Module M2103 – Java
TP 7 : Interfaces et Classes Abstraites

L'objectif de ce TP est l'étude des mécanismes de rappel en utilisant interfaces ainsi que classes et méthodes abstraites. Vous implémenterez une version du schéma Emetteur-Souscripteur pour la bibliothèque informatisée.

Partie I : Mécanismes de Rappel

Lorsqu'un membre fait une réservation pour un document, il désire être notifié lorsque le document devient disponible. Ceci peut être mis en oeuvre en utilisant le mécanisme de rappel. Cet exercice vous guide dans l'implémentation de rappels pour simuler les notifications faites aux membres de la bibliothèque.

Tâche 1 : Infrastructure pour le mécanisme de rappel – L'un des prérequis pour la mise en place d'un mécanisme de rappel est l'écriture d'une interface avec des méthodes qui seront appelées pour établir la notification.
Ecire une interface nommée `Notifiable`. Les classes qui implémenteront cette interface pourront être notifiées concernant la disponibilité d'un document placé préalablement dans la section spéciale 'réservations'. La méthode suivante devra être mise en oeuvre :

> **docDisponible(DocBibliotheque)** - ce message est envoyé à un objet pour lui notifier le fait que le `DocBibliotheque` spécifié en tant que paramètre de la méthode est désormais disponible.

Tâche 2 : Ecriture de classes souscriptrices – Tous les membres doivent être notifiés lorsque les documents qu'ils ont réservés deviennent disponibles. Toutefois, les deux types de membres réagiront de manière différente : les étudiants iront directement emprunter le document alors que les membres du personnel attendront généralement quelques jours en raison de leur emploi du temps chargé.
Faire en sorte que la classe `MembreBibliotheque` implémente l'interface `Notifiable`. Vous n'aurez toutefois pas besoin d'écrire le corps de la méthode de l'interface dans cette classe. Vous remarquerez que pour que la classe `MembreBibliotheque` soit compilée, vous devrez la rendre abstraite. Pourquoi ?

Si vous essayez de compiler les sous-classes, le compilateur vous indiquera qu'elles doivent être déclarées abstraites. Toutefois, rendre ces classes abstraites serait inapproprié sachant que `MembrePersonnel` et `MembreEtudiant` sont bien des classes concrètes. Vous devrez donc écrire le corps de la méthode *docDisponible* dans chacune de ces classes :

- Pour la classe `MembrePersonnel`, il suffira d'afficher un message à l'écran de la sorte : "Le document *d* qui a été réservé par le membre du personnel *mp* est désormais disponible à l'emprunt au bureau des réservations".
- Pour la classe `MembreEtudiant`, il faudra afficher un message notifiant que le document est disponible et demandant à l'étudiant (l'utilisateur) s'il désire l'emprunter ou non. Si l'utilisateur répond par l'affirmative, alors le document sera emprunté par le `MembreEtudiant` immédiatement.

Tâche 3 : Ecriture des méthodes de souscription dans la classe publiante – Dans le cadre du TP2, vous avez écrit deux méthodes dans la classe `DocBibliotheque` pour pouvoir réserver un document et annuler la réservation. Celles-ci peuvent être considérées comme des méthodes de souscription et d'annulation de souscription, formant de la sorte une base pour le mécanisme de rappel. Ces méthodes prendront en paramètre une variable de type `Notifiable` (au lieu de `MembreBibliotheque`). Vous implémenterez ces méthodes de manière à ce que les contraintes suivantes soient respectées :

- La classe `DocBibliotheque` doit se rappeler la valeur du paramètre passé à l'appel de la méthode de réservation de document. Celui-ci représente la personne qui doit être informée lorsque le document est retourné. C'est le processus de souscription.
- La méthode d'annulation de réservation doit verifier que l'objet passé en paramètre est le même objet ayant fait la reservation. S'ils correspondent, alors l'objet préalablement enregistré ne doit plus l'être. C'est le processus d'annulation de souscription.

Vous ne serez toutefois pas en mesure de compiler avec succès jusqu'à ce que vous terminiez la tâche 4...

Tâche 4 : Mise à jour de la classe test – Comme vous avez précédemment modifié les paramètres des méthodes de réservation et d'annulation de réservation, vous devrez donc mettre à jour la classe test.

Tâche 5 : Ecriture du processus de notification dans la classe émettrice – Pour que l'objet soit notifié du retour du `DocBibliotheque`, qui peut donc être emprunté, le `DocBibliotheque` doit envoyer un message

de notification en utilisant la (ou les) méthode(s) de l'interface de rappel. Cette notification doit intervenir pendant le traitement de la méthode de retour de document dans la classe `DocBibliotheque`, puisque le changement d'état de l'objet s'y déroule (les notifications servent généralement à informer d'autres objets d'un changement d'état).

Modifier le corps de la méthode de retour d'un document de manière à ce que si elle determine qu'une reservation a été faite sur un document, alors l'objet qui a fait la reservation sera notifié par l'appel de sa méthode *docDisponible*. Ce `DocBibliotheque` particulier qui génère et envoie cette notification doit être passé en paramètre du message. Cette alerte doit être générée après le changement d'état du document.

Partie II : Autre Classe Souscriptrice

Dans la partie précédente, l'interface `Notifiable` était implémentée par la classe `MembreBibliotheque`, et donc également par ses classes filles. Dans cette partie, vous écrirez une autre classe (qui n'est pas liée à la classe `MembreBibliotheque`) implémentant l'interface `Notifiable`.

Le personnel de la bibliothèque doit occasionnellement retirer certains documents de la bibliothèque pour une variété de raisons (e.g. faire des photocopies, remplacer un document abîmé…). Ces documents ne sont donc pas 'empruntés' mais 'utilisés' dans le cadre de la bibliothèque. Toutefois, si un membre de la bibliothèque a emprunté le document lorsqu'un membre du personnel de la bibliothèque (qui est un employé, et non un membre de la bibliothèque) en a besoin, ce dernier doit signaler le document (ce qui est équivalent à effectuer une réservation).

- **Tâche 1** – Ecrire une classe nommée `EmployeBibliotheque`, qui doit être déclarée comme implémentant l'interface `Notifiable`. L'employé doit avoir une identification d'employé (qui comprend jusqu'à 4 caractères) et un nom.
- **Tâche 2** – Implémenter la méthode *docDisponible* de manière à ce que lorsque le document devient disponible pour un `EmployeBibliotheque`, il est supprimé du `CatalogueBibliotheque` (c'est-à-dire qu'il n'apparaît plus dans le catalogue lorsque la méthode affichant tous les documents de la bibliothèque est appellée).
- **Tâche 3** – Mettre à jour la classe test de manière à pouvoir instancier des objets `EmployeBibliotheque` qui pourront utiliser les documents de la bibliothèque. On vérifiera donc que les méthodes implémentées dans cette partie fonctionnent correctement.

Partie III : Pour Aller Plus Loin…

Dans le cadre du TP précédent, nous avons caractérisé des documents qui ne peuvent être empruntés, comme par exemple les `DocURL`.

Ecrire une interface nommée `Empruntable` pour indiquer les documents pouvant être empruntés, rendus et réservés. Les méthodes mises en oeuvre dans l'interface permettront :

- L'emprunt d'un document par un `MembreBibliotheque` (nécessite donc un paramètre approprié)
- Le retour d'un document
- La prise en compte d'une réservation par un `MembreBibliotheque`
- L'annulation d'une réservation

Ces méthodes existent déjà bien sûr dans la classe `DocBibliotheque` mais ne sont pas pertinentes pour tous les documents de la bibliothèque. On considèrera les étapes suivantes pour modifier les classes filles `Livre` et `CD` :

- S'assurer du fait que `Livre` et `CD` implémentent l'interface `Empruntable`
- Extraire les 4 méthodes de la classe `DocBibliotheque` pour éviter que toutes les classes filles puissent en hériter
- Et également…

Toutefois, désormais les processus d'emprunt, de retour et de réservation ne sont plus encapsulés dans la classe `DocBibliotheque`. On les déplacera dans la classe `CatalogueBibliotheque` qui présente déjà des méthodes pour l'emprunt et la réservation de documents. On modifiera ces deux dernières en s'assurant du fait qu'un document particulier est un document `Empruntable` (utilisation de *instanceof*).