INF1010

Programmation Orientée-Objet

Travail pratique #4

Polymorphisme et héritage multiple

Objectifs :	Permettre à l'étudiant de se familiariser avec les concepts d'héritage multiple et de polymorphisme
Remise du travail :	Lundi 16 mars 2015, 8h
Références :	Notes de cours sur Moodle & Chapitre 8 et 19 du livre Big C++ 2e éd.
Documents à remettre :	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

Travail à réaliser

Vous allez réaliser un programme qui permet d'afficher plusieurs *Afficheur* à la console, dont un afficheur de temps et de date ainsi qu'un afficheur de météo.

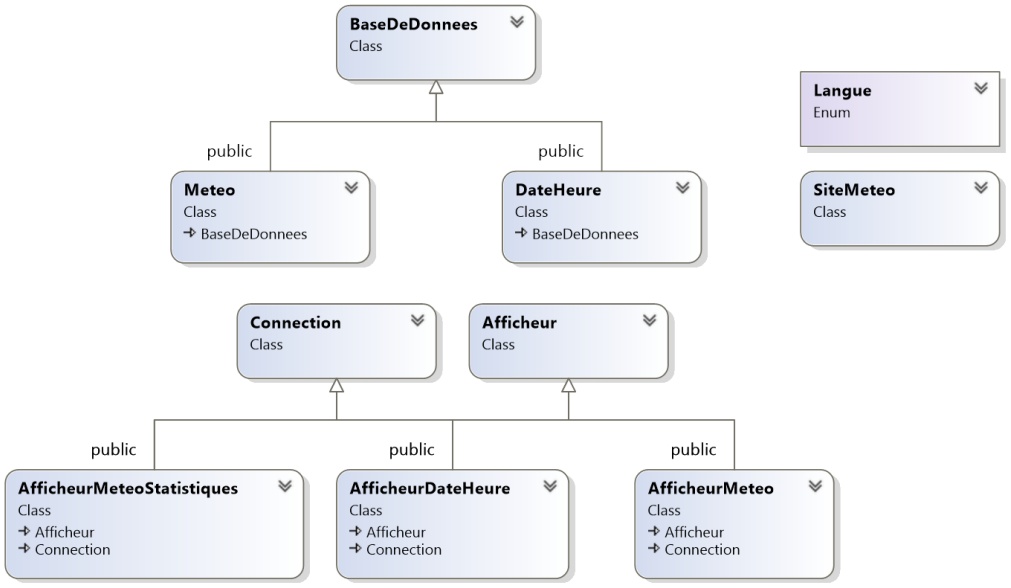

Figure 1 : Hiérarchies de classes

Classe Afficheur

La classe *Afficheur* est la classe de base abstraite des différents types d'afficheurs. Elle contient les méthodes publiques suivantes :

- **afficherFrancais et afficherAnglais** : méthodes virtuelles pures devant être implémentées par les classes dérivées
- **getTypeAfficheur :** méthode virtuelle qui retourne un *string* qui représente le type de l'afficheur. Pour la classe Afficheur, elle retourne le nom de la classe (typeid et la fonction membre name()
- un destructeur au besoin (virtuel ?)

Classe BaseDeDonnees

Classe de base abstraite de toutes les bases de données. D'autres objets peuvent se connecter à une base de données et être mis à jour lorsque les données de la base de données sont modifiées.

L'attribut *protected* suivant doit être présent :

- **connections_:** vecteur de pointeurs d'objets *Connection (vector)*. La relation entre la classe *BaseDeDonnees* et *Connection* est une relation d'agrégation

Les méthodes publiques suivantes doivent être présentes :

- **ajouterConnection**: prend en paramètre un pointeur de *Connection*. Elle retourne *true* lorsqu'une *Connection* est ajoutée avec succès. Elle retourne *false* lorsque la *Connection* est déjà présente.

- **retirerConnection** : prend en paramètre un pointeur de *Connection*. Elle retourne *true* lorsqu'une *Connection* est retirée avec succès. Elle retourne *false* lorsque la *Connection* n'est pas présente.
- **mettreAJourConnections** : met a jour toutes les *Connection* a l'aide de la méthode *Connection::mettreAJourConnection.* C'est à dire, pour les éléments contenus dans l'attribut **connections_,** on fait appel à la méthode **mettreAJourConnection** de la classe Connection.
- **mettreAJourDonnees :** méthode virtuelle pure devant être implémentée par les classes dérivées. Elle met à jour les données de la base de données.

Classe Connection

Il s'agit d'une interface qui permet à ses classes dérivées de se connecter à une base de données pour une mise à jour de celle-ci. Elle contient la méthode publique suivante :

- **mettreAJourConnection :** méthode virtuelle pure. Elle devra être implémentée par les classes dérivées.

Classe Meteo

Il s'agit d'une base de données contenant des données météorologiques, soient la température, le vent et la visibilité. Elle hérite de la classe *BaseDeDonnees*.

Les attributs privés suivants doivent être présents :

- **temperature_, vent_ et visibilite_:** trois *float* qui représentent respectivement la température en degrés Celsius, la vitesse du vent en km/h et la visibilité en km

Les méthodes publiques suivantes doivent être présentes :

- Un constructeur par défaut qui met les trois attributs à zéro
- Un constructeur par paramètres qui prend trois *float* en paramètres, soit la température, le vent et la visibilité.
- Des méthodes d'accès pour les trois attributs.
- **mettreAJourDonnees** : implémentation de la méthode virtuelle pure de la classe de base. Elle met à jour la température, la vitesse du vent et la visibilité selon la formule suivante :
    - *nouvelleValeur* = *ancienneValeur* ± *X*
    - *X* est choisi aléatoirement dans l'intervalle $[0, ancienneValeur * 20\%]$
    - Exemple :
        - Si *ancienneValeur* = 10, *nouvelleValeur* doit être choisis aléatoirement dans l'intervalle [8, 12]

  Elle met aussi à jour toutes ses connexions à l'aide de la méthode *mettreAJourConnections.*

Classe DateHeure

Il s'agit d'une base de données contenant la date et l'heure courante. Elle hérite de la classe *BaseDeDonnees*.

Les attributs privés suivants doivent être présents :

- **heure_, minute_, seconde_, jourMois_, jourSemaine_, mois_ et annee_:** sept entiers non signés qui représentent respectivement l'heure (0 à 24), les minutes (0 à 60), les secondes (0 à 60), la journée du mois (1 à 31), la journée de la semaine (0 à 6), le mois (0 à 11) et l'année.
- **http://www.cplusplus.com/reference/ctime/localtime/ et**
- **http://www.cplusplus.com/search.do?q=struct+tm**

Les méthodes publiques suivantes doivent être présentes :

- Un constructeur par défaut qui met tous les attributs à zéro.
- Des méthodes d'accès pour tous les attributs.
- **mettreAJourDonnees** : implémentation de la méthode virtuelle pure de la classe de base. Elle met à jour l'heure courante.

Classe AfficheurMeteo

Classe qui permet d'afficher la météo, soient la température la vitesse du vent et la visibilité. Elle hérite de la classe *Afficheur*. Elle hérite aussi de la classe *Connection* pour pouvoir se connecter à la base de données *Meteo* et être à jour avec les dernières données météorologiques.

L'attribut privé suivant doit être présent :

- **donnees_:** un objet *Meteo* représentant les données météorologiques les plus récentes.

Les méthodes publiques suivantes doivent être présentes :

- **afficherFrancais et afficherAnglais :** implémentation des méthodes virtuelles pures de la classe *Afficheur*. Elles affichent la température, la vitesse du vent ainsi que la visibilité. La première fait l'affichage en français tandis que la deuxième en anglais. Référez-vous à la capture d'écran plus bas pour le format d'affichage.
- **getTypeAfficheur :** méthode virtuelle qui retourne un *string* qui représente le type de l'afficheur. Pour la classe *AfficheurMeteo*, elle retourne le nom de la classe.
- **mettreAJourConnection** : implémentation de la méthode virtuelle pure de la classe *Connection*. Cette méthode est appelée par la classe *Meteo* lorsque les données météorologiques doivent être mises à jour. Cette méthode met donc simplement l'attribut *donnees_* à jour selon la valeur du paramètre de la méthode.

Classe AfficheurMeteoStatistiques

Classe qui permet d'afficher des statistiques sur la météo, soient la moyenne annuelle de la température, de la vitesse du vent et de la visibilité. Elle hérite de la classe Afficheur et de la classe *Connection* pour pouvoir se connecter à la base de données *Meteo* et être à jour avec les dernières données météorologiques.

Les attributs privés suivants doivent être présents :

- **donneesDerniereAnnee_:** un tableau dynamique (pas un vecteur) pouvant contenir 365 objets de classe *Meteo*. Ce tableau contient toutes les données météorologiques des derniers 365 jours. La relation entre la classe *AfficheurMeteoStatistique* et la classe *Meteo* est une relation de composition
- **nbDonnees_ :** un entier non signé représentant le nombre d'éléments dans le tableau.

Les méthodes publiques suivantes doivent être présentes :

- **afficherFrancais et afficherAnglais :** implémentation des méthodes virtuelles pures de la classe *Afficheur*. Elles affichent la moyenne annuelle de la température, de la vitesse du vent ainsi que de la visibilité. La première fait l'affichage en français tandis que la deuxième en anglais. Référez-vous à la capture d'écran plus bas pour le format d'affichage
- **getTypeAfficheur :** méthode virtuelle qui retourne un *string* qui représente le type de l'afficheur. Pour la classe *AfficheurMeteoStatistique*, elle retourne le nom de la classe.
- **mettreAJourConnection** : implémentation de la méthode virtuelle pure de la classe *Connection*: cette méthode est appelée par la classe *Meteo* lorsque les données météorologiques sont mises à jour. Elle ajoute le contenu pointé du paramètre de la méthode de type BaseDeDonnees* (dernière donnée météorologique) à la fin du tableau et retire la donnée la plus ancienne (au début du tableau) s'il ne reste plus d'espace. L'ordre des données doit être maintenu lorsqu'un élément est retiré.

Classe AfficheurDateHeure

Classe qui permet d'afficher la date et l'heure. Elle hérite de la classe *Afficheur* et *Connection*.

L'attribut privé suivant doit être présent :

- **dateHeure_** : un objet de type *DateHeure*

Les méthodes publiques suivantes doivent être présentes :

- **afficherFrancais et afficherAnglais :** implémentation des méthodes virtuelles pures de la classe *Afficheur*. Elle affiche la date et l'heure. La première fait l'affichage en français tandis que la deuxième en anglais. Référez-vous à la capture d'écran plus bas pour le format d'affichage
- **getTypeAfficheur :** méthode virtuelle qui retourne un *string* qui représente le type de l'afficheur. Pour la classe *AfficheurDateHeure*, elle retourne le nom de la classe.

- **mettreAJourConnection** : implémentation de la méthode virtuelle pure de la classe *Connection*. Cette méthode met l'attribut *dateHeure_* à jour selon la valeur du paramètre de la méthode.

Classe SiteMeteo

Représente le site internet d'une chaine météo.

Les attributs *private* suivants doivent être présent :

- **afficheurs_:** vecteur contenant des pointeurs d'afficheurs. La relation entre la classe *SiteMeteo* et *Afficheur* est une relation d'agrégation.
- **langue_ :** représente la langue utilisée pour l'affichage

Les méthodes publiques suivantes doivent être présentes :

- **afficherSite :** elle affiche tous les afficheurs qu'elle contient. Le type de l'afficheur (methode *getTypeAfficheur*) est affiché au-dessus de chaque afficheur.
- **ajouterAfficheur** : prends un pointeur d'afficheur en paramètre. Permet simplement d'ajouter un afficheur au site. Un même afficheur peut être ajouté plusieurs fois.
- **retirerAfficheur** : prends un pointeur d'afficheur en paramètre. Permet de retirer un afficheur du composite. L'ordre des éléments du tableau doit être maintenu
- une fonction d'accès et de modification de l'attribut langue.

Main.cpp

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

Voici le format d'affichage attendu :

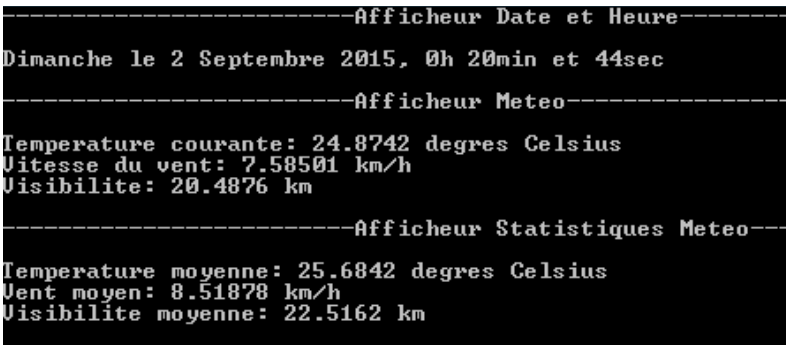

Figure 2 : Affichage en français

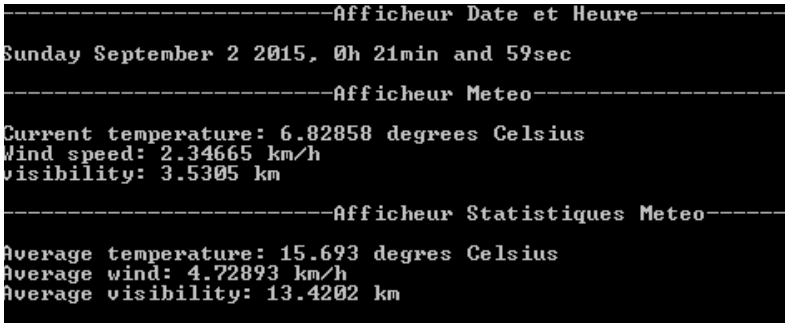

Figure 3 : Affichage en Anglais

Correction

La correction du TP4 se fera sur 20 points. Voici les détails de la correction:

- (10 points) Compilation et exécution exacte des différentes méthodes ;
- (04 points) Utilisation adéquate du polymorphisme ;
- (03 points) Gestion correction de la mémoire ;
- (02 point) Documentation du code ;
- (01 point) Utilisation correcte du mot-clé *const* ;