TD-5– R2.01

Semaine 8

1 - Suite de l'application Messagerie

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

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. Imprime un message disant qu'il n'y a pas de nouveau message si le retour du serveur est null.
- – **`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 des mots particuliers. A chaque réception d'un objet MailItem, le serveur de messages (`MailServer`) doit vérifier le contenu du message en faisant appel à un AntiSpam.

Voir le diagramme de classes ci-dessous.

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).

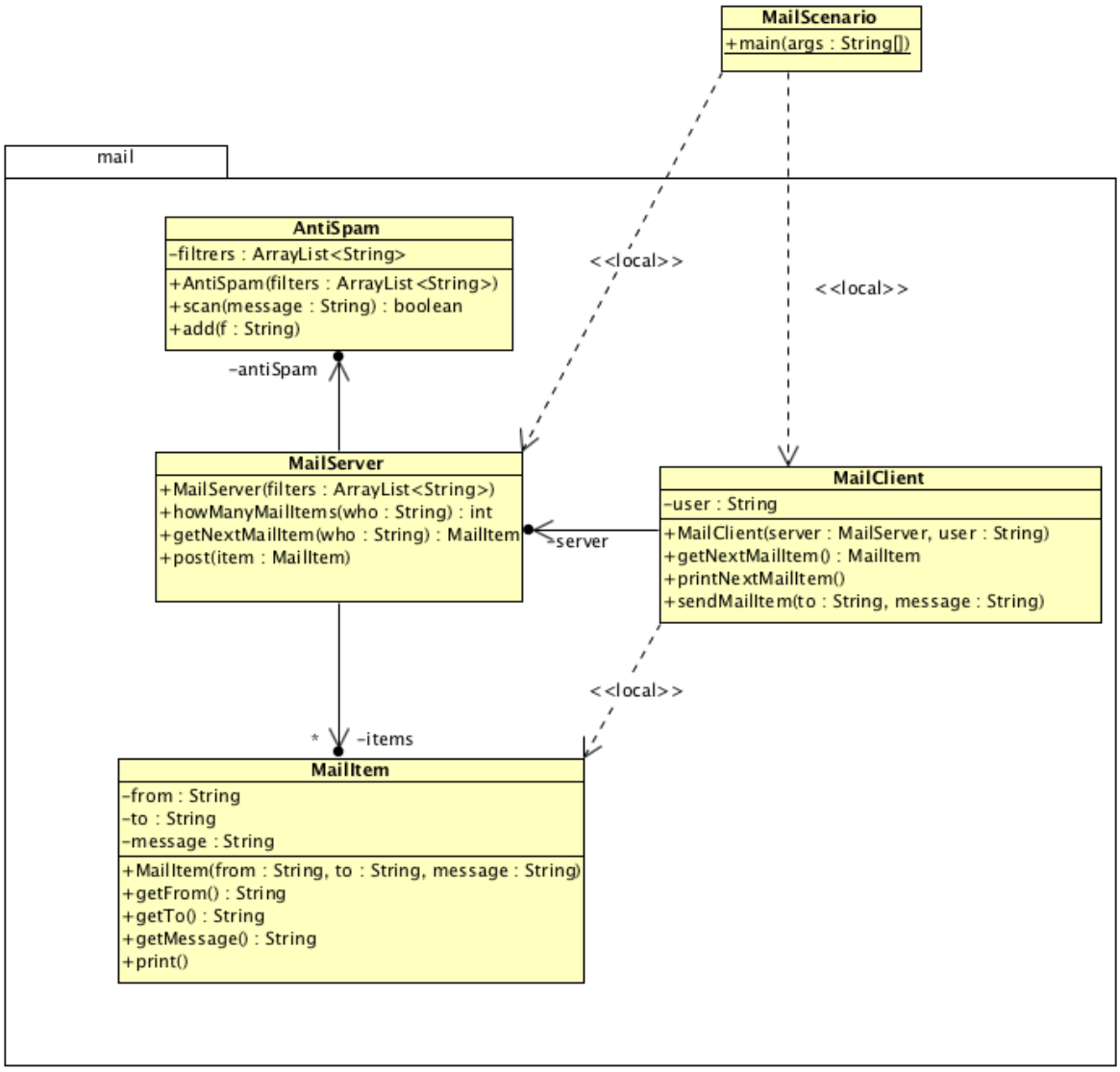
Les méthodes pour AntiSpam :

- 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 des filtres 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



Faire :

- Le code de la classe MailClient dans le package mail.
- Le code de la classe AntiSpam dans le package mail.
- Les modifications nécessaires à la classe MailServer

TP 5– R2.01

Semaine 8

Travail à rendre : code source + javadoc générée

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

Questions supplémentaires pour les plus rapides.

1. Ajouter un attribut `ArrayList<MailItem>` spams dans MailServer. Les mails considérés comme des spams doivent être ajoutés à cette liste (et pas à items). Leur message doit être précédés de « [SPAM] ». Ajouter un moyen pour permettre au MailClient de lire les spams qui lui sont destinés.
2. Ajouter un attribut `ArrayList<String> spammers` qui contient le noms des expéditeurs de spams. Dès qu'un spam est envoyé, l'identifiant de l'expéditeur est ajouté à la liste `spammers` s'il n'y est pas déjà. Le message d'un `MailItem` non détecté comme spam mais provenant d'un spammer de la liste doit être précédé de « [Attention : SPAMMEUR NOTOIRE] ». Ce mail est ensuite ajouté à items.
3. Ajouter une fonction pour afficher le contenu du server (liste des mails, listes des spams, liste des spammeurs). Soignez l'affichage.
4. Rajouter une façon de faire de l'envoi de mail groupé (envoi à plusieurs destinataires).
5. L'antispam ne doit pas être sensible à la casse.
6. Faire en sorte que chaque client puisse conserver ses mails (et spams) lus dans une même liste.
7. Faire en sorte que les `MailItem` contienne également la date et l'heure de création du mail (format : yyyy/mm/dd HH:mm:ss). Ajouter cette information dans l'affichage du mail.
8. Vérifier que le destinataire d'un mail a une adresse mail correcte (usage du package Regex) type abc@defg.hij
9. Faire une fonction pour afficher les mails lus triés par expéditeur et une fonction pour afficher les mails lus triés par date du mail.