INF1010
Programmation Orientée-Objet

Travail pratique #3
Héritage

Objectifs :	Permettre à l'étudiant de se familiariser avec l'héritage, une des notions de base de la programmation orientée-objet.
Remise du travail :	Lundi 18 février 2013, 10h
Références :	Notes de cours sur Moodle & Chapitre 8 du livre Big C++ 2e édition
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 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

---

Un comité de Polytechnique souhaite organiser un Vins & Fromages au sein de l' É cole, et vous demande de les aider à concevoir un système permettant d'enregistrer les individus qui seront présents, tels que les représentants d'entreprise, les professeurs et les étudiants.

Travail à réaliser :

Dans ce travail, vous aurez à concevoir les classes *Individu*, modélisant un individu quelconque, *Entreprise*, modélisant un représentant d'entreprise, *Professeur*, modélisant un professeur, *Etudiant*, modélisant un étudiant, et *VinsEtFromages*, permettant d'enregistrer les informations sur les individus présents.

Les classes *Entreprise*, *Professeur* et *Etudiant*, dérivent publiquement de la classe *Individu*..

Pour la classe *VinsEtFromages*, vous devez utiliser trois vecteurspour stocker les objets des classes *Entreprise*, *Professeur* et *Etudiant*.

Vous devez compléter les fichiers *VinsEtFromages.h* et *Individu.h*, *Etudiant.h*, *Professeur.h*, et *Entreprise.h* ainsi que leurs fichiers sources (.cpp) en définissant les attributs et les méthodes

demandées dans le présent sujet. Vous devrez finalement écrire un programme principal *main.cpp* permettant d'utiliser les classes ainsi définies.

Classe *Individu*

- Un constructeur par défaut ;
- Un constructeur par paramètres ;
- Un constructeur par copie ;
- Un destructeur ;
- Un attribut privé **nom_** (string) ;
- Un attribut privé **prenom_** (string) ;
- Un attribut privé **listeVinsPreferes_** (tableau dynamique de string dont la taille initiale sera 0 et qui augmentera ou diminuera au fur et à mesure des ajouts/suppressions) ;
- Un attribut protégé **prixEntree** ;
- Les méthodes d'accès **get*()** et de modification **set*()** pour les attributs privés excepté **listeVinsPreferes_** ;
- Les méthodes d'ajout **add*()** et de suppression **del*()** pour l'attribut privé **listeVinsPreferes_**. Pour ces deux méthodes, afficher une erreur si le vin ajouté existe déjà (ajout) ou n'existe pas (suppression) dans la liste.
- Une méthode d'accès **getPrix()** pour l'attribut protégé **prixEntree** ;
- Une surcharge de l'opérateur de comparaison d'égalité « == », considérant que deux individus sont le même s'ils ont en commun leur nom et leur prénom ;
- Une surcharge de l'opérateur d'assignation « = » ;
- Une méthode **getString()** qui retournera le texte :

Individu: [Prénom] [Nom] - Prix: [prix d'entrée] CAD - Vins: [liste des vins séparés par une virgule et un espace]

  Ou :

Individu: [Prénom] [Nom] - Prix: [prix d'entrée] CAD

  Selon qu'il y a des vins préférés définis ou non pour l'objet courant.

Le prix d'entrée **prixEntree** par défaut vaudra **0** CAD.

Classe *Etudiant*

- Un constructeur par défaut ;
- Un constructeur par paramètres ;
- Un destructeur ;
- Un attribut privé **departement_** (string) ;

- Un attribut privé **niveau_** (string) ;
- Les méthodes d'accès **get*()** et de modification **set*()** pour les attributs privés ;
- Une méthode **getString()** qui retournera le texte :

Etudiant: [Prénom] [Nom] ([Département], [Niveau]) - Prix: [prix d'entrée] CAD - Vins: [liste des vins séparés par une virgule et un espace]

  Ou :

Etudiant: [Prénom] [Nom] ([Département], [Niveau]) - Prix: [prix d'entrée] CAD

  Selon qu'il y a des vins préférés définis ou non pour l'objet courant.

Lors de l'initialisation d'un objet étudiant, vous prendrez soin de modifier le prix d'entrée **prixEntree** pour que celui-ci vaille **14.5** CAD.

Classe *Professeur*

- Un constructeur par défaut ;
- Un constructeur par paramètres ;
- Un destructeur ;
- Un attribut privé **departement_** (string) ;
- Un attribut privé **bureau_** (string) ;
- Les méthodes d'accès **get*()** et de modification **set*()** pour les attributs privés ;
- Une méthode **getString()** qui retournera le texte :

Professeur: [Prénom] [Nom] ([Département], [Bureau]) - Prix: [prix d'entrée] CAD - Vins: [liste des vins séparés par une virgule et un espace]

  Ou :

Professeur: [Prénom] [Nom] ([Département], [Bureau]) - Prix: [prix d'entrée] CAD

  Selon qu'il y a des vins préférés définis ou non pour l'objet courant.

Classe *Entreprise*

- Un constructeur par défaut ;
- Un constructeur par paramètres ;
- Un destructeur ;
- Un attribut privé **nomEntreprise_** (string) ;
- Un attribut privé **stagiairesCherches_** (entier) ;
- Un attribut privé **emploisProposes_** (entier) ;
- Les méthodes d'accès **get*()** et de modification **set*()** pour les attributs privés ;
- Une méthode **getString()** qui retournera le texte :

```
Entreprise: [Prénom] [Nom], [NomEntreprise] ([NombreStages] stages, [NombreEmplois] emplois) - Prix: [prix d'entrée] CAD - Vins: [liste des vins séparés par une virgule et un espace]
```

Ou :

```
Entreprise: [Prénom] [Nom], [NomEntreprise] ([NombreStages] stages, [NombreEmplois] emplois) - Prix: [prix d'entrée] CAD
```

Selon qu'il y a des vins préférés définis ou non pour l'objet courant.

Classe *VinsEtFromages*

- Un constructeur par défaut ;
- Un destructeur ;
- Un attribut privé **listeEntreprises**, vecteur contenant des objets **Entreprise** ;
- Un attribut privé **listeProfesseurs**, vecteur contenant des objets **Professeur** ;
- Un attribut privé **listeEtudiants**, vecteur contenant des objets **Etudiant** ;
- Les méthodes **add*()** et **del*()**, ou ***** ira pour *Entreprise*, *Professeur* ou *Etudiant*, pour les trois vecteurs précédents – les méthodes **add*()** n'ajouteront l'*Individu* que s'il n'est pas déjà dans la liste concernée, s'il y est déjà, elles afficheront à l'écran, en envoyant sur la sortie erreur, un message affichant le retour de l'appel **getString()** sur l'objet et en précisant qu'il existe déjà ;
- Une méthode **afficherPresents()**, qui parcourra les vecteurs **listeEntreprises**, **listeProfesseurs** puis **listeEtudiants** pour appeler la méthode **getString()** de chacun des objets et l'afficher à l'écran de la manière suivante :

```
Liste des présents :
# Entreprises présentes :
# - Entreprise: P N, E (2 stages, 6 emplois) - Vin: Français - Prix: 0 CAD
# - Entreprise: P2 N2, E2 (5 stages, 2 emplois) - Vin: Français - Prix: 0 CAD
# Professeurs présents :
# - Professeur: P3 N3 (Génie Info, Bureau C) - Vin: Italien - Prix: 0 CAD
# - Professeur: P4 N4 (Génie Log, Bureau C) - Vin: Australien - Prix: 0 CAD
# Etudiants présents :
# - Etudiant: P5 N5 (Génie Info, 2ème année) - Vin: Beaujolais - Prix: 14.5 CAD
```

- Une méthode **getListeIndividus()**, qui retournera un vecteur d'objets **Individu** créé localement, qui contiendra des copies des objets **Entreprise**, **Professeur** et **Etudiant** contenus dans les autres vecteurs.
- Une méthode **afficherIndividus()** qui ne séparera pas les classes dérivées d'Individu dans son affichage, qui appelera la méthode **getListeIndividus()** pour ensuite parcourir le vecteur retourné en appelant la méthode **getString()** de chacun des objets. L'affichage à l'écran devra être de la forme suivante :

```
Individus présents :
# - …
# - …
# - …
```

Pour chacune des classes, vous pouvez implémenter d'autres fonctions si elles vous semblent nécessaires et pertinentes.

Main.cpp

Le programme principal sera constitué du fichier *main.cpp* dans lequel se trouveront les instructions permettant de réaliser l'entrée de quelques *Individu* dans un objet *VinsEtFromages* et de lui faire afficher son contenu avant que le programme ne se termine.

Question

Joindre à votre archive un fichier Question.txt ou Question.pdf contenant la réponse.

Dans la méthode **afficherIndividus()** de la classe *VinsEtFromages*, les différents objets hérités de la classe *Individu* perdent leurs spécificités, excepté pour le prix d'entrée au Vins & Fromages. Expliquez en quelques mots ce qui se passe dans cette méthode justifiant ce comportement.

Correction :

La correction du TP3 se fera sur 20 points. Voici les détails de la correction :

- (10 points) Compilation et exécution exactes des différentes méthodes ;
- (02 points) Documentation du code ;
- (02 point) Respect des consignes de l'énoncé ;
- (03 points) Implémentation correcte des méthodes d'ajout et de suppression du tableau dynamique **listeVinsPreferes_** prenant en compte l'évolution de l'espace mémoire lui étant alloué, ainsi que du constructeur par copie et de la surcharge d'opérateur « = » ;
- (01 point) Utilisation correcte du mot-clé *const* ;
- (02 points) Réponse à la question.