Projet Smalltalk
Système de gestion de bibliothèque

Kévin Vythelingum

9 juin 2013

1 Introduction

Le système de gestion de bibliothèque est un exemple classique de conception orientée objet. Il consiste à créer un système permettant de gérer une bibliothèque élémentaire en fournissant les services habituellement rendus par une bibliothèque. Ainsi l'objectif de ce projet est de mettre en oeuvre un tel système proposant principalement les services suivants :

- permettre d'ajouter ou de détruire des livres à l'ensemble des livres possédés par la bibliothèque ;
- ajouter ou supprimer des emprunteurs de la liste des usagers de la bibliothèque ;
- enregistrer la sortie ou le retour d'un livre ;
- chercher un livre ;
- fournir la liste des livres empruntés et de leurs emprunteurs ;
- fournir la liste des livres empruntés dont la date limite de retour est dépassée.

Pour répondre à ce cahier des charges, je vais d'abord présenter mon analyse du problème puis exposer la conception globale du système. Enfin, un jeu de test sera explicité afin de montrer que le programme fourni répond au cahier des charges.

2 Analyse

L'analyse de ce problème consiste à identifier les différents cas, généraux et limites, des fonctionnalités proposées par le système de gestion de bibliothèque.

Permettre d'ajouter ou de détruire des livres à la bibliothèque.
Tout d'abord, l'utilisateur doit pouvoir créer et ajouter un livre à la bibliothèque en une seule action, mais il doit également pouvoir ajouter un livre qu'il avait déjà créé auparavant. Aussi, plusieurs exemplaires d'un même livre (même titre, même auteur, etc.) peuvent être présents dans la bibliothèque, il doivent donc être différenciés par un identifiant, que j'appellerai code barre. Il faut donc qu'il soit impossible d'ajouter un livre s'il porte le même code barre qu'un livre déjà présent dans la bibliothèque.

Ensuite, la fonction de destruction d'un livre doit permettre la suppression de ce dernier en fournissant uniquement son code barre. Le cas où on essaye de supprimer un livre avec un code barre inconnu doit être géré.

Ajouter ou supprimer des usagers à la bibliothèque.
De la même manière que pour les livres, l'utilisateur doit pouvoir créer et ajouter un utilisateur à la bibliothèque en une seule action, mais il doit également pouvoir ajouter un usager qu'il avait déjà créé auparavant. De plus, comme plusieurs usagers peuvent avoir leur nom en commun, il sera nécessaire de les distinguer à l'aide d'un identifiant.

Catalogue et liste des usagers.
Le système devra être capable d'afficher le catalogue de la bibliothèque ainsi que la liste des usagers inscrits, notamment pour vérifier que les livres et les usagers sont bien enregistrés ou supprimés de la bibliothèque.

Enregistrer la sortie ou le retour d'un livre.
La sortie ou le retour d'un livre doit pouvoir s'effectuer à partir du code barre du livre désiré. Pour la sortie, on demandera l'identifiant de l'abonné qui souhaite emprunter ce livre. La date d'emprunt sera conservée car les emprunts sont à durée limitée. Les règles suivantes devront être respectées :
- l'emprunt d'un livre déjà emprunté doit être refusé ;
- le retour d'un livre déjà disponible doit être signalé ;
- l'emprunt ou le retour d'un livre qui n'appartient pas à la bibliothèque doit afficher un message d'erreur.

Chercher un livre.
Il faut être capable d'afficher la liste des livres comportant une certaine information. Par exemple, on peut vouloir afficher tous les livres dont l'éditeur est "Folio".

En outre, le système doit permettre de retourner un livre et d'afficher les informations concernant celui-ci à partir de son code barre. Un message indiquera que le livre recherché n'appartient pas à la bibliothèque en cas de code barre inexistant.

Chercher un usager.
De la même manière que pour un livre, il doit être possible de rechercher un usager à partir de son identifiant. Un message indiquera si l'identifiant indiqué ne correspond pas à un usager de la bibliothèque.

Éditer la liste des livres empruntés et de leurs emprunteurs.
Le système doit être capable d'afficher la liste des livres empruntés en indiquant leur code barre, titre, date d'emprunt et l'identifiant de l'emprunteur.

Éditer la liste des livres empruntés dont la date limite de retour est dépassée.
Le système doit être capable d'afficher la liste des livres empruntés dont la date limite de retour est dépassée, en indiquant leur code barre, titre, date d'emprunt et l'identifiant de l'emprunteur. Pour tester cette fonction, il sera nécessaire de faire une fonction qui permet de reculer la date d'emprunt d'un livre emprunté. Cette dernière indiquera qu'il faut d'abord emprunter le livre avant de le retarder dans le cas où on essairait de retarder un livre disponible.

Dans tous les cas, un message devra indiquer à l'utilisateur le succès ou l'échec de son action. Par ailleurs, il s'agira de rester aussi flexible que possible vis à vis des données qui ne sont pas fournies par le cahier des charges. Par exemple, il n'est pas précisé si l'on doit connaître le genre d'un livre (policier, romantique, fantastique, ...) donc la fonction de recherche doit pouvoir être adaptée rapidemment et facilement si cette information doit être indiquée ou non. Aussi, l'interface utilisateur doit être la plus simple et accessible possible, en utilisant des boîtes de dialogue par exemple, ce qui permettra de masquer d'autant plus la structure des données.

3 Conception

Pour concevoir ce système, ma démarche a d'abord été d'identifier les différentes entités composant la bibliothèque. Ensuite, j'ai défini quelles informations et quelles fonctionnalités seraient proposées par chaque entité avant d'établir la hiérarchie entre les entités.

Ainsi, il a été naturel de concevoir les entités Bibliothèque, Livre et Usager. Il existe une relation entre un livre et un usager qui est celle d'emprunt. Il était possible de représenter celle-ci au niveau de la bibliothèque qui tiendrait à jour une liste des emprunts, mais j'ai préféré mettre l'identifiant de l'emprunteur en attribut d'un livre emprunté. Ce choix est justifié par le fait qu'un livre emprunté à un comportement propre et il me paraît justifié que l'on puisse accéder à l'emprunteur à partir du livre.

Bibliothèque. La bibliothèque comporte deux dictionnaires, l'un constitué de l'ensemble des livres, indexé par leur code barre, l'autre constitué de l'ensemble des usagers. Concernant les usagers, j'ai choisi de les identifier par leur adresse e-mail, qui est forcément unique. De plus, c'est à cette bibliothèque qu'on enverra les messages exécutant les fonctionnalités du cahier des charges.

Livre. Un livre est identifié de manière unique par un code barre. Il comporte également des champs en attribut indiquant son titre, auteur, éditeur, genre et statut.

Mis à part les autres attributs qui contiennent une chaîne de caractère, l'attribut statut, qui permet de distinguer un livre emprunté d'un livre disponible, est particulier. En effet, ce dernier a amené à la conception d'une classe abstraite `Statut` constitué de deux sous classes `StatutDisponible` et `StatutEmprunté`. Cette organisation permet de traiter facilement de manière différente les méthodes qui n'ont pas le même comportement sur un livre emprunté et un livre disponible. Ainsi, lorsqu'un livre recevra une telle méthode, il transmettra ce message à son statut qui implémentera un comportement différent selon le statut du livre. Par exemple, lorsqu'on voudra emprunter un livre, on enverra le message *emprunter* à l'instance du livre désiré, qui transmettra à son tour le message *emprunter* à son statut. Alors, si le statut est un `StatutDisponible`, le livre sera effectivement emprunté et le statut passera à un `StatutEmprunté` et si le statut est un `StatutEmprunté`, il répondra que le livre est déjà emprunté.

De plus, en vue de la fonction de recherche par mot clef, une méthode de recherche renverra un booléen selon que le mot clef passé en argument de la méthode correspond à un attribut du livre ou non. Ainsi, la fonction de recherche de la bibliothèque aura juste à faire appel à cette méthode sur chacun des livres de son catalogue.

Pour finir, un livre est capable de s'afficher de deux manières différentes. Soit en *mode ligne* de manière à fournir toutes ses informations sur une seule ligne, soit en *mode fiche* qui permet d'avoir une vue moins compacte des informations disponibles.

Usager. Comme annoncé précédemment, un usager est identifié par son adresse e-mail. Il comporte également des attributs précisant son nom, prénom, et adresse. De la même manière que pour les livres, il est capable de s'afficher de deux manières différentes.

L'organisation adoptée est flexible et évolutive dans la mesure où il est possible d'ajouter autant d'attributs que l'on veut aux classes `Livre` et `Usager`, sans perturber les fonctions de recherche et d'administration. Aussi, la classe `Statut` est susceptible d'accueillir d'autres sous-classes facilement comme un statut *réservé* si on souhaite instaurer un tel service ou encore *abîmé*, *perdu*, etc.

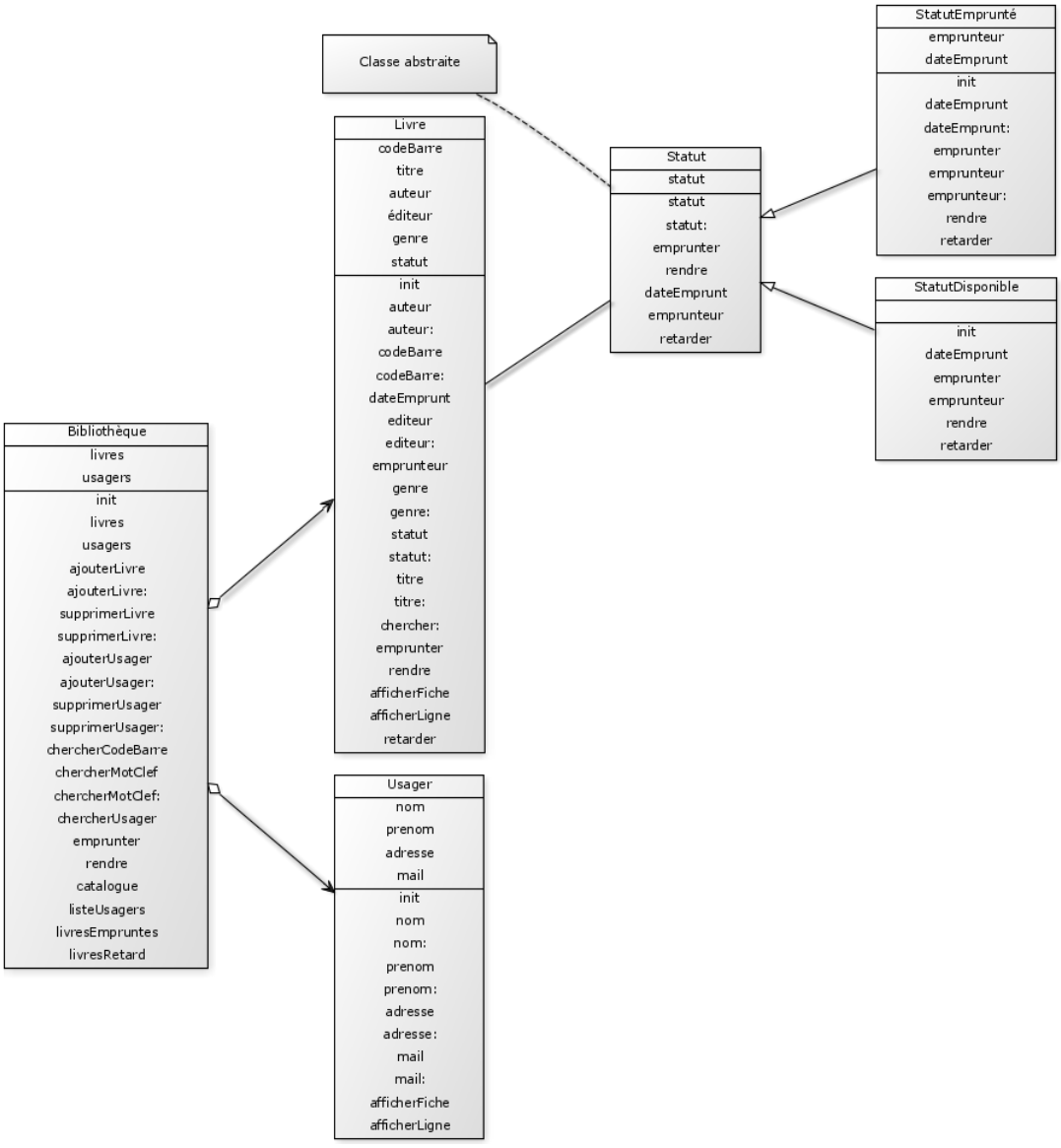

Fig. 1 – Modèle conceptuel

4 Tests

Le système de gestion de bibliothèque peut être testé par les commandes suivantes :

```
biblio := Bibliotheque new.

biblio ajouterLivre.
biblio catalogue.
biblio supprimerLivre.

biblio ajouterUsager.
biblio listeUsagers.
biblio supprimerUsager.

biblio emprunter.
biblio rendre.

biblio chercherCodeBarre.
biblio chercherMotClef.

biblio livresEmpruntes.

biblio chercherCodeBarre retarder.
biblio livresRetard.

biblio chercherUsager.
```

L'objet de ces instructions est détaillé dans le code de la classe Bibliothèque. Elle ont permis de valider l'ensemble des cas généraux et limites exposés dans la partie d'analyse du problème.

5 Conclusion

En somme, l'ensemble des fonctionnalités exposées en analyse du problème ont été implémentées avec succès. Le principal obstacle que j'ai rencontré était la compréhension du fonctionnement des classes du langage, notamment Date et Transcript. On peut apporter un bémol à la conception sur le fait que la durée maximale d'un prêt est indiquée dans la méthode qui permet d'afficher la liste des livres en retard au lieu d'être fixée en attribut de la bibliothèque.

Pour aller plus loin, il est possible d'ajouter au système de gestion de bibliothèque un service de réservation des livres, où un livre ne peut être emprunté que par un usager qui figure dans la liste des usagers qui ont effectué une demande de réservation sur ce livre. Pour cela, il suffirait de créer une sous-classe StatutRéservé de la classe Statut en mettant en attribut la liste des usagers demandeurs et en effectuant une vérification sur cette liste lors de la réception du message *emprunter*.