INF1010
Programmation Orientée-Objet

Travail pratique #4
Polymorphisme

Objectifs :	Permettre à l'étudiant de se familiariser avec le polymorphisme.
Remise du travail :	Mercredi 27 février 2013, 10h
Références :	Notes de cours sur Moodle & Chapitres 8-9 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

---

Dans la continuité du travail que vous avez accompli précédemment, le même comité de Polytechnique vous demande d'améliorer le logiciel d'enregistrement des individus pour les Vins & Fromages en y ajoutant quelques fonctionnalités. Vous décidez en même temps d'en améliorer la conception en utilisant le polymorphisme.

Travail à réaliser :

Dans ce TP, vous aurez à intégrer le concept de polymorphisme dans votre programme. Vous devrez pour cela compléter et modifier les classes du premier TP en y ajoutant et supprimant quelques méthodes. Vous devez obligatoirement vous servir de la solution fournie avec l'énoncé comme point de départ pour ce TP. En plus des modifications demandées dans la suite de l'énoncé, vous devez modifier les méthodes lorsque c'est pertinent (par exemple : destructeurs).

Classe *VinsEtFromages*

- Suppression des listes d'étudiants, de professeurs et d'entreprises et ajout d'un vecteur (attribut privé) contenant des pointeurs vers des objets Individus ;

- Modification des méthodes **add*** et **del*** pour s'adapter au vecteur d'individus. Elles prendront des pointeurs vers des individus en paramètres ;
- Ajout de la méthode **getIndividu()** recevant en paramètre deux variables de type string représentant le nom et le prénom de l'individu recherché et retournant la référence de l'individu ;
- Modification de la méthode **afficherIndividus()** pour s'adapter au nouvel attribut. Elle appellera la méthode **getString()** de chaque individu et l'affichage sera réalisé de la façon suivante :

```
Individus présents ([NombreIndividus]) :
# - …
# - …
# - …
```

- Modification de la méthode **afficherPresents()** pour s'adapter au nouvel attribut. Cette méthode doit toujours afficher les individus en les séparant par catégorie en utilisant la méthode **typeId()** de chaque classe. L'affichage sera réalisé de la façon suivante :

```
Entreprises présentes ([NombreEntreprises]) :
# - …

Etudiants présents ([NombreEtudiants]) :
# - …

Professeurs présents ([NombreProfesseurs]) :
# - …
```

- Ajout de la méthode **nombreIndividusDansCategorie()** qui prendra en paramètre une référence vers une variable de type Individu et qui retournera un entier correspondant au nombre d'individus de la classe passée en paramètre dans le vecteur **listeIndividus**.

Classe *Individu*

- Le prix d'entrée par défaut est maintenant de 5 $.
- Modification de la méthode **getString()** pour le polymorphisme ;
- Ajout de la méthode virtuelle pure **calculerPrixEntree()**.

Classe *Entreprise*

- Modification de la méthode **getString()** pour le polymorphisme ;
- Ajout de la méthode virtuelle **calculerPrixEntree()** retournant le prix d'entrée des entreprises correspondant au quadruple du prix d'entrée par défaut ;

Classe *Professeur*

- Modification de la méthode **getString()** pour le polymorphisme ;
- Ajout de la méthode virtuelle **calculerPrixEntree()** retournant le prix d'entrée des professeurs correspondant au double du prix d'entrée par défaut ;

Classe *Etudiant*

- Modification de la méthode **getString()** pour le polymorphisme ;
- Ajout de la méthode virtuelle **calculerPrixEntree()** retournant le prix d'entrée des étudiants correspondant au prix d'entrée par défaut additionné d'un montant dépendant du niveau de l'étudiant. Ce montant additionnel est repris dans le tableau suivant :

Niveau de l'étudiant	Montant additionnel
1er cycle	5 $
2ème cycle	10 $
3ème cycle	15 $

- Ajout d'un attribut privé **listeEntreprisesPreferees_** de type double pointeur. Cet attribut est un tableau dynamique contenant des pointeurs vers des entreprises ;
- Ajout d'un attribut privé **nbEntreprisesPreferees_** de type entier contenant le nombre d'entreprises préférées ;
- Ajout de la méthode **getEntreprisePreferee()** prenant en paramètre la position de l'entreprise voulue dans le tableau et retournant le pointeur vers cette entreprise ;
- Ajout de la méthode **nbEntreprisesPreferees()** retournant le nombre d'entreprises préférées ;
- Ajout de la méthode **addEntreprisePreferee()** prenant en paramètre un pointeur vers une entreprise. Cette méthode ajoute celui-ci au tableau **listeEntreprisesPreferees_** en gérant efficacement la mémoire.

Main.cpp

Le fichier main.cpp contiendra les instructions permettant de vérifier l'implémentation des différentes méthodes de vos classes. Vous devez implémenter un bloc d'instructions permettant d'ajouter des individus dans le vecteur `vector<Individu*> individus` à partir de la lecture du fichier *individus.txt*.

L'ajout de vins préférés aux individus doit être réalisée en lisant le fichier *vinsPreferes.txt* et en utilisant la méthode **getIndividu()** de la classe *VinsEtFromages*.

Un exemple d'affichage de l'ajout d'individus et de vins est présenté sur la figure suivante :

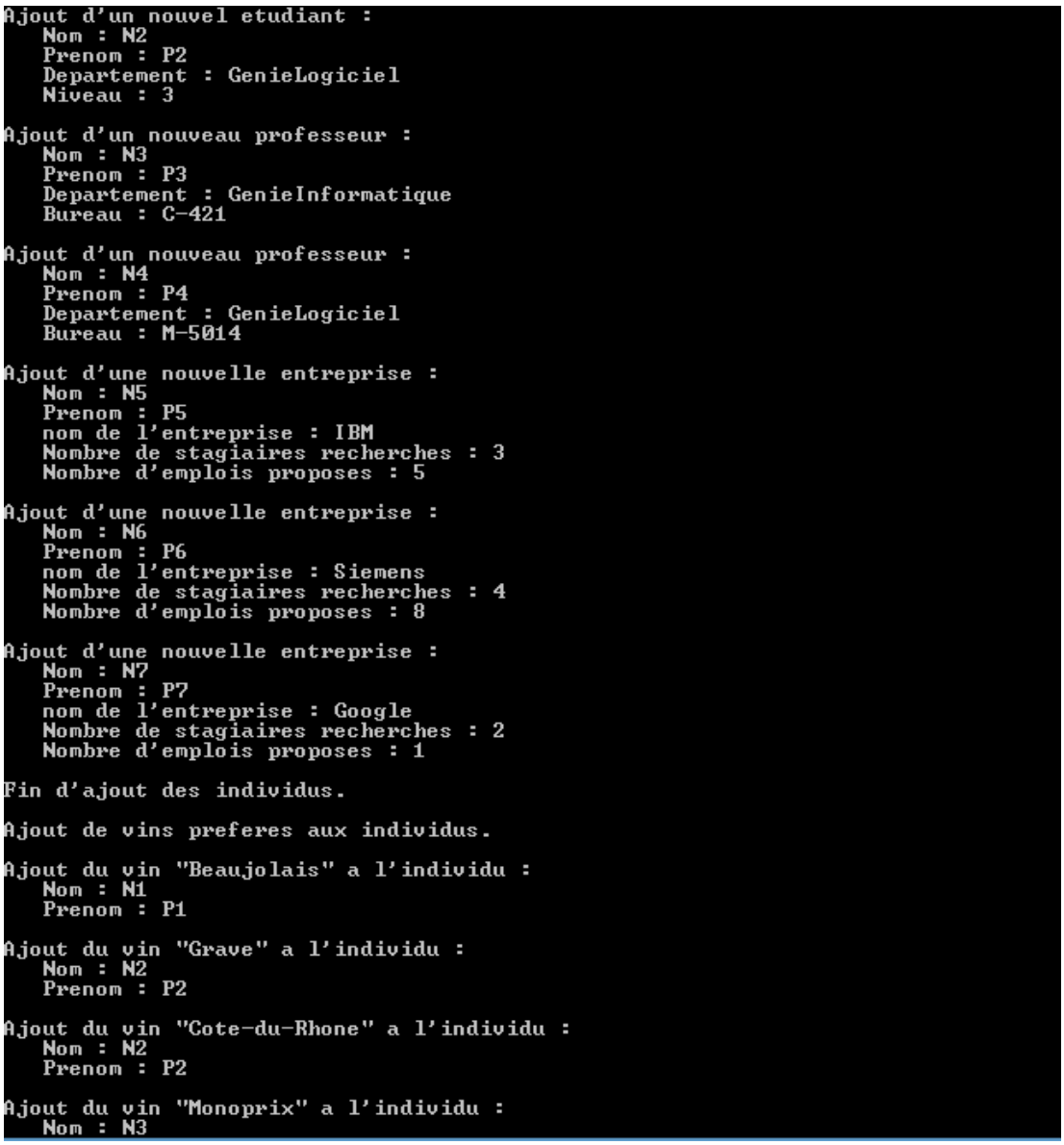

Figure 1 : Exemple d'ajout d'individus par lecture de fichier

Veuillez suivre les autres instructions fournies.

Question

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

Dans votre programme, lors de la suppression des individus, quel(s) destructeur(s) est (sont) appelé(s) ?

Correction :

La correction du TP4 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 points) Respect des consignes de l'énoncé ;
- (04 points) Implémentation correcte du polymorphisme ;
- (02 points) Réponse à la question.