Projet n°2 de programmation objet

1 Objectif

L'objectif du projet est de mettre en œuvre la programmation objet pour simuler un système de messagerie électronique. Un tel système comporte des utilisateurs et des listes de diffusion. Parmi ces utilisateurs, certains sont des administrateurs disposant de droits spécifiques. Parmi ces administrateurs, *root* est l'administrateur principal créé en même temps que le système lui-même.

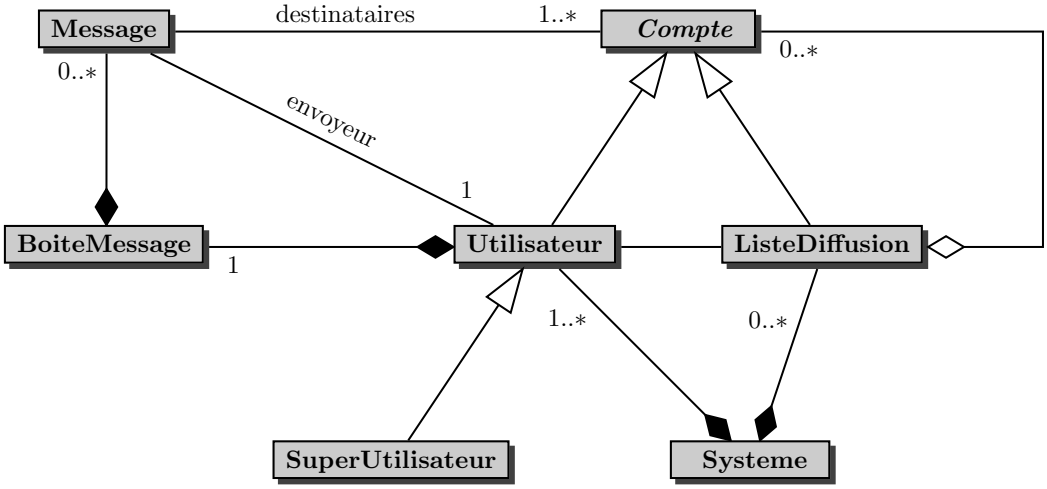
– Chaque utilisateur a un nom de *login*, un mot de passe et une adresse de messagerie. De plus, il possède une boîte de messages et il peut envoyer, lire ou supprimer un message. Seul un administrateur peut créer ou détruire un utilisateur. Seul *root* peut créer ou détruire un administrateur.

– Une liste de diffusion rassemble des utilisateurs et d'autres listes. Une liste est la propriété d'un utilisateur (celui créant la liste). Une liste a un accès public, ainsi tout le monde peut y poster un message, ou restreint quand seuls les membres de la liste ou les administrateurs peuvent y poster un message. Le choix de l'accès est fait au moment de la création de la liste.

Chaque message est constitué d'un envoyeur, d'une liste de destinataires, d'une date, d'un sujet et d'un texte. Une boîte de messages regroupe des messages pouvant être triés de différentes façons, par exemple par dates ou par noms des envoyeurs de manière croissante ou décroissante.

2 Travail

Un exemple de diagramme de classes est proposé ci-dessous, sachant qu'il est possible (permis) de le modifier pour considérer de nouvelles classes ou des interfaces. Dans ce diagramme, la classe `Systeme` permet de gérer l'ensemble des utilisateurs et des listes de diffusion et la classe `Compte` constitue le support de la récursivité entre listes.



2.1 Utilisateurs et listes

En considérant le diagramme proposé, il est recommandé dans un premier temps de spécifier et d'implémenter les classes `Compte`, `Utilisateur`, `SuperUtilisateur` et `ListeDiffusion`. Ces classes seront munies des méthodes nécessaires à la création des instances, de méthodes de comparaison et de méthodes d'affichage (utiles pour la phase de mise au point). La classe des listes sera équipée de méthodes permettant d'ajouter ou de supprimer des abonnés, et d'une méthode réalisant un test d'appartenance. Avant de passer à la section suivante, il faudra tester ces classes, en particulier pour vérifier la correction de la structure récursive des listes.

2.2 Système

Dans un second temps, il est recommandé de programmer la classe `Systeme`. Rappelons que l'utilisateur *root* est créé en même temps que le système. Il faudra munir cette classe de méthodes d'ajout et de suppression d'utilisateurs et de listes. En particulier, la suppression d'une entité du système (utilisateur ou liste) doit entraîner sa suppression des listes auxquelles elle est abonnée.

2.3 Messages et boîtes

Dans un troisième temps, il faudra gérer les messages, ce qui concerne en particulier la programmation des classes `Message` et `BoiteMessage` et la modification des autres classes pour gérer l'envoi et la réception des messages. Il sera nécessaire de prendre en compte les droits des utilisateurs et les modes d'accès des listes pour distribuer les messages.

2.4 Question bonus : Analyseur de commande

Cette question vise à créer un programme permettant de lire des commandes au clavier pour gérer un système de messagerie, la commande `quit` permettant de terminer le programme. Un exemple est donné ci-dessous.

```
localhost> login root
   enter password : ********
root@localhost> create user martin-l provisoire leo.martin@univ-nantes.fr
   user 'martin-l' created
root@localhost> create list x3i0020 x3i0020@univ-nantes.fr
   list 'x3i0020' created
root@localhost> logout
localhost> login martin-l
   enter password : **********
martin-l@localhost> quit
```

Il faudra associer une commande à chaque fonctionnalité du système, par exemple `send` pour envoyer un message, `ls` pour consulter sa boîte, `attach` pour s'abonner à une liste, `detach` pour se désabonner, etc. Le bonus du bonus est de sérialiser les objets pour la persistance des données.

3 Réalisation et rendu du projet

Les règles sont celles du projet précédent. La date de rendu (programme source et rapport) est fixée au plus tard le mardi 17 décembre à 23h59.