|  |  |
| --- | --- |
|  | Trimestre : Automne 2016  Enseignants : R. Champagne  B. Galarneau  Y.Ross  Chargé de lab. : A. Grenier |

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. Il s'agit d'un guichet automatique bancaire (GAB). Votre tâche pour la première partie consiste à étudier l'architecture actuelle de ce système, et à la revoir afin d'en améliorer certains attributs de qualités.

La seconde partie du laboratoire consiste à analyser la structure de ce système. Après avoir 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.

# Fonctionnalité du système existant

Le système est entièrement documenté au <http://www.math-cs.gordon.edu/courses/cs211/ATMExample/>. Ce système est utilisé comme exemple dans un cours de conception orientée objet. Les artéfacts du site d'origine n'ont pas tous la même utilité pour votre travail ici.

Bien que le code source du système soit disponible sur ce même site, le projet a été réorganisé en projet Maven, ce qui vous facilite la tâche pour le présent travail. Pour le code source, utilisez l'archive de départ qui vous est fournie sur le partage Google Drive de votre groupe.

Le système tel qu'il existe en ce moment contient en fait deux parties distinctes:

* le logiciel destiné au guichet automatique lui-même;
* un simulateur de guichet automatique, dont le but est de ne pas avoir à posséder l'équipement physique pour réaliser les tests sur le logiciel principal.

# Architecture du système

Le système existant n'utilise pas de style architectural spécifique. Votre travail consiste à réorganiser la structure du code existant pour utiliser une architecture en couches, afin d'améliorer la séparation des préoccupations, la portabilité et la testabilité.

# 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).

Pour compiler le projet, consultez le fichier **LISEZMOI.html** à la racine de l'archive fournie.

Pour utiliser le système, consultez la documentation sur le site d'origine, dans la section "Executable Applet". Bien que ces directives soient destinées à une version "applet" de ce système (qui a été retirée du code source fourni), les mêmes directives s'appliquent pour la version "standard" qui vous est fournie.

# Partie 1: réorganisation du système original

Du point de vue implémentation, votre tâche consiste à revoir l'architecture de ce système. Il n'y a pas de fonctionnalité à ajouter ou modifier (à moins que vous trouviez des bogues que vous décidez de corriger). Fonctionnellement, les attentes sont que votre système final soit équivalent au système initialement fourni.

Vous devez réorganiser le code en utilisant le style architectural "en couches". Votre réorganisation doit permettre d'améliorer:

1. la séparation des préoccupations: entre autres, le code actuel ne fait pas de distinction nette entre le simulateur et l'application destinée au guichet automatique lui-même. De même, la distinction entre l'aspect interface utilisateur, logique d'affaires et accès aux ressources n'est pas nette.
2. la portabilité du code lié au guichet automatique lui-même (pas le simulateur): ce point découle du précédent. Comme les préoccupations sont mal séparées, le code est peu portable (par exemple vers différentes sortes de matériel).
3. la testabilité: ce point découle aussi d'une séparation des préoccupations discutable. Il est actuellement difficile de tester, de façon automatisée, la logique applicative destinée à un éventuel GAB physique

Notez que vous devrez décider du nombre de couches, du niveau de "rigidité" de votre architecture (pontage permis ou pas, utilisation de nacelles latérales ou pas, …), et que toutes vos décisions architecturales devront être justifiées dans les réponses aux questions de la seconde partie.

## Tests

En plus de ces diverses modifications, vous devez ajouter au projet des tests automatisés pour la logique applicative destinée au GAB. La seule prescription pour cette partie est que vos tests doivent s'exécuter automatiquement à la ligne de commande via Maven. Vous pouvez utiliser la librairie de tests qui vous convient (Junit, TestNG, …). Votre livraison devrait donc inclure un script **pom.xml** qui permette de lancer la commande **mvn test**. Vos tests devraient évidemment tous passer, tout en testant la fonctionnalité de manière significative.

Les fonctions à tester de façon automatisée sont les quatre services décrits dans la section "Requirements" de la documentation web du projet initial.

# Partie 2: Analyse architecturale

1. Fournissez la matrice de dépendance générée (matrice principale des paquetages et matrices "internes" des paquetages intéressants) par **dsm-maven-plugin**. Commentez cette/ces vue(s) architecturale(s) en termes des attributs de qualité pertinents pour ce laboratoire. Indiquez entre autres les endroits où il y a des dépendances nuisibles évitables. Au besoin, complémentez votre analyse avec le plugin STAN installé dans la distribution d'eclipse pour ce cours.
2. Produisez une vue architecturale de votre solution finale, en incluant les couches et les classes. Utilisez la notation de votre choix. Si vous avez changé des classes, vous devez expliquer les changements faits par rapport aux classes originales. Expliquez comment vos modifications architecturales ont amélioré la situation par rapport aux attributs de qualité pertinents pour ce laboratoire. Justifiez toutes vos décisions architecturales.
3. Discutez la valeur ajoutée des considérations architecturales par rapport à la conception orientée objet originale seulement.

# Critères d'évaluation

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

* bon fonctionnement de votre implémentation (on vous a fourni une application fonctionnelle, elle doit encore fonctionner à la fin du lab);
* la qualité de vos programmes (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): 20 %
* Question c): 10 %