Projet M1 Java - Consignes

M1 Informatique - Algorithmie et Programmation Avancée

Année Universitaire 2019 - 2020

1 Présentation

Ce projet permet à l'étudiant de réaliser un logiciel en passant par toutes les étapes de son cycle de vie : spécifications, analyse et conception, codage, vérification et maintenance. Cette année, un seul sujet commun à tous les étudiants réalisé en groupe est proposé, mais il se décline ensuite en trois sous-projets spécifiques. Il présente une problématique générale et une approche pour résoudre de façon pratique cette problématique. Vous devrez héberger le logiciel sous la forme d'un dépôt Github.

2 Date limite et rendu

Le rapport et le logiciel (le lien du dépôt Github) doivent être rendus tous les deux avant le **9 Janvier 2020**. Le détail du rapport est précisé ci-dessous.

A chaque fois, les documents électroniques (format `.pdf` imposé) doivent être envoyés par email à `antoine.gourru@univ-lyon2.fr` avec `julien.velcin@univ-lyon2.fr` en copie. N'oubliez pas de bien préciser dans votre mail le nom de **chacun** des participants. L'email doit avoir comme sujet le mot `projetJava` suivi du nom des étudiants composant le groupe.

3 Documents à fournir :

Lorsque ce projet sera terminé, vous aurez à fournir :

1. Un rapport (format .pdf uniquement, d'une taille inférieure à 2Mo). Celui-ci rappelle la problématique du projet et en donne les spécifications. Il détaille ensuite l'analyse effectuée par les étudiants et comment celle-ci est implémentée (conception). Un soin particulier doit être pris pour expliquer comment les diverses tâches ont été réparties entre les différents étudiants travaillant sur le projet. En effet, au moment de l'évaluation, l'examinateur se réserve le droit d'interroger chaque étudiant individuellement sur le projet et de prendre en compte ses réponses pour attribuer une note individuelle. Il est très important également de rappeler que ce rapport **ne doit pas** comporter les lignes de codes du logiciel. Enfin, le rapport doit présenter les tests effectués, une conclusion et des perspectives.

   Les différents éléments du rapport sont détaillés dans la section 4.

2. Le lien du dépot Github contenant le code source. Il comprendra l 'ensemble des fichiers (comprenant obligatoirement le code source `.java`), provenant d'un projet Eclipse ou d'un projet NetBeans utilisant la version 1.8 de Java. Le code doit être correctement présenté (indentation), clair (utilisation de norme précise pour les noms) et commenté (intelligemment). Les librairies éventuelles doivent également être fournies au format `.jar`. Le bon fonctionnement du programme sera jugé sur **une version recompilée** des fichiers sources avec le compilateur et la machine virtuelle de Java 1.8.

4 Le rapport :

Le rapport doit comporter :

1. Les spécifications : partie normalement rédigée conjointement avec le client qui comprend une description en langage naturel de ce que doit faire le programme. Dans notre cas, les spécifications seront rédigées à partir des énoncés et des questions posées à l'enseignant.

2. L'analyse : cette partie explique comment les spécifications vont être réalisées. Elle comporte :

   - quel sera l'environnement de travail (vous pouvez justifier le choix de votre environnement) ?
   - quelles sont les données identifiées dans les spécifications ?
   - un diagramme des classes (simplifié bien sûr), c'est-à-dire :
     - préciser les classes utilisées et si elles sont abstraites ou non,
     - préciser les relations d'héritage entre classes,
     - préciser les relations d'utilisation entre les classes,
     - préciser les interfaces implémentées sur chaque classe,
     - préciser les méthodes `public` de ces objets.
   - spécifier, par classe, les méthodes `public` identifiées à l'étape précédente : ce qu'elles doivent réaliser (en langage naturel).

3. La conception : cette partie explique comment l'analyse est implémentée, c'est-à-dire :

   - préciser clairement la manière dont vous les tâches ont été partagées entre les différents étudiants travaillant sur le même projet,
   - détailler la manière avec laquelle vous avez implémenté les relations d'utilisations (cf. note 1 ci-dessous),
   - détailler les algorithmes spécifiques à l'application et d'une certaine complexité (si nécessaire),
   - détailler les problèmes que vous avez rencontrés, comment vous les avez résolus, contournés ou non.
   - donnez un exemple commenté d'utilisation du logiciel du début à la fin.

4. La validation : spécifier les procédures de test de votre logiciel, c'est-à-dire :

   - les tests unitaires : comment les méthodes ont été testées individuellement ? Donnez quelques résultats ;
   - les test globaux : en utilisant votre interface, quels cas particuliers avez-vous testés ?

5. La maintenance : donnez quelques évolutions possibles du logiciel. Quelles sont les fonctionnalités que l'on peut rajouter et comment ? Pourquoi cela sera-t-il facile ou difficile (cf. note 2 ci-dessous) ?

4.1 Remarques :

1. Exemple :

   Dans les TDs, nous avions une relation d'utilisation entre la classe BaseDeNews et la classe MesNews, ainsi qu'entre la classe News et la classe BaseDeNews. Dans BaseDeNews, la collection des actualités a été stockée dans un objet de type TreeSet. Nous pouvons justifier ce choix : les actualités devaient être stockés sans redondance et leur ordre était important (classement par ordre alphabétique des titres, par exemple).

2. Exemple :

   Les TDs prévoyaient la sauvegarde et le chargement de la base depuis un fichier stocké sur le disque dur. Normalement, ce fichier doit être accédé relativement à l'arborescence du projet afin de ne pas poser de problème suivant le système d'exploitation utilisé. Cependant, vous pouvez décider d'employer une base de données relationnelle (ou tout autre stockage, y compris sur Internet) mais il faudra justifier votre choix et, surtout, vous assurer que l'exécution de l'application ne posera aucun problème avec la machine employée. Ainsi, il faut éviter au maximum de demander l'installation de trop nombreuses librairies ou de logiciels.

5 Notation du projet :

La notation du projet tiendra compte des éléments suivant :

1. Fonctionnement général du logiciel :

   - état d'avancement du projet : jusqu'où ont été réalisées les spécifications sans bugs (étant entendu par bugs tout plantage inopiné du programme).
   - amélioration du projet : quels sont les plus apportés aux spécifications (pris en compte seulement si les spécifications ont d'abord été réalisées).
   - les codes non valides (dans le cas où le programme ne marche pas ou mal), s'ils sont identifiés et commentés comme tels.

2. Code (dans l'ordre d'importance) :

   - Présentation (pas d'indentation, pas de lecture),
   - Commentaire : en lisant les commentaires, on doit pouvoir comprendre le programme,
   - Clarté : pensez à utiliser une norme pour les noms de fichiers.

3. Dossier :

   - Analyse : qualité du choix des classes et des interfaces (au sens Java), prise en compte des mécanismes objets, qualité des algorithmes du programme (mis en avant ici).
   - Conception : correspondance avec la réalité, qualité de l'interface.
   - Présence de chaque partie avec la mise en avant des informations pertinentes (qualité générale)
   - Parties supplémentaires (tests et maintenance) : dans le cas où toutes les autres parties ont déjà été réalisées.

6 Notes complémentaires :

- L'environnement de développement n'est pas imposé, mais il doit permettre un import dans **NetBeans** ou **Eclipse** car c'est sur ce dernier IDE que le projet sera testé.
- Il faut respecter le caractère multi-plateforme de Java. Aussi, veillez à ne pas utiliser d'éléments spécifiques à un certain OS (ex. : ne pas utiliser d'adresse absolue comme "C:\...\...").
- Pensez à fournir les librairies nécessaires à l'installation de votre logiciel.
- Un programme qui fonctionne mais qui fait peu de choses sera mieux noté qu'un programme qui ne fonctionne pas du tout (par définition qui ne fait rien). Autrement dit : l'étendue des fonctionnalités réalisables par l'interface (et qui fonctionne) est un critère important.

- Plus les mécanismes objets et les particularités de Java (ex. : l'héritage, le polymorphisme, les collections, les flux, etc.) seront utilisés, plus la note sera élevée (et inversement).
- L'utilisation du code d'autres auteurs (et libre de droit) est **absolument interdite**. Une telle utilisation peut annuler tout le travail des étudiants (autrement dit amener une note de 0/20).

Bonne chance !