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

Trimestre : Automne 2013
Professeur : R. Champagne
Chargés de lab. : F. El-Kassis, S. Djeffal

LOG430 - ARCHITECTURE LOGICIELLE

LABORATOIRE 1: 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. Dosez vos efforts!

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 élémentaire de gestion de projet. Il sert à assigner des ressources (humaines) à des projets de développement logiciel. 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 assigner des individus à des projets. Le système fournit une fonctionnalité rudimentaire et maintient plusieurs listes. Les deux listes principales sont:

1. une liste de ressources (les individus);
2. une liste de projets.

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

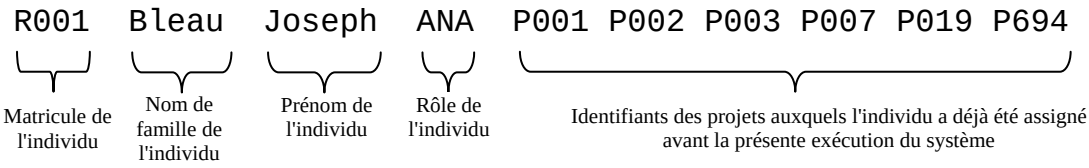
1. une liste des projets auxquels la ressource a déjà été assignée **avant** la présente exécution du système;
2. une liste des projets assignés à la ressource **durant** l'exécution courante du système.

Un objet de classe `Project` maintient quant à lui une liste interne, consistant en une liste des ressources (individus) assignées à ce projet.

Deux fichiers sont fournis avec le système initial, soit un fichier contenant une liste d'individus et un autre fichier contenant une liste de projets. Le fichier contenant la liste des individus a le format montré à la

figure suivante. Le rôle d'un individu est soit ANA (analyste), DES (concepteur), PRG (programmeur) ou TST (testeur).

Figure 1 - Les divers champs dans le fichier d'entrée contenant les informations liées aux ressources (individus).



Le second fichier, contenant les informations liées aux projets passés, en cours et à venir, est constitué de divers champs liés à la gestion de ces projets. La description des champs de ce fichier est montrée dans le tableau ici-bas.

Champ	Taille	Exemples de valeurs
Identifiant	P111	P999
Nom	Chaîne de caractères (sans espace)	Projet_1
Date de début de projet	9999-99-99 (AAAA-MM-JJ)	2013-12-21 (12 décembre 2013)
Date de fin de projet	9999-99-99 (AAAA-MM-JJ)	2013-12-21 (12 décembre 2013)
Priorité du projet	L'un de {H, M, L}	H

Tableau 1 : Description des champs du fichier des projets

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

1. Afficher la liste des ressources (individus);
2. Afficher la liste des projets;
3. Afficher la liste des projets assignés à une ressource durant l'exécution courante;
4. Afficher la liste des ressources affectées à un projet durant l'exécution courante;
5. Affecter une ressource à un projet;

X. Quitter le système.

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

Option 2: Affiche la liste des projets. Les projets à considérer sont ceux inscrits dans le fichier des données d'entrée dont un exemple est fourni avec le système original (fichier **`projects.txt`**)

Option 3: Demande à l'usager de fournir le matricule d'une ressource. Une fois entré, le système cherche l'individu associé à ce matricule et affiche la liste des projets assignés à cet individu durant l'exécution courante du système.

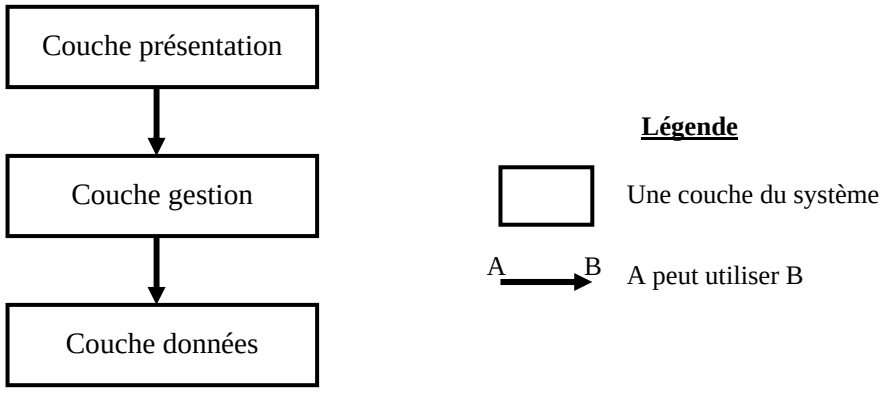
Option 4: Demande à l'usager de fournir l'identifiant d'un projet. Une fois cette information entrée, le système trouve le projet et affiche la liste des ressources assignées à ce projet durant l'exécution courante du système.

Option 5: Demande à l'usager de fournir le matricule d'une ressource et l'identifiant d'un projet. Une fois ces informations entrées, le système ajoute le projet à la liste des projets assignés à cette ressource et ajoute la ressource à la liste des ressources assignées à ce projet. Aucune vérification n'est faite quant aux multiples attributions de projets ou ressources 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 sources suivants:

`ResourceAssignment.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 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.

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

`Project.java`: Objet représentant un projet dans ce système.

`Resource.java`: Objet représentant une ressource dans ce système.

`ProjectList.java`: Fournit une liste d'objets de classe `Project`.

`ResourceList.java`: Fournit une liste d'objets de classe `Resource`.

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

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

Compilation et exécution du système original

Vous pouvez travailler dans un IDE (environnement de développement intégré), mais les instructions suivantes supposent un outillage minimal (une ligne de commande DOS sous Windows).

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:\lab1`):

```
C:\lab1> javac ResourceAssignment.java
```

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

```
C:\lab1> java ResourceAssignment [nomFichierProjets] [nomFichierRessources]
```

Note: Un fichier de projets par défaut vous est fourni et se nomme `projects.txt`. Un fichier de ressources par défaut vous est également fourni et se nomme `resources.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 ResourceAssignment projects.txt resources.txt
```

Partie 1: Modifications au système original

Votre première tâche 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!

Modification 1: Ajoutez une option qui permet d'afficher la liste des projets auxquels une ressource était déjà affectée avant l'exécution courante du système.

Modification 2: Ajoutez une option qui permet d'afficher tous les rôles ayant été assignés à un projet spécifique, incluant les rôles assignés avant l'exécution courante **et** les rôles assignés durant l'exécution courante.

Modification 3: Modifiez le système existant de telle sorte qu'une erreur soit signalée lorsqu'une ressource est surchargée. Pour cette modification, vous devez tenir compte de la priorité, de la date de début et de la date de fin de projet de tous les projets (ceux déjà assignés avant **et** durant l'exécution courante, et celui qu'on est en train d'essayer d'assigner).

- Un projet avec une priorité "H" (élevée) occupe une ressource à 100%.
- Un projet avec une priorité "M" (moyenne) occupe une ressource à 50%.
- Un projet avec une priorité "L" (basse) occupe une ressource à 25%.

Une ressource ne peut pas être occupée plus de 100% du temps**,** ne serait-ce qu'une journée. Un message d'erreur significatif doit donc être affiché si on essaie d'affecter un projet à une ressource et que son emploi du temps dépasse 100% durant au moins une journée.

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. À la correction interactive de votre laboratoire, vous aurez à exécuter votre plan de tests devant les chargés de laboratoire. Assurez-vous que votre plan de test est réalisable en peu de temps.

Aussi, il est possible que votre laboratoire soit à nouveau testé par les chargés de laboratoire après la correction interactive (pas en votre présence). **Prenez note que pour ces tests, votre laboratoire ne sera pas nécessairement évalué avec vos données. Faites attention de bien tester les nouvelles fonctionnalités**.

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 final 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 %