Contrôle continu 2

*Durée : **2h00**. Documents autorisés. Faites une archive (tar.gz ou zip) ne contenant que les sources et postez-le sur l'espace Moodle dans le devoir « Contrôle Continu 2 ».*

1. Écrire une classe `formation` représentant une formation effectuée à l'université. Une formation est représentée par un niveau d'études, parmi 5 niveaux (L1, L2, L3, M1, M2), et un nom (ex : `Informatique`). Déclarer la classe et donner constructeur, accesseurs, opérateurs de comparaison `==` et `!=`, méthode `tostring` retournant une chaîne du type « `L3 - Informatique` ».

2. Un `enseignement` sera représenté par un nom de matière (ex : POO), un type d'enseignement (parmi `Cours`, TD, TP) et la `formation` dans lequel l'enseignement a lieu (il n'est pas demandé d'utiliser ici un pointeur sur formation, vous pouvez vous contenter d'avoir un attribut de type `formation` dans `enseignement`). Déclarer la classe et donner constructeur, accesseur à la formation, et méthode `tostring` retournant une chaîne du type « `L3 – Informatique : TP POO` ».

3. On désire maintenant représenter des informations sur des personnes. Chaque personne a un nom (chaîne), et il y a trois types de personnes qui fréquentent l'université :
   les étudiants, qui ont un numéro unique d'étudiant (qui devra être attribué automatiquement à la construction) et sont inscrits dans une `formation` ;
   les enseignants, qui ont un grade parmi deux valeurs possibles : MCF et Professeur. Il faut aussi mémoriser l'ensemble des `enseignements` assuré par chaque enseignant ;
   les personnels administratifs qui gèrent les formations : chaque personnel administratif a la responsabilité d'un ensemble de formations.
   Définir une (des) classe(s) pour représenter ces personnes, et les constructeurs associés.

4. Écrire un opérateur de sortie sur flux sortant les informations sous cette forme :
   `Etudiant – Nom – Formation – Numéro`
   `MCF` *ou* `Professeur – Nom - Nombre d'enseignements`
   `Administratif – Nom – Noms des formations gérées`.

5. Écrire une méthode `intervientdansniveau` prenant comme paramètre un niveau d'études (L1 à M2) et retournant un booléen valant `true` si la personne fait partie de ce niveau d'études et `false` sinon : Dans le cas d'un étudiant, il s'agira de tester s'il est inscrit dans une formation de ce niveau, dans le cas d'un enseignant, il s'agira de tester s'il effectue au moins un enseignement à ce niveau, et dans le cas d'un administratif, s'il gère une formation de ce niveau. Essayer d'utiliser au mieux les services offerts par la bibliothèque standard afin d'écrire le code le plus court et le plus clair possible.

6. Écrire une classe `universite` mémorisant un ensemble de personnes fréquentant cette université ainsi qu'un ensemble d'enseignements devant être dispensés.

7. Définir si nécessaire des méthodes afin que la mémoire soit gérée correctement et qu'il soit possible de réaliser des copies d'`universite` par les moyens habituels de C++. La copie obtenue doit être totalement indépendante de l'objet copié.

8. Écrire une méthode `ajouter` prenant comme paramètre un enseignement. Si l'enseignement existe déjà dans l'ensemble des enseignements mémorisés, la méthode lèvera une exception de type `exceptionuniversite`, sous-classe de `std::exception`, que vous définirez, et qui ne comporte qu'un seul attribut : un message d'erreur (chaîne).

9. Écrire une méthode `ajouter` prenant comme paramètre une personne (pouvant être un étudiant, enseignant ou administratif) et ajoutant cette personne à l'`universite`. Aucune vérification ne sera effectuée dans le cas de l'ajout d'un enseignant ou d'un administratif. Dans le cas d'un étudiant, la méthode vérifiera que l'ensemble des personnes de l'université ne contient pas déjà un étudiant ayant le même numéro. Si c'est le cas elle lèvera une exception de type `exceptionuniversite`

10. Écrire une méthode `saisiecoursl1` permettant à l'utilisateur de saisir des informations sur un enseignement de type Cours de L1 et l'ajoutant à l'`universite`. La méthode demandera donc le nom de la formation et le nom du cours et tentera de l'ajouter à l'`universite`. Si l'ajout est refusé par la

méthode `ajouter`, la méthode demandera un nouveau nom de cours (et uniquement nom) jusqu'à ce que l'ajout soit fait. Par exemple, l'exécution de la méthode sur une université contenant déjà un cours Algo1 en L1 MPCIE provoquera le résultat suivant (en italique, ce qui est saisi par l'utilisateur)

```
Nom formation : MPCIE
Nom cours : Algo1
Enseignement déjà présent
Nom cours : Math
OK
```

11. Écrire une méthode `supprimerformation` qui supprime toute référence à cette formation dans l'université (étudiants inscrits à cette formation, enseignements de cette formation, enseignements donnés par les enseignants, responsabilité de la formation par un administratif).

12. Écrire une classe `saisieenseignementsqt` donc le constructeur prendra comme paramètre une `universite` et permettant à l'utilisateur de saisir un nom d'enseignement (saisie libre), un type (parmi une liste déroulante), niveau (parmi une liste), nom formation (saisie libre), puis cliquer sur un bouton ajouter et ajoutant l'enseignement à l'université. Si l'ajout a été impossible, un texte d'erreur sera affiché dans un `QLabel` invitant à recommencer, sinon, la fenêtre sera fermée.