Université du Québec en Outaouais Automne 2023 **21 sept., 2023**

Département d'informatique et d'ingénierie

INF1163 : Modélisation et conception orientée objet

Devoir et projet

Devoir à soumettre au plus tard le 7 octobre à 23 h 55

Projet à soumettre au plus tard le 12 décembre à 23 h 55

C'est un travail en équipe (3 membres par équipe pour le projet mais devoir individuel). Le but de ce projet est d'appliquer les principes de l'analyse et de la conception orientées objet vus en classe, notamment l'approche unifiée et UML. L'implémentation se fera en Java (Proposé : l'éditeur de code VS Code). Le projet vous donne aussi l'occasion d'expérimenter avec quelques outils logiciels performants utilisés dans un environnement de développement professionnel (Git, GitHub). La collaboration entre les membres de chaque équipe est indispensable afin de pouvoir respecter les délais impartis. Les étudiants doivent donc apprendre à travailler en équipe. Ils doivent notamment apprendre à se répartir les activités équitablement, à développer leur sens de la communication et à synchroniser leur travail afin d'être productifs dans leurs efforts. Il faudra donc nommer un chef d'équipe pour coordonner le travail et veiller à la bonne progression du projet. Le rôle de chef d'équipe pourra être joué, à tour de rôle et pour une période déterminée à l'avance, par chacun des membres de l'équipe. Afin de gérer efficacement le développement de votre code source et de pouvoir rapidement revenir à une version qui fonctionne en cas de problème, il est fortement conseillé d'utiliser l'outil de gestion de version Git (http://git-scm.com/). Afin de permettre à plusieurs membres de l'équipe de développer en parallèle et d'intégrer automatiquement leurs différentes portions de code, il est demandé d'utiliser l'outil de gestion de contrôle GitHub (https://github.com/). GitHub est en fait la partie serveur pour l'outil Git. Git est gratuit et peut être installé sur votre PC. GitHub est un serveur sur le nuage qui requiert que vous ouvriez un compte (gratuit). Il faut fournir un accès lecture dans le rapport faisant partie des livrable de votre projet. La correction de votre projet prend en considération les traces des « commit » et « issues » , « Pull requests » , « tags » qui témoignent d'un vrai travail de collaboration sur votre repos sur Github y compris l'historique de production du code (pas fait hâtivement la veille de la soumission) qui doit montrer les début de production significative à partir du 08 juillet.

La version de java qui sera utilisée pour la compilation et l'exécution de votre projet est **Java 15**.

Une compagnie de développement de logiciel X veut concevoir un logiciel qui offre une assistance automatisée de calcul de temps, de salaires, et de contrôle de budget de ses projets. Ce système nommé TimeLog doit être installé sur une machine dédiée sur laquelle tout employé peut se connecter pour signaler qu'il a commencé une activité en la caractérisant par nom du projet, discipline de travail (design, implémentation, etc.). Un employé doit se reconnecter pour signaler son arrêt de travail sur une activité. Une seule activité est permise à la fois par employé. L'heure et date de ce signalement de début ou d'arrêt d'une activité sont enregistrées par le système.

Les disciplines d'activités adoptés pour initialiser TimeLog sont : design1 (haut niveau), design2 (détaillé), implémentation, test, et déploiement. Cet ensemble de disciplines est inspiré du processus unifié de dév. de logiciel qu'on va voir en classe.

Un projet doit comprendre : une date de début, date de fin, nbre d'heures budgétées pour chacune des disciplines, un nom et un numéro d'identification.

Un employé doit être caractérisé par un historique de ses taux horaires de base ; et un autre taux pour temps supplémentaire; un numéro identifiant ID; un nom; une date d'embauche, et possible date de départ, un numéro d'assurance social, poste.

Le système doit fournir les services suivants :

Le but de ce système est de permettre de fournir les services suivants moyennant une interface en ligne de commande (Interface graphique pas demandée) qui doit être interactive permettant à l'utilisateur de choisir des éléments de menu et de fournir des valeurs à partir de listes d'options (comparables aux listes déroulantes des interfaces graphique):

1. Un rapport d'état de chaque projet choisi. L'état d'un projet indique le nombre d'heures travaillées pour chacune de disciplines, et le pourcentage estimée d'avancement total et par discipline, basés sur le nombre d'heures travaillées en chaque discipline et le nombre d'heures budgétées.
2. Idem avec un rapport d'état global de l'ensemble des projets.
3. Pour chaque employé TimeLog peut fournir un rapport des valeurs (en salaire) des heures travaillées de l'employé depuis une date et heure quelconque; avec par défaut depuis le début de la dernière semaine impaire. Ce rapport donne le salaire brut pour la période choisie, pour chacun des projets sur lesquels l'employé a travaillé durant la période choisie.
4. Pour chaque employé on peut avoir un talon de paie pour la dernière période de paie ou pour une date de paie à spécifier (On paye toutes les deux semaines). Le talon de paie indique le salaire brut et le salaire net comme étant 0.6 du salaire brute (en raison de simplification dans le cadre de ce projet).
5. Idem avec les totaux des salaires bruts et nets de l'ensemble des employés. Ce rapport fournit donc seulement deux valeurs; total des salaires brutes et total des salaires nets.
6. Modifier les paramètres et certaines données de TimeLog par un admin : NPE expliqué plus loin, la liste des projets, la liste des employés, et assigner des employés à des projets.

7. La liste des disciplines de travail est par défaut : design1, design2, implémentation, test et déploiement. Néanmoins ça sera un plus si lors du démarrage du logiciel on pourra spécifier une liste de disciplines de notre choix.
8. Le système doit persister les informations sur des fichiers texte au format Json.
9. Avec le compte admin, on peut assigner des employés aux projets. Un employé ne peut travaillez que pour les projets auxquels il est assigné. Une contrainte à implémenter est de ne pas assigner un employé à plus d'un nombre NPE de projets. NPE est par défaut égale à 2, mais l'admin peut modifier la valeur de NPE.
10. L'administrateur du système se connecte avec le nom d'usager admin et un mot de passe admin qu'il peut changer plus tard.
11. Le compte admin peut modifier les noms d'usager et le ID de tout employé ainsi que les siens.
12. Le compte admin, permet de modifier la liste des projets et leurs caractéristiques, la liste des employés, leurs caractéristiques et leurs assignations aux différents projets.

13. Un employé se connecte au système en fournissant son nom d'usager et son ID.

14. Avec un compte employé (non admin) on peut se connecter pour signaler les débuts et les terminaisons de ses heures de ses activités;
15. Un employé peut aussi se connecter pour demander le nombre d'heures travaillées de base et supplémentaires pour une période spécifiée.

Toutes les sorties (outputs) doivent être textuelles au format Json bien indenté avec fonctionnalité offerte par une librairie Java pour Json, et ce dans le but de lisibilité de vos sorties.

Par exemple, un employé qui travaille le matin 2 h pour projet1, et qui ensuite continue sa journée en travaillant les 5.5 heures restantes sur un projet2, doit se connecter à plusieurs reprises : le matin; 2 heures après pour terminer une activité et commencer une autre; et en terminant sa journée de travail.

Interfaçage avec un futur sous système

TimeLog doit pouvoir **s'interfacer** avec un futur sous système de production et impression de chèques de paie aux deux semaines en lui envoyant en paramètre une liste d'objets de type **PayInfo**. Ce futur sous système est nommé **Payroll** et il implémente une interface **PayrollInterface** qui comprend une opération qui imprime les chèques et les talons de paies, cette opération est nommée printPay (pas impression mais affichage), et comme mentionné plus haut, elle accepte un seul paramètre sous forme d'une liste d'objets de type **PayInfo**.

a. Un objet PayInfo comprend les données suivantes pour une période de paie : identifiant d'un employé, nombre d'heures de base, nbre d'heures supplémentaires, taux horaire de base, taux horaire supplémentaires, les dates indiquant la période de paie considérée.

b. PayrollInteface comprend aussi les opérations : netFromBrute, DeductionsReport
c. Vous devez inclure dans vos modèles du projet l'interface **PayrollInterface** et le type **Payroll** mentionnées plus haut.
d. Votre projet doit permettre d'appeler correctement, avec les bonnes infos de paies, le sous système de paies mentionner plus haut. Donc, une partie du menu en ligne de command doit permettre de stimuler correctement la fonctionnalité de production de talons de paies et des chèques sans besoin de l'accomplir, ceci veut dire qu'il est demandé de créer un objet qui représente le système Payroll et qui peut peut répondre tout simplement à une invocation demandant l'impresssion de talon de paie par un simple message texte fournissant la date et l'heure de votre demande et préférablement avec un affichage de la liste des objets reçus de type PayInfo reçus en paramètre. La représentation textuelle pour ces objets est au format Json avec une indentation permettant la lisibilité.

Initialisation de votre projet

Votre implémentation doit comprendre 3 projets nommés projet1, projet2, projet3 avec leurs budgets pour toutes les disciplines + 3 employés avec noms d'usagers et IDs : employé1, ID=1; employé2, ID=2; employé3, ID=3; + un compte admin nommé admin avec nom usager admin et ID=0. D'autres valeurs des noms et ID doivent être possible, mais une initialisation est demandée avec les exemples simples mentionnés pour des raisons de test et d'uniformité des projets des étudiants.

Chaque employé est assigné à deux projets parmi les 3 projets, avec les durées de travail chaque jour, et ce durant les deux dernières semaines, comme suit pour chaque jour :

employé1 ayant travaillé 1.1 heure pour chaque discipline de deux projet projet1 et projet2;
employé2 ayant travaillé 1.2 heures pour chaque discipline de deux projet projet2 et projet3;
employé3 ayant travaillé 1. 3 heure pour chaque discipline de deux projet projet3 et projet1.

Travail demandé pour le devoir au plus tard le 7 octobre à 23 h 55

Pour la soumission du devoir, on demande un seul fichier zip nommé devoir-nomsDefamilleDeÉtudian.zip qui comprend un fichier pdf et un fichier lien comme suit:

- Un ou plusieurs diagrammes de cas d'utilisation qui comprennent tous les cas d'utilisation de TimeLog, à inclure dans un fichier pdf.

- Une documentation détaillée du cas d'utilisation « Signaler fin d'activité » relié au point numéro 14 plus haut. Le cas d'utilisation doit montrer l'aspect interactif, et on rappelle que l'interface de TimeLog est en ligne de commande qui permet, comme une interface graphique, une séquence d'interactions guidant l'utilisateur. Cette documentation doit être inclue dans le même fichier pdf mentionné dans le point précédent. (style essentiel)
- Le **modèle du domaine**
- Un fichier de type lien pointant le projet de l équipe sur Github, nommé github-nomDeFamillesÉtudiant. Votre repos sur Github peut contenir, entre autres, vos fichiers de modèles de Modelio, votre pdf du devoir, ou être vide à la date de soumission de votre devoir.

Soumission du projet à soumettre au plus tard le 12 décembre à 23 h 55

Les Livrables avec barème indicatif sur 20 points

La soumission du projet doit comprendre :

1. (6 pts= 4 *1.5) Les artéfacts UML finaux qui justifient la solution proposée et qui sont: Le **modèle du domaine,** le **DSS du cas** d'utilisation « Signaler fin d'activité », **un diagramme d'interaction d'une opération du DSS** de ce cas d'utilisation, et le **diagramme de classes** (comprenant les attributs et les opérations des classes).

Il faut tirer le maximum d'expressivité possible des modèles en UML pour une description du domaine d'application et la solution à envisager, pour des modèles qui seront la plus complets que possible. Tous les modèles doivent être créés avec Modélio, même s'ils sont à fournir comme figures dans le fichier pdf mentionné plus loin.

2. (absence=> fait perdre des points) Un fichier lien vers votre projet sur Github.

3. (absence=> fait perdre des points) Un bref texte explicatif résumant vos choix de conception et expliquant tous les artéfacts produits.

4. (7.5 points : le fichier Jar du projet exécutable, les **données**, et la **lisibilité** du code et son **correspondance** aux autres artéfacts. Les 7.5 points sont distribués en 0.5 pour chacune des 15 fonctionnalités énumérées plus haut) Le code source en Java, adéquatement **documenté** et un fichier de distribution de votre application au format jar. Le livrable doit comprendre un minimum de données d'initialisation déjà indiqué dans cet énoncé

5. (1 pts, mais pourrait avoir un impact négatif sur les autres parties du projet si la procédure n'est pas claire ou n'est pas correcte) Dans le document pdf mentionné plus loin, il faut fournir toutes les **explications nécessaires** pour permettre à une tierce personne de compiler, exécuter et utiliser votre projet, et ce en se limitant à l'utilisation des commandes de l'invite de commande, sans besoin d'utiliser un quelconque outil de développement intégré. Si vous avez

utilisé des librairies non standards, il faut inclure la procédure de leur installation, et donner leur URL et toute autre explication nécessaire à leur intégration dans votre code.

6. (note négative s'il manque) Une annexe très brève qui synthétise la contribution de chaque membre de l'équipe au projet. Par exemple : « Jean a collaboré à l'écriture des cas d'utilisation 2 et 4, a développé entièrement le diagramme de séquence 9 et a codé 20 % de l'application (ou a codé 2 des 3 interfaces utilisateur), Sam a créé les procédures de cas de test et les script et code associé ainsi que représentation des données au format json, etc.», etc.

7. (4 points) Un ensemble de documents de **cas de test** et des scripts et/ou JUnit tests associés qui permet de fournir une procédure complète pour tester chacune ou des groupes de **fonctionnalités** numérotées de 1 à 15. On doit numéroter ces documents et fichiers selon la numérotation fournie des fonctionnalités dans l'énoncé de ce projet ci-haut. Pour chaque cas de test une brève procédure complète et concise est demandée fournissant les données à utiliser et selon quelle séquence, en plus d'un script et/ou du code java pour automatiser le test. Un test peut tester **plusieurs fonctionnalités**, par exemple un test peut être nommé test_13_14.java pour tester une connexion et un signalement de début et fin d'une activité. Ainsi, **le nombre de tests est surement inférieur à 15**.

8. (1.5 points) pour la bonne prise en considération de « interfaçage avec le future sous système » dans vos modèles et votre implémentation. Il faut pointer l'illustration de ceci clairement dans votre réponse, créez une section intitulée « Illustration de l'interfaçage avec le futur système.

À l'exception du code source et du fichier de distribution **jar**, et du **fichier lien** vers votre repos github, tous les autres éléments du livrable doivent être incorporés **dans un fichier unique au format PDF**. Votre texte doit être tapé en utilisant un traitement de texte et vérifié en utilisant un correcteur orthographique. Ce livrable final est constitué (i) d'un fichier unique contenant votre texte et vos diagrammes et (ii) d'un répertoire contenant tous les fichiers de code source et le fichier jar, doit être compressé dans un fichier zip qui portera obligatoirement le nom : Projet_Equipe_X.zip et devra être déposé via Moodle; Equipe_X doit être remplacé par les noms de familles des membres de l'équipe. Toute soumission qui ne se conformera pas à ces directives pourrait être refusée.

Présentations des projets

La dernière séance du 14 décembre sera réservée à la présentation des projets, selon l'ordre alphabétique des noms de familles des étudiants. La présentation est demandée pour permettre aux étudiants de fournir une démonstration de l'utilisation des fonctionnalités du système, si possible toutes les fonctionnalités requises; sa durée est 20 minutes avec une période ouverte de question pour vérifier la participation de chaque membre d'une équipe à toutes les parties du projet. L'utilisation de Powerpoint n'est pas demandée, ni exclue. Vu que le temps d'une seule séance pourrait pas être suffisant pour toutes les fonctionnalités à présenter, certains

étudiants auront à présenter ultérieurement à des dates et heures que le correcteur des projets va leur offrir et ce au plus tard avant la fin de la session.

Bon travail

Voir Annexe plus bas

Annexe

Voici une documentation sur le format Json en ce lien . Vous y trouvez des références vers une bibliothèque Java pour manipulation de fichiers Json et leur représentation en dictionnaires ou autres objets comme celle dans ce lien.

Un exemple au format json peut prendre la forme suivante pour stocker des données structurées comme où l'on trouve une liste de dictionnaires qui peuvent être utilisés comme des enregistrements dans une base de données :

```
{[
      {
            color: "red",
            value: "#f00"
      },
      {
            color: "green",
            value: "#0f0"
      },
      {
            color: "blue",
            value: "#00f"
      },
      {
            color: "cyan",
            value: "#0ff"
      },
      {
            color: "magenta",
            value: "#f0f"
      },
      {
            color: "yellow",
            value: "#ff0"
      },
```

```
        {
                color: "black",
                value: "#000"
        }
] }
```

Pour ceux qui préfèrent la sérialisation Json des objets, la sérialisation Json des instances de chaque classe peut aussi être envisagée pour stocker les données de plusieurs objets des différents types.