École de technologie supérieure
Département de génie logiciel et des TI
Baccalauréat en génie logiciel

Trimestre : Hiver 2012
Professeur : R. Champagne
Chargé de lab. : S. Djeffal

LOG430 - ARCHITECTURE LOGICIELLE

LABORATOIRE 2: LE STYLE ARCHITECTURAL EN COUCHES ET LES IMPLÉMENTATIONS ORIENTÉES OBJET

Description du problème

Le but de ce laboratoire est de mieux comprendre le mappage d'une implémentation orientée objet à une architecture. La première partie de ce laboratoire consiste à modifier une implémentation existante et fonctionnelle, afin de vous permettre de faire le lien entre les concepts architecturaux et l'implémentation. Ce cours n'est pas un cours de programmation et l'emphase du laboratoire est mise sur les concepts architecturaux. Le laboratoire est divisé en deux parties.

Pour la première partie, un système existant vous est fourni. Ce système fait partie d'un système plus large et consiste en un système d'attribution de cours à des enseignants utilisé par des directeurs de départements universitaires au début de chaque session. Votre tâche pour la première partie consiste à modifier le code source du système original afin de satisfaire des nouveaux besoins, lesquels sont énumérés plus bas.

La seconde partie du laboratoire consiste à analyser la structure de ce système. Après avoir analysé et modifié le système, vous devrez répondre à des questions liées aux décisions de conception que votre équipe a prises dans la première partie.

Nous vous recommandons fortement d'adopter la technique de développement incrémental, c'est-à-dire de faire une partie de l'implémentation tout en considérant soigneusement les aspects architecturaux. Ceci revient à dire que vous devriez itérer entre la première et la deuxième partie du laboratoire. À la limite, vous pouvez faire l'analyse avant l'implémentation.

Fonctionnalité du système existant

La fonction de base du système actuel consiste à attribuer des cours à des enseignants. Le système fournit une fonctionnalité rudimentaire et maintient plusieurs listes. Les deux listes principales sont:

1. une liste d'enseignants;
2. une liste de cours.

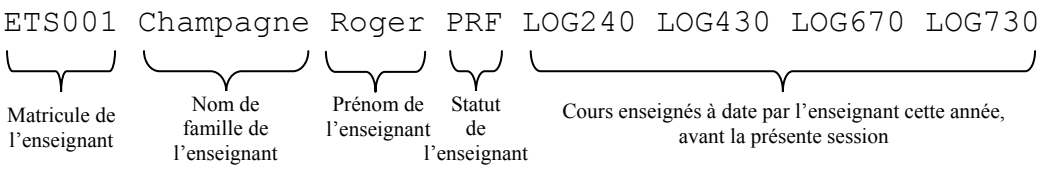
Dans ce système, un objet de la classe `Teacher` maintient deux listes internes:

1. une liste de cours déjà enseignés par l'enseignant cette année (avant la session courante);
2. une liste de cours attribués à l'enseignant pour la session courante.

Un objet de classe `Course` maintient quant à lui une liste interne, consistant en la liste des enseignants auxquels le cours est attribué. Un cours peut être enseigné par plus d'un enseignant.

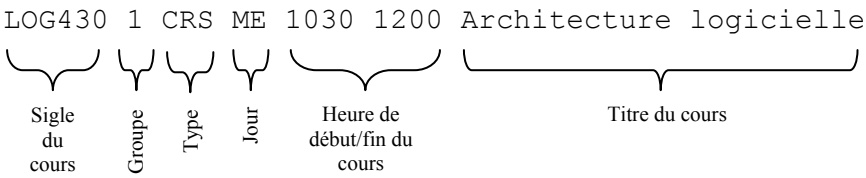
Deux fichiers sont fournis avec le système initial, soit un fichier contenant une liste d'enseignants et un autre fichier contenant une liste de cours. Le fichier contenant la liste d'enseignants a le format montré à la figure suivante. Le statut de l'enseignant est soit PRF (professeur) ou CHR (chargé de cours/laboratoires)

Figure 1 - Les divers champs dans le fichier d'entrée contenant les informations liées aux enseignants.



Le second fichier, contenant les informations liées aux cours offerts pour la session courante, contient divers champs liés aux cours. Un exemple d'une ligne de ce fichier est montré à la figure ici-bas. Le type est soit CRS (cours) ou LAB (laboratoire).

Figure 2 - Les divers champs dans le fichier d'entrée contenant les informations liées aux cours.



Le système original comporte une interface textuelle basée sur des menus. Le menu principal offre les options suivantes:

1. afficher la liste des enseignants;
2. afficher la liste des cours offerts cette session;
3. afficher la liste des cours attribués à un enseignant cette session;
4. afficher les enseignants auxquels un cours est attribué cette session;
5. attribuer un cours à un enseignant;

X. Quitter le système.

Option 1: Affiche la liste des enseignants. Les enseignants dans le système sont ceux inscrits dans le fichier des données d'entrée dont un exemple est fourni avec le système original (fichier **`enseignantsLOG.txt`**).

Option 2: Affiche la liste des cours offerts cette session. Les cours offerts sont ceux inscrits dans le fichier des données d'entrée dont un exemple est fourni avec le système original (fichier **`coursLOG.txt`**)

Option 3: Demande à l'usager de fournir le matricule de l'enseignant. Une fois entré, le système cherche l'enseignant associé à ce matricule et affiche la liste des cours attribués à cet enseignant pour la session en cours.

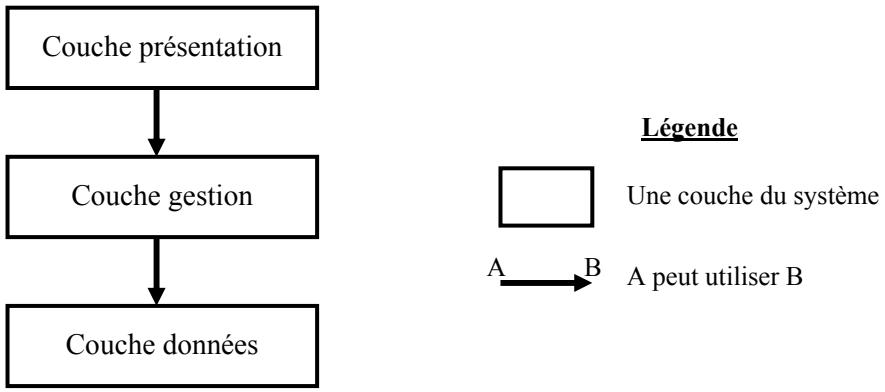
Option 4: Demande à l'usager de fournir un sigle de cours et un groupe. Une fois ces informations entrées, le système trouve le cours-groupe dans la liste des cours et affiche les enseignants auxquels est attribué le cours-groupe.

Option 5: Demande à l'usager de fournir le matricule d'un enseignant, un sigle de cours et un groupe. Une fois ces informations entrées, le système ajoute le cours à la liste des cours attribués à l'enseignant et ajoute l'enseignant à la liste des enseignants assignés à ce cours. Aucune vérification n'est faite quant aux multiples attributions de cours au même enseignant ou aux conflits d'horaire.

Option X: Quitte le système d'inscription.

Architecture du système existant

Le système existant est caractérisé par une architecture à trois couches, laquelle est implémentée (au niveau détaillé) en orienté objet. Les trois couches sont divisées comme suit:



Chaque couche contient un certain nombre d'objets qui fournissent la fonctionnalité pour cette couche. Les connexions inter-couches sont accomplies en utilisant la relation "peut utiliser". Le système original est implémenté en Java et contient les fichiers source suivants:

`TeacherAssignment.java`: Fichier principal (contient la méthode `main()`).

`Termio.java`: Contient divers services d'entrées/sorties pour la console.

`Menus.java`: Affiche les menus sur le terminal.

`Displays.java`: Affiche les diverses listes (cours et étudiants) sur le terminal.

`List.java`: Fournit les divers services et définitions liés aux listes. Les listes sont implantées dans ce système sous forme de vecteurs Java.

`CourseList.java`: Fournit une liste d'objets de classe `Course`.

`TeacherList.java`: Fournit une liste d'objets de classe `Teacher`.

`Teacher.java`: Objet représentant un enseignant dans ce système.

`Course.java`: Objet représentant un cours dans ce système.

`LineOfTextFileReader.java`: Fournit divers services d'entrées/sorties liés aux fichiers. Lit des lignes de texte à partir d'un fichier.

`CourseReader.java`: Décode ("parse") les lignes de texte du fichier des cours, créé un objet de classe `Course` pour chaque ligne du fichier et retourne une liste de cours.

`TeacherReader.java`: Décode ("parse") les lignes de texte du fichier des enseignants, créé un objet de classe `Teacher` pour chaque ligne du fichier et retourne une liste d'enseignants.

Compilation et exécution du système original

Vous devez d'abord désarchiver le contenu du fichier comprenant le code source et placer son contenu dans votre répertoire de travail. Pour compiler le programme, ouvrez une fenêtre DOS, déplacez-vous dans votre répertoire de travail et invoquez la commande suivante (cet exemple suppose que votre répertoire de travail est `C:\lab2`):

```
C:\lab2> javac TeacherAssignment.java
```

Une fois le programme compilé, vous pouvez l'exécuter et invoquant la commande suivante:

```
C:\lab2> java TeacherAssignment [nomDuFichierDesCours] [nomDuFichierDesEnseignants]
```

Note: Un fichier de cours par défaut vous est fourni et se nomme `coursLOG.txt`. Un fichier d'enseignants par défaut vous est également fourni et se nomme `enseignantsLOG.txt`.

En utilisant les fichiers par défaut qui vous sont fournis, vous pouvez donc utiliser le système original de la façon suivante:

```
C:\lab2> java TeacherAssignment coursLOG.txt enseignantsLOG.txt
```

Partie 1: Modifications au système original

Ce système est plus complexe que celui du premier laboratoire, vous êtes donc encouragés à commencer ce laboratoire dès que possible. Votre première tâche dans ce laboratoire consiste à ajouter de la fonctionnalité au système existant. Votre nouveau système doit inclure tous les ajouts énumérés ici-bas. Utilisez de bonnes pratiques de programmation, et ne vous gênez pas pour inclure des commentaires dans votre code. Chaque entête de fichier java contient un registre des modifications. Utilisez-le!

Modifications:

Modification 1: Ajouter une option qui permet aux usagers d'afficher la liste des cours (pas les laboratoires!) qu'un enseignant a déjà enseignés avant la présente session.

Modification 2: Ajouter une option qui permet d'afficher la liste des cours (pas les laboratoires!) qui n'ont pas été encore attribués cette session.

Modification 3: Modifier le système existant de telle sorte qu'un avertissement soit émis lorsqu'on essaie d'attribuer à un enseignant un cours ou laboratoire déjà attribué (pour le même groupe). Le système ne doit permettre la saisie dans cette situation, une fois que l'utilisateur a pris connaissance de l'avertissement.

Modification 4: Modifier le système de telle sorte qu'une vérification de conflits soit effectuée. Vérifiez spécifiquement pour les types de conflits suivants:

- conflit d'horaire – vous devez vérifier à chaque fois que le cours ou laboratoire qu'on assigne à un enseignant n'entre pas en conflit avec un autre cours ou laboratoire déjà assigné au même enseignant.
- limite de cours à attribuer à un enseignant – si l'enseignant a le statut de professeur (PRF), on ne peut lui assigner plus de trois (3) cours dans une même année. Si l'enseignant est un chargé de cours (CHR), on ne peut lui assigner plus de cinq (5) cours dans une même année. Notez que les laboratoires n'affectent pas cette règle, seulement les **cours**. Un prof peut donner trois cours plus des laboratoires, mais pas plus de trois **cours**. Un chargé peut donner cinq cours plus des laboratoires, mais pas plus de cinq **cours**.

Tests

En plus de ces diverses modifications, vous devez fournir un plan de test. Vous devez inclure une description des tests, tous les fichiers requis pour effectuer ces tests, et les procédures à utiliser pour tester votre système. Votre laboratoire ne sera pas corrigé si vous ne fournissez pas ces artéfacts. **Prenez note que votre laboratoire ne sera pas nécessairement évalué avec les données fournies. Faites attention de bien tester les nouvelles fonctionnalités**. Un exemple de plan de tests, sous forme de tableau, est disponible sur le site web du cours dans la section « Autres documents ».

Partie 2: Analyse architecturale

a) Vues architecturales:
   - Système original
     i. Produisez la matrice de dépendance (DSM) du système original avec l'outil Lattix LDM.
     ii. Produisez un diagramme de classes du système original.
     iii. En utilisant l'architecture du système original, effectuez un mappage des classes aux couches de l'architecture.
   - Système final (votre solution)
     i. Produisez la matrice de dépendance (DSM) du système original avec l'outil Lattix LDM.
     ii. Produisez une vue architecturale (couches) de votre nouveau système (celle-ci peut être différente ou non de la vue originale).
     iii. Produisez un diagramme de classes de votre nouveau système.
     iv. Effectuez un mappage des classes du nouveau système sur votre nouvelle vue architecturale.

b) Expliquez les critères adoptés pour effectuer le mappage des classes aux couches de l'architecture.

c) Comment interpréter la matrice de dépendance et comment peut-on l'utiliser pour améliorer notre architecture?

d) En utilisant votre nouveau système comme base de discussion, comment modifieriez-vous le système afin d'accéder à une base de données au lieu d'utiliser des fichiers texte? Produisez une vue en couches avec les classes "conceptuelles" d'une telle solution pour illustrer votre propos (n'oubliez pas de commenter la/les vues).

Critères d'évaluation

Votre solution ainsi que votre analyse seront corrigées selon les critères suivants:

- bon fonctionnement de votre implémentation et satisfaction des exigences fonctionnelles;
- la qualité de vos programmes (commentaires, bonne structure);
- la qualité et le contenu de votre analyse, démontrant votre compréhension des concepts architecturaux.

Plus spécifiquement, les points seront attribués comme suit:

Partie I - Implémentation

- Nouveau système (incluant les tests): 50 %.

Partie II - Rapport

- Question a): 20 %
- Question b): 10 %
- Question c): 10 %
- Question d): 10 %