INF1010

Programmation Orientée-Objet

Travail pratique #3

Héritage

Objectifs :	Permettre à l'étudiant de se familiariser avec le concept d'héritage.
Remise du travail :	Lundi 23 Février 2015, 8h
Références :	Notes de cours sur Moodle & Chapitre 8 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

Due à de récents vols, votre université vous demande de créer un système de contrôle d'accès. Ce système s'assure que seules les personnes autorisées puissent entrer dans un local à un moment précis.

Pour chacune des classes décrites ci-dessous, vous devez créer les fichiers d'en-têtes .h et les fichiers sources .cpp.

Classe Employe

La classe Employe est la classe de base pour tous les employés de l'université.

Les attributs suivants doivent être présents:

- Attributs *protected* :
  - **nom_** : Le nom de l'employé
  - **prenom_** : le prénom de l'employé
- Attribut privé :
  - **accesEmploye_** : un entier constant non signé qui représente le niveau d'accès minimal que possède un employé. **Il doit être initialisé avec la valeur 5**

Il faut implémenter les méthodes publiques suivantes:

- Un constructeur qui prend en paramètre le nom et le prénom de l'employé
- Des méthodes d'accès pour le nom et le prénom
- Une méthode *getNiveauAcces* qui retourne le niveau d'accès d'un employé
- Une méthode *getClasseEmploye* qui retourne le nom de la classe Employe. Utilisez l'operateur *typeid*

Une fonction amie operator << pour afficher tous les attributs de la classe Employe, ainsi que le nom de la classe.

Classe Professeur

La classe *Professeur* dérive de la classe *Employe*.

L'attribut privé suivant doit être présent :

- **accesProfesseur_** : un entier constant non signé qui représente le niveau d'accès que possède un Professeur. **Il doit être initialisé avec la valeur 10**

Il faut implémenter les méthodes publiques suivantes:

- Un constructeur qui prend en paramètre le nom et le prénom du professeur
- La surcharge de la méthode *getNiveauAcces*. Elle retourne la somme des niveaux d'accès d'un professeur et celui de la classe Employe
- La surcharge de la méthode *getClasseEmploye* qui retourne le nom de la classe Professeur. Utilisez l'operateur *typeid.*

Une fonction amie operator << pour afficher tous les attributs de la classe Professeur, ainsi que le nom de la classe (penser à utiliser l'operator << de la classe de base).

Classe AgentSecurite

La classe *AgentSecurite* dérive de la classe *Employe*.

L'attribut privé suivant doit être présent :

- **accesAgentSecurite_** : un entier constant non signé qui représente le niveau d'accès que possède un agent de sécurité. **Il doit être initialisé avec la valeur 20.**

Il faut implémenter les méthodes publiques suivantes:

- Un constructeur qui prend en paramètre le nom et le prénom de l'agent de sécurité
- La surcharge de la méthode *getNiveauAcces*. Elle retourne la somme des niveaux d'accès d'un agent de sécurité et celui de la classe Employe
- La surcharge de la méthode *getClasseEmploye* qui retourne le nom de la classe AgentSecurite. Utilisez l'operateur *typeid*

Une fonction amie operator << pour afficher tous les attributs de la classe AgentSecurite, ainsi que le nom de la classe (penser à utiliser l'operator << de la classe de base).

Classe Etudiant

Bien qu'un étudiant ne soit pas un employé, vous décidez de le faire dériver de la classe *Employe* pour vous simplifier la vie en vous disant qu'il suffit simplement d'initialiser son attribut privé d'accès avec la valeur appropriée et puis le tour est joué. Quel est le pire qui pourrait arrivé? (voir la question plus bas).

L'attribut privé suivant doit être présent :

- **accesEtudiant_** : un entier constant non signé qui représente le niveau d'accès que possède un Étudiant. **Il doit être initialisé avec la valeur 1**

Il faut implémenter les méthodes publiques suivantes:

- Un constructeur qui prend en paramètre le nom et le prénom de l'étudiant
- La surcharge de la méthode *getNiveauAcces*. Elle retourne le niveau d'accès d'un étudiant
- La surcharge de la méthode *getClasseEmploye* qui retourne le nom de la classe Etudiant. Utilisez l'operateur *typeid.*

Une fonction amie operator << pour afficher tous les attributs et le nom de la classe Etudiant (penser à utiliser l'operator << de la classe de base).

Classe PirateInformatique

Un pirate informatique est parvenu à modifier votre programme, ce qui lui a permis d'hériter de la classe *AgentSecurite*.

Il implémente les fonctions publiques suivantes afin de se faire passer pour un agent de sécurité :

- Un constructeur qui prend en paramètre un agent de sécurité
- La surcharge de la méthode *getNiveauAcces*. Elle retourne la somme des niveaux d'accès des classes Employe et AgentSecurite.

Classe RegleAcces

Cette classe définit le niveau d'accès requis pour accéder à un certain local à un certain moment de la journée.

Les attributs privés suivants doivent être présents:

- **local_ :** chaine de caractères représentant le numéro d'un local
- **niveauAccesRequis_ :** un entier non signé qui représente le niveau d'accès requis pour pouvoir accéder au local pendant la période spécifiée
- **periode_ :** période pendant laquelle cette règle s'applique

Il faut implémenter les méthodes publiques suivantes:

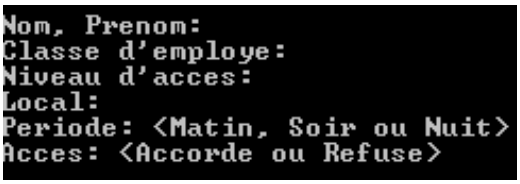
- Un constructeur par défaut. Il initialise le niveau d'accès requis à 1, le local à la chaine de caractères vide et la période à ''Matin''
- Un constructeur qui prend en paramètre le niveau d'accès requis, le local et la période
- Des méthodes d'accès pour les trois attributs
- La surcharge de l'opérateur d'égalité (==)

Classe SystemeSecurite

La classe SystemeSecurite renferme les règles d'accès aux différents locaux et décide si l'accès à un local est accepté ou refusé.

Les attributs privés suivants doivent être présents:

- **regles_ :** un vecteur d'objets de type RegleAcces
- **journalAcces_ :** un vecteur de string qui enregistre chaque tentative d'accès à un local
    - Voici le format d'un accès à un local :



Il faut implémenter les méthodes publiques suivantes:

1. Un constructeur par défaut
2. La méthode *accederLocal* qui prend en paramètre le nom, le prénom, la classe d'employé et le niveau d'accès d'une personne ainsi que le local et la période. La fonction retourne un booléen qui indique si l'accès a été accepté ou refusé. On accepte la personne dans le local si le niveau d'accès de la personne est plus petit ou égal au niveau d'accès de la règle, ne fois que l'on a trouvé la règle associée au local et la période dans l'attribut vecteur d'objets de type RegleAcces.
3. Elle inscrit aussi les informations sur la tentative d'accès dans le journal. Referez-vous à la capture d'écran ci-dessus pour le contenu de chaque tentative d'accès.

   La classe *stringstream* peut être utile pour l'implémentation de cette fonction :

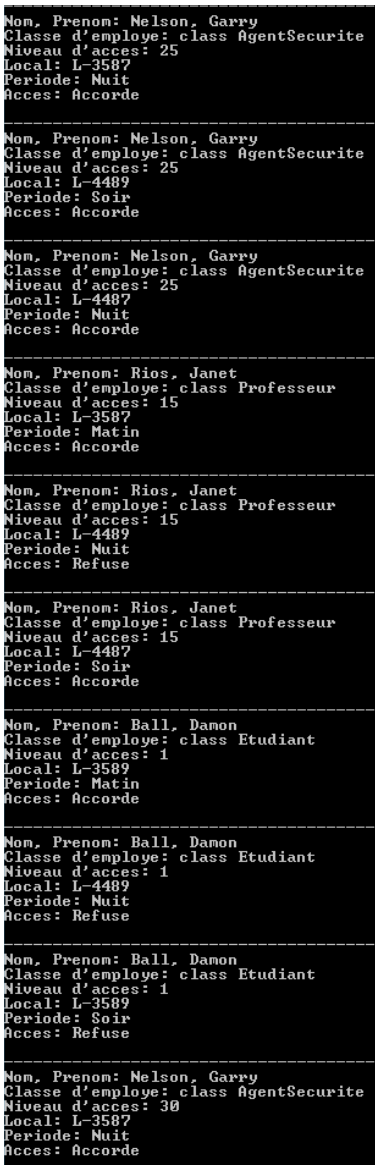
   http://www.dreamincode.net/forums/topic/95826-stringstream-tutorial/

4. Trois surcharges de la méthode *accederLocal* avec les paramètres suivants (pensez à réutiliser la fonction *accederLocal ci-dessus*) :
   a. Un agent de sécurité, le local, la période
   b. Un professeur, le local, la période
   c. Un étudiant, le local, la période
5. La méthode *ajouterRegle* qui ajoute une règle si elle n'est pas déjà présente
6. La méthode *imprimetJournal* qui affiche tout le contenu du journal (c'est à dire l'attribut **journalAcces_)**

   .

Main.cpp

Le programme principal contient des directives à suivre pour instancier différents objets et essayer les différentes méthodes implémentées. Voici l'affichage attendu:



Question

Répondre en dessous de la fonction main en commentaire.

Quelques jours après le lancement de votre système de contrôle d'accès, vous vous rendez compte, en inspectant le journal d'accès, qu'un étudiant a réussi à entrer dans le **local L-3589, le soir**. Cela nécessite un niveau d'accès d'une valeur de 5 (comme la classe Employe) alors qu'un étudiant à un niveau d'accès d'une valeur de 1. Sans modifier aucune classe et sans modifier les règles d'accès dans la fonction main, donnez un extrait de code qui permettrait à un étudiant d'avoir accès au local L-3589 pendant le soir si ce code été placé dans la fonction main. Expliquez clairement pourquoi votre code fonctionne.

Correction

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

- (10 points) Compilation et exécution exacte des différentes méthodes
- (03 points) Utilisation adéquate de l'héritage
- (02 points) Documentation du code et respect des normes de codage
- (02 points) Utilisation correcte du mot-clé *const*.
- (03 points) Réponse à la question