

## Devoir et projet (Énoncé en première version)

Devoir à soumettre au plus tard le 12 juillet à 23 h 55

Projet à soumettre au plus tard le 03 août à 23 h 55

C'est un travail en équipe de deux membres. 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. Généralement il faut 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 recommandé 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 livrables de votre projet. La correction de votre projet peut prendre 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 le plutôt possible.

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

## Brève description du logiciel TimeLog

---

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 TimLog sont : design1 (haut niveau), design2 (détaillé), implémentation, test, et déploiement. Cet ensemble de disciplines est inspiré du processus unifié de dev. de logiciel vu 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 :

La description suivante du système dépasse les limites des livrables demandés pour le devoir et le projet, mais elle offre une vue assez complète pour permettre l'élaboration des livrables.

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 les dernières 26 périodes de paie (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, et qui sont : 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 de l'initialisation 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 travailler 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 3, mais l'admin peut modifier la valeur de NPE.
10. L'administrateur du système se connecte avec le nom d'utilisateur admin et un mot de passe admin qu'il peut changer plus tard.
11. Le compte admin peut modifier les noms d'utilisateur 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'utilisateur 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 argument 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 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 votre modélisation du projet l'interface **PayrollInteface** et le type **Payroll** mentionnées plus haut.
- d. Le solution doit permettre de communiquer correctement les données de paies au sous-système de paie mentionné plus haut.

## 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'utilisateurs et IDs : employé1, ID=1; employé2, ID=2; employé3, ID=3; + un compte admin nommé admin avec nom utilisateur 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 certains 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 11 heures pour chaque discipline du projet projet1; employé2 ayant travaillé 21 et 22 heures pour chaque discipline de deux projets projet1 et projet2 respectivement; employé3 ayant travaillé 31, 32 et 33 heures pour chaque discipline des trois projets projet1, projet2 et projet3 respectivement .

Travail demandé pour le devoir au plus tard le 12 juillet à 23 h 55 ( compte pour 5 pts)

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

1. Un ou plusieurs diagrammes de cas d'utilisation qui comprennent tous les cas d'utilisation de TimeLog, à inclure dans un fichier pdf.
2. Une documentation détaillée des deux cas d'utilisation « débiter une activité » et « Terminer une activité » relié au point numéro 14 plus haut. Un cas d'utilisation doit montrer l'aspect interactive, 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. La documentation demandée doit être limitée au scénario principal avec seulement **un seul scénario alternatif** pour chaque cas. Cette documentation doit être incluse dans le même fichier pdf mentionné dans le point 1 ci-haut.

3. Un fichier de type lien pointant votre projet 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, en attendant de le peupler plus tard quand vous compléter votre projet.

## Soumission du projet à soumettre au plus tard le 03 aout à 23 h 55

### Les Livrables avec barème indicatif sur 15 points

Votre travail n'est pas d'implémenter tout le système TimeLog, mais plutôt correspond essentiellement à une partie de la phase d'élaboration, ainsi on se limite aux livrables ci-dessous.

Ainsi, la soumission du projet doit comprendre les livrables suivants :

1. Le modèle du domaine en diagramme de classes conceptuel.
2. Un diagramme de séquence **DS pour le scénario principal du cas** d'utilisation « Débuter une activité »;
3. Un diagramme de séquence **DS pour le scénario principal du cas** d'utilisation « Terminer une activité »;
4. Le diagramme des classes de conception. Ce diagramme doit montrer, entre autres, les opérations illustrées dans les deux DS mentionnés plus haut.
5. En utilisant la génération de code de Modélio ou d'un autre outil que vous proposez à l'enseignant, veuillez créer un prototype en Java du projet qui implémente au minimum les scénarios principaux des deux cas d'utilisation « Débuter une activité » et « Terminer une activité »;
  - a. Remarque : la génération du code Java doit créer un squelette de votre projet qui automatise la création des opérations (sans implémentation) identifiées dans les DS demandés.
  - b. Vous êtes supposés implémenter en java les opérations du point a. ci-haut de façon à permettre d'exécuter les scénarios principaux des deux cas d'utilisation « Débuter une activité » et « Terminer une activité ». Cette implémentation doit sauvegarder les infos de début et de terminaison d'activité dans un fichier Json (ou un fichier texte de votre choix comme une alternative de dernier recours) .
    - i. Pour alléger la partie implémentation, on se contente d'identifier l'utilisateur en lui permettant de saisir son nom et son ID, ceci vous épargne l'implémentation d'une gestion de comptes d'utilisateurs pour les employés.

6. On vous demande de fournir séparément le projet Modélio que vous avez utilisé pour la génération du code mentionné plus haut, et d'indiquer la procédure que vous avez suivi pour la génération du code.

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 les plus complets possible. Tous les modèles doivent être créés avec Modélio, même s'ils sont à fournir comme figures dans le fichier pdf mentionné plus loin.

À l'exception du code source et du fichier de distribution **jar**, et du **fichier lien** vers votre repos github, tous les autres éléments des livrables 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 de (i) d'un fichier unique contenant votre texte et vos diagrammes, (ii) un répertoire nommé « diags » contenant les fichiers des diagrammes demandés nommés, modeleDuDomaine, DS1, DS2 et diagClasses; (iii) d'un répertoire nommé « TimeLogCode » contenant tous les fichiers de code source et le fichier jar. Les trois parties (i), (ii) et (iii) doivent 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 séance du 6 aout sera réservée principalement 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 des deux cas d'utilisation demandés; sa durée est de 10 minutes. L'utilisation de Powerpoint n'est pas demandée, ni exclue. Vu que le temps d'une seule séance pourrait ne pas être suffisant pour toutes les présentations, 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 d'été.

*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.

