TP Noté 2

08/12/2021

*Durée : **2h00**. Documents autorisés. Faites une archive (tar.gz ou zip) ne contenant que les sources et déposez-le sur l'espace Moodle dans le devoir « TP Noté 08/12/21 ».*
Il est inutile d'écrire les accesseurs qui ne sont pas demandés et dont vous n'avez pas besoin.

1. Écrire une/des classe(s) pour représenter différents types de personnes qui interviennent dans un des projets donnés dans le cadre d'une formation universitaire :
   `etudiantM1` (on mémorisera numéro d'étudiant unique attribué automatiquement à la construction, nom, langages de programmation maîtrisés (**au moins deux** parmi `cplusplus`, `javascript`, `php`, `python`) ;
   `etudiantM2` (on mémorisera numéro d'étudiant unique attribué automatiquement à la construction, nom, niveau d'expertise en gestion de projets (un entier compris **entre 1 et 10**) ;
   `enseignant` (nom, numéro de bureau).
   Pour cette/ces classe(s), vous écrirez un constructeur[1] permettant de fixer toutes les informations sur la personne. Il n'y aura pas de mutateurs : les valeurs des attributs seront fixées à la construction.
2. Écrire une méthode `typepers` retournant le type de personne (M1, M2, Enseignant).
3. Écrire pour les étudiants de M1 une méthode `maitriselangage` prenant comme paramètre un langage et retournant vrai si l'étudiant maîtrise ce langage, faux sinon.
4. Écrire un opérateur de sortie sur flux permettant de sortir sur le flux l'ensemble des informations mémorisées sur une personne, y compris le type de la personne.
5. Un *expert* est soit un enseignant, soit un M2 ayant un niveau d'expertise supérieur à 7, soit un M1 maîtrisant au moins 3 langages. Écrire une méthode `expert` retournant vrai si la personne est un expert, faux sinon.
6. Écrire une classe `exceptionuniversite`, sous-classe de `std::exception` dont le constructeur prendra comme argument une chaîne et dont la méthode `what` retournera la chaîne en question.
7. Écrire une classe `universite` contenant un ensemble de personnes. *Il est conseillé d'avoir une vue sur la totalité du problème (notamment les questions 9 et 11) avant de traiter cette question.*
8. Ajouter à `universite` une méthode `ajout` permettant d'ajouter une personne. Cette méthode refusera d'ajouter une personne s'il existe déjà une personne de même nom dans l'université, dans ce cas-là, elle lèvera une `exceptionuniversite` avec le message « Nom en double : *lenom* ».
9. Écrire une méthode `rechercherdevs` prenant comme paramètre un langage et retournant l'ensemble des étudiants qui maîtrisent ce langage.
10. Écrire une méthode `saisiem2` demandant à l'utilisateur de saisir les informations permettant de créer un `etudiant` de M2 et de l'ajouter à l'université. Si la création de l'objet est impossible (niveau non compris entre 1 et 10, présence d'une personne de même nom), un message sera affiché, et l'utilisateur recommencera sa saisie[2]. ***Important** : Si vous faites la question 15 (`saisiem2qt`), inutile de faire cette question 10 : les points de `saisiem2` vous seront donnés si vous écrivez correctement cette partie du traitement dans `saisiem2qt` (qui n'est rien d'autre que ce qui vous est demandé ici, mais sous la forme d'une fenêtre Qt).*
11. Dans le cadre d'un enseignement de M1-M2, nous organisons des projets composés d'étudiants de M1, M2, encadrés par des enseignants. Écrire une classe `projet` pour représenter cela. Une personne peut faire partie de plusieurs projets. À la construction, un projet sera « vide ».

1 Dans la correction, le respect des conditions particulières (au moins deux langages de programmation maîtrisés, niveau compris entre 1 et 10) sera pris en compte. Ainsi, il ne devrait pas être possible de construire un étudiant de M1 avec moins de deux langages maîtrisés, ou un étudiant de M2 avec un niveau inférieur à 1 ou supérieur à 10.

2 La bonne façon de faire est de ne pas écrire ici un code qui teste si le niveau saisi est compris entre 0 et 10 ou s'il existe déjà une personne de ce nom : il faut appeler les méthodes, et prendre en compte le succès ou l'échec de ces méthodes.

12. Fournir une méthode `ajout` permettant d'ajouter une personne à un projet en respectant les conditions suivantes : il doit y avoir plus (ou autant) d'étudiants de M1 que de M2 ; il ne peut y avoir plus de deux enseignants. En cas d'erreur lors de l'ajout, la méthode lèvera une `exceptionuniversite` avec un message décrivant le problème.

13. Modifier la classe `universite` pour y rajouter un ensemble de `projet`.
Écrire une méthode `ajout` permettant de rajouter un `projet` à une `universite`. (aucune vérification à faire)

14. Écrire une méthode `verifierprojets` qui vérifie que les projets d'une `universite` sont composés uniquement de personnes qui font partie de l'`universite`. Cette méthode retournera la liste des personnes impliquées dans au moins[1] un projet et qui ne sont pas membres de l'`universite`.

15. Écrire une classe `saisiem2qt` faisant la même chose que `saisiem2`, mais dans une fenêtre Qt composée de deux `QLineEdit` pour la saisie, un bouton permettant de faire l'ajout, un bouton pour fermer, et un `QLabel` pour afficher le résultat de l'ajout. Le constructeur de cette classe prendra comme argument une `universite` existante et fera l'ajout sur cette instance. En cas d'erreur d'ajout, le message d'erreur sera affiché dans le `QLabel`. Si l'ajout se fait correctement, ce `QLabel` affichera « Étudiant ajouté : *nom* », et la boîte ne se fermera pas afin que l'utilisateur puisse ajouter d'autres étudiants.

Barème donné à titre indicatif

Question	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**main**	**Total**
Points	5	1	0,5	1,5	1,5	0,5	1	2	1,5	2	0,5	2,5	0,5	2	3	0,5	**25,5**

1 En d'autres termes si une personne est membre de deux projets, mais ne figure pas dans la liste des personnes de l'université, elle figurera **une** fois dans la liste retournée (et non deux fois).