aussi basé sur des combats entre des créatures.

L'ajout de ces fonctionnalités permet d'introduire une notion fondamentale de la programmation orientée objet : *le polymorphisme*. Une classe représente habituellement un concept ou une généralisation, cependant il se peut que deux classes soient deux déclinaisons d'un même concept. Il est alors pertinent de faire intervenir le polymorphisme plutôt que d'implémenter les mêmes méthodes. Cette notion va de pair avec la notion de d'héritage que vous avez vu au cours des prochaines semaines et au cours du TP3.

Suite à une réunion avec les responsables design et gameplay, il vous est demandé de permettre que certaines attaques modifient l'état de la créature de votre adversaire. Cependant, seuls les créatures magiques ont cette chance. Autrement dit, les créatures ne possèdent pas d'attaque magique, mais les créatures magiques en ont. Pour user de leur attaque magique, elles doivent l'utiliser avant d'effectuer une attaque régulière sur leur adversaire.

ATTENTION : Tout au long du TP, assurez-vous d'utiliser les opérateurs sur les objets et non sur leurs pointeurs ! Vous devez donc déréférencer les pointeurs si nécessaires.

ATTENTION : Il est fortement recommandé d'utiliser les fichiers fournis, plutôt que de continuer avec vos fichiers du TP3.

ATTENTION : Sauf mention explicite du contraire, c'est à vous de déterminer la visibilité de vos attributs (protected, private, public).

ATTENTION : L'un des principes du polymorphisme étant de limiter la duplication du code, pensez à utiliser au maximum les méthodes des classes mères.

ATTENTION : Beaucoup de méthodes dans ce TP sont déjà implémentées pour vous. Or, il est à vous de déterminer quelles méthodes doivent être virtuelles ou non (même parmi ces méthodes déjà implémentées).

Classe *AttaqueMagique*

Cette classe est une **classe abstraite**.

L'attribut suivant doit être créé :

- La durée de l'attaque

Les méthodes suivantes doivent être implémentées/modifiées :

- Un constructeur qui prend en paramètre une durée
- Un destructeur
- Un accesseur et un modificateur de l'attribut
- Une méthode *obtenirTypeAttaque()* qui retourne le type de l'attaque. Pour cette classe, le type de l'attaque est « AttaqueMagique ». **Pensez à utiliser *typeid*.**
- Une méthode virtuelle pure *appliquerAttaque()*
- La méthode *estFini*() qui retourne toujours *true*
- Un opérateur << (voir image à la fin du document)

Classe *AttaqueMagiquePoison*

Cette classe hérite de AttaqueMagique et représente une attaque de poison.

Les méthodes suivantes doivent être implémentées/modifiées :

- Un constructeur par défaut qui initialise la durée à 2
- Un constructeur qui prend en paramètre une durée
- Un destructeur
- Une méthode *obtenirTypeAttaque()* qui retourne le type « AttaqueMagiquePoison ». **Pensez à utiliser *typeid.***
- Une méthode *appliquerAttaque*(). L'attaque magique poison permet, dans le tiers des cas, de réduire l'énergie de la créature adverse de 2 points, dans le cas où celle-ci à au moins 5 points d'énergie (si bien-sûr la durée de l'attaque magique n'est pas nulle). Dans les autres cas, rien ne se passe. **Vous devez seulement indiquer si cette méthode est virtuelle ou non**
- Une méthode *estFini*() qui retourne vraie si la durée est nulle

Classe *AttaqueMagiqueConfusion*

Cette classe hérite de AttaqueMagique et représente une attaque qui rend la créature confuse.

Les méthodes suivantes doivent être implémentées/modifiées :

- Un constructeur par défaut qui initialise la durée à 2
- Un constructeur qui prend en paramètre une durée
- Un destructeur

- Une méthode *obtenirTypeAttaque()* qui retourne le type « AttaqueMagiqueConfusion ». **Pensez à utiliser *typeid***
- Une méthode *appliquerAttaque*(). L'attaque magique poison permet, dans le tiers des cas, de réduire les points de vie de la créature adverse de 2 points, dans le cas où celle-ci à au moins 5 points de vie (si bien-sûr la durée de l'attaque magique n'est pas nulle). Dans les autres cas, rien ne se passe. **Vous devez seulement indiquer si cette méthode est virtuelle ou non**
- Une méthode *estFini*() qui retourne vraie si la durée est nulle

Classe *CreatureMagique*

Cette classe hérite de Creature. Son attribut de type pointeur représente l'attaque que la créature magique peut infliger à son adversaire. Seules les Créatures Magiques peuvent utiliser des Attaques magiques contre leur adversaire (pas les Créatures). **L'*AttaqueMagique* a une relation de type <u>composition</u> avec la *CreatureMagique*.**

<u>Les attributs suivants ont été créés :</u>

- Un pointeur de type *AttaqueMagique* : attaqueMagique_
- L'attribut bonus_ est conservé.

<u>Les méthodes suivantes doivent être implémentées/modifiées :</u>

- Le constructeur par défaut et le constructeur par paramètres qui recois un bonus et une Créature
- Le constructeur de copie, et l'opérateur =. **Pour savoir si l'attaque magique qui doit être copiée est de type *AttaqueMagiquePoison* ou *AttaqueMagiqueConfusion*, pensez à utiliser *typeid*.**Un destructeur
- Les accesseurs et les modificateurs des deux attributs
- Une méthode *obtenirTypeCreature()* qui retourne « CreatureMagique ». **Pensez à utiliser *typeid***
- Une méthode *attaquer()* qui prend en paramètre les mêmes attributs que pour une attaque régulière entre Creatures, soit le pouvoir à utiliser et une créature à attaquer. Cette méthode:
    - Ajoute le bonus qu'elle possède en attribut à ses points de vie si le total du bonus et de ses points de vie est inférieur au nombre de points de vie total.
    - L'effet du pointeur d'AttaqueMagique est appliqué sur son adversaire avant l'attaque, seulement si l'attaque magique n'est pas encore terminée. Finalement, une attaque régulière est faite.
- Les méthodes *apprendreAttaqueMagique* et *oublierAttaqueMagique* qui ajoutent/changent ou effacent l'attribut pointeur d'attaque magique
- L'opérateur << qui affiche non seulement les informations de la créature, mais aussi s'il s'agit d'une Créature ou d'une Créature magique. Son Attaque doit aussi être affichée avec la durée correspondante (voir un exemple du *main* plus bas).

Classe *Creature*

Les méthodes suivantes doivent être implémentées/modifiées :

- Une méthode *obtenirTypeCreature()* qui retourne « Creature ». **Pensez à utiliser *typeid***

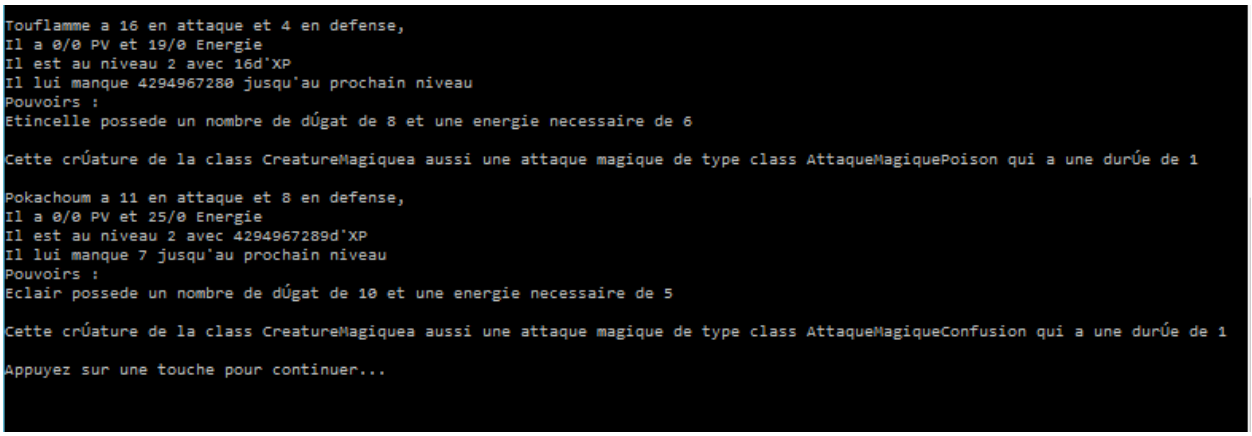
Main.cpp

Le programme principal contient des directives à suivre pour instancier différents objets et essayer les différentes méthodes implémentées.

Le résultat final devrait être similaire à ce qui suit :

```
CREATION DES DRESSEURS

CREATION DES CREATURES

CRfATION DES POUVOIRS

CRfATION DES CREATURES MAGIQUES

CRfATION DES ATTAQUES MAGIQUES

APPRENTISSAGE DES POUVOIRS

APPRENTISSAGE DES ATTAQUES MAGIQUES

AJOUT DE CREATURES ET DE DRESSEURS A POLYLAND

Salimouche a bien ÚtÚ ajoutÚ !
Carapouce a bien ÚtÚ ajoutÚ !
Balbazar a bien ÚtÚ ajoutÚ !
Pokachu a bien ÚtÚ ajoutÚ !
Touflamme a bien ÚtÚ ajoutÚ !
Pokachoum a bien ÚtÚ ajoutÚ !
Regis a bien ÚtÚ ajoutÚ !
Pierre a bien ÚtÚ ajoutÚ !
Sasha a bien ÚtÚ ajoutÚ !
TEST D'AFFICHAGE

Regis possede 1 creature(s) et appartient a l'equipe Equipe de Poly
Pierre possede 1 creature(s) et appartient a l'equipe Equipe de Poly
Sasha possede 1 creature(s) et appartient a l'equipe Team de feu


COMPETITION

TESTS DE COMBAT
Un Salimouche surgit
Vous avez rencontrÚ un Salimouche sauvage qui vous attaque...
Salimouche lance Boule de feu qui inflige 5 degat a Pokachoum
Pokachoum a encore 45 PV
Pokachoum lance Eclair qui inflige 20 degat a Salimouche
Salimouche a encore 23 PV
Pokachoum lance Eclair qui inflige 20 degat a Salimouche
Salimouche a encore 1 PV
Pokachoum lance Eclair qui inflige 20 degat a Salimouche
Pokachoum a gagnÚ -7 XP
Salimouche a encore 0 PV
Vous avez battu un Salimouche, vous pouvez maintenante le capturer
Felicitation vous avez attrapÚ un Salimouche !

Vous trouvez une potion magique, vous dÚcidez de l'utilisez sur Pokachoum
l'objet Potion magiquefournit un bonus de 15
```

```
Touflamme se jette sur votre Pokachoum
Un duel entre Pokachoum et Touflamme est engagÚ
Pokachoum lance Eclair qui inflige 20 degat a Touflamme
Touflamme a encore 25 PV
Touflamme lance Etincelle qui inflige 8 degat a Pokachoum
Touflamme a gagnÚ 16 XP
Pokachoum a encore 0 PV
Salimouche lance Boule de feu qui inflige 10 degat a Touflamme
Salimouche a gagnÚ 72 XP
Touflamme a encore 0 PV
Votre Pokachoum a ÚtÚ battu mais heuresement votre Salimouche finit par vaincre Touflamme
Pokachoum et Salimouche n'arrete pas de se chamailler, vous decidez d'abandonner Pokachum
Vous avez bien relachÚ Pokachoum !

DERNIER TEST AFFICHAGE
Salimouche a 12 en attaque et 3 en defense,
Il a 0/45 PV et 10/20 Energie
Il est au niveau 1 avec 72d'XP
Il lui manque 28 jusqu'au prochain niveau
Pouvoirs :
Boule de feu possede un nombre de dÚgat de 5 et une energie necessaire de 5


Carapouce a 10 en attaque et 1 en defense,
Il a 55/55 PV et 25/25 Energie
Il est au niveau 1 avec 0d'XP
Il lui manque 100 jusqu'au prochain niveau
Pouvoirs :
Pistolet a eau possede un nombre de dÚgat de 6 et une energie necessaire de 6


Salimouche a 12 en attaque et 3 en defense,
Il a 0/45 PV et 10/20 Energie
Il est au niveau 1 avec 72d'XP
Il lui manque 28 jusqu'au prochain niveau
Pouvoirs :
Boule de feu possede un nombre de dÚgat de 5 et une energie necessaire de 5


Balbazar a 11 en attaque et 2 en defense,
Il a 50/50 PV et 22/22 Energie
Il est au niveau 1 avec 0d'XP
Il lui manque 100 jusqu'au prochain niveau
Pouvoirs :
Lance feuille possede un nombre de dÚgat de 5 et une energie necessaire de 5


Pokachu a 10 en attaque et 2 en defense,
Il a 50/50 PV et 25/25 Energie
Il est au niveau 1 avec 0d'XP
Il lui manque 100 jusqu'au prochain niveau
Pouvoirs :
Eclair possede un nombre de dÚgat de 10 et une energie necessaire de 5
Tonerre possede un nombre de dÚgat de 3 et une energie necessaire de 5
```



Questions

1- Pourquoi la classe AttaqueMagique est-elle une classe abstraite ?
2- Dans la classe Créature, quelle(s) méthode(s) avez-vous déclarées virtuelles et pourquoi ?
3- Pourquoi est-il important que les destructeurs de Créature et de CréatureMagique soient virtuels ?

Correction

La correction du TP4 se fera sur 20 points.

Voici les détails de la correction :

- (03 points) Compilation du programme;
- (03 points) Exécution du programme;
- (04 points) Comportement exact des méthodes du programme;
- (04 points) Utilisation adéquate du polymorphisme;
- (2,5 points) Gestion correcte de la mémoire;
- (01 point) Documentation du code;
- (01 point) Utilisation correcte du mot-clé *this* et const;
- (1,5 points) Réponses aux questions.