

TD-TP 5– M2103

1 - Suite de l'application Messagerie

Il s'agit de terminer l'application messagerie : les classes MailClient et la classe scénario MailTest.

La classe MailClient

Elle possède deux attributs : une chaîne de caractères pour le nom du client et une référence sur le serveur, c'est-à-dire une instance de **MailServer**.

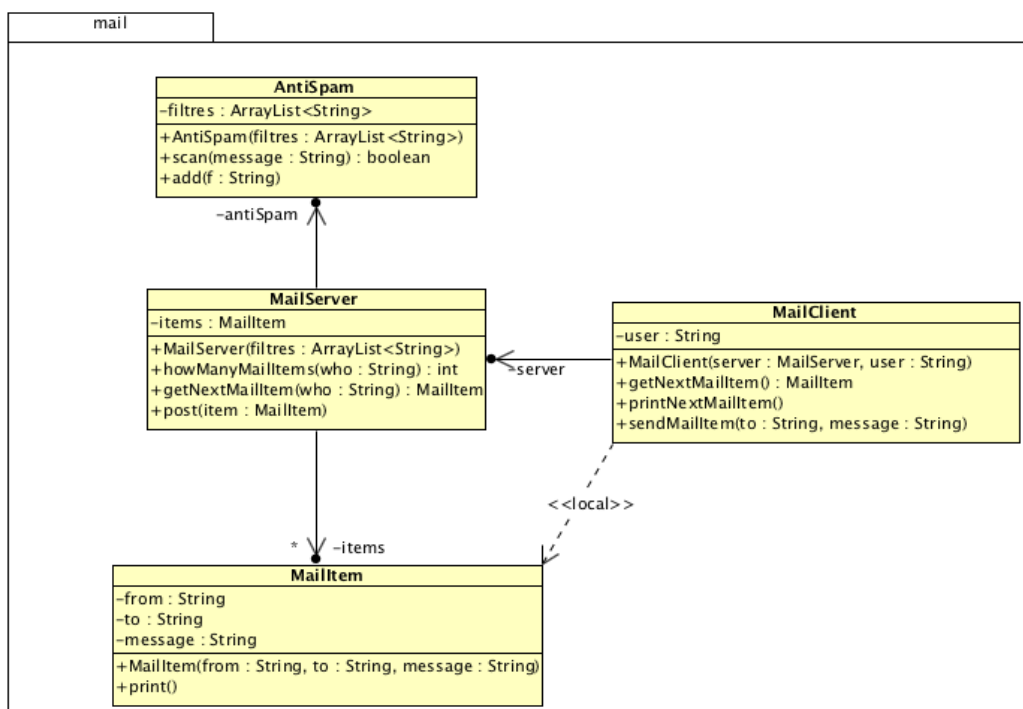
Les méthodes

- – **MailClient(MailServer server, String user)** le constructeur initialise les attributs avec les paramètres.
- – **MailItem getNextMailItem()** retourne un MailItem en envoyant au serveur le message **getNextMailItem(user)** où **user** est le nom du client.
- – **printNextMailItem()** fait appel aussi au serveur avec le message **getNextMailItem(user)** et imprime le message (méthode print de MailItem) s'il s'agit bien d'un message.
- – **sendMailItem(String to, String message)** reçoit en paramètre le nom du destinataire et le texte du message, crée une instance de MailItem et envoie le message **post()** avec le mailItem en paramètre au serveur.

Remarque : Pour que cette application fonctionne on comprendra que tous les objets de type MailClient doivent partager le même objet MailServer.

2- Extension du système de mail : détection de spam

On souhaite maintenant détecter les messages qui sont des spams, c'est-à-dire des messages qui contiennent de mots particuliers. A chaque réception d'un MailItem, le serveur de messages (MailServer) doit vérifier le contenu du message en faisant appel à un AntiSpam.



Il faut pour cela concevoir une classe `AntiSpam` possédant une liste de filtres pour analyser le texte d'un `MailItem`. Pour simplifier nous allons considérer que les filtres sont des mots et que l'antispam a comme attribut cette liste (implémentée avec un `ArrayList`).

Les méthodes pour `AntiSpam` :

- un constructeur qui reçoit en paramètre la liste de mots (spams).
- `void add(String f)` : pour ajouter le mot `f` à la liste de filtre
- `boolean scan(String message)` : analyse le texte d'un message (un objet `String`) et recherche si un des mots de la liste est contenu dans ce texte. Cette méthode doit retourner `true` si le message analysé est un spam ou `false` sinon.

Les modifications de la classe `MailServer` :

- ajout d'un attribut `antiSpam`
- modification du constructeur qui reçoit la liste de filtres et crée l'objet `antiSpam`
- modification de la méthode `post` qui lance l'analyse de l'`antiSpam` avant d'ajouter le message

Travail minimum à rendre

- Implémenter en Java la classe `MailClient` dans le package `mail` sans oublier la javadoc
- Terminer par implémenter la classe de `MailTest` qui est un scénario de l'ensemble de l'application et qui se trouve à l'extérieur du package `mail`.
- Déposer le code Java de toutes les classes de l'application messagerie (`MailItem`, `MailServer`, `MailClient`, `MailTest`) et la javadoc générée de l'ensemble des classes.

Travail supplémentaire

- Déposer l'ensemble de vos classes avec les mêmes exigences que ci-dessus mais avec les modifications, ainsi que la classe `AntiSpam` et un scénario d'utilisation.