INF2050 – Outils et pratiques de développement logiciel
Projet de session – Automne 2024

Demande initiale

Votre client vous demande de lui livrer ces fonctionnalités pour le **6 octobre 2024, avant 23h55**. La date de livraison n'est pas négociable.

L'application à développer est un logiciel qui effectuera la validation des déclarations d'activités de formation continue pour les membres d'un ordre professionnel.

Chaque ordre professionnel impose à ses membres d'effectuer un certain nombre d'heures d'activité de formation pour une période donnée. La période en question est appelée un cycle. Dans ce cas-ci, le cycle dure 2 ans et les membres de l'ordre doivent effectuer un minimum de 40 heures de formation continue durant le cycle.

Le logiciel ne possèdera pas d'interface utilisateur car il est destiné à être invoqué à partir d'une application web. Le contrat ne consiste donc qu'au développement du moteur de validation de l'application.

Fonctionnalités

Le fichier d'entrée, en format JSON, aura l'air de ceci :

```
{
    "numero_de_permis": "A0001",
    "cycle": "2023-2025",
    "heures_transferees_du_cycle_precedent": 2,
    "activites": [
       {
             "description": "Cours sur la déontologie",
             "categorie": "cours",
             "heures": 14,
             "date": "2024-03-20"
       },
       {
             "description": "Séminaire sur l'architecture contemporaine",
             "categorie": "séminaire",
             "heures": 10,
             "date": "2024-01-07"
       },
       {
             "description": "Rédaction pour le magazine Architecture moderne",
             "categorie": "rédaction professionnelle",
             "heures": 6,
             "date": "2024-10-22"
       },
       {
             "description": "Participation à un groupe de discussion sur le
                         partage des projets architecturaux de plus de 20 ans",
             "categorie": "groupe de discussion",
             "heures": 6,
             "date": "2024-04-01"
       },
       {
             "description": "Visite d'établissements architecturaux",
             "categorie": "voyage",
             "heures": 2,
             "date": "2024-02-02"
       }
    ]
}
```

Le fichier de résultat généré par le logiciel devra ressembler à ceci :

```
{
    "complet": false,
    "erreurs": [
            "L'activité Visite d'établissements architecturaux est dans une
             Catégorie non reconnue. Elle sera ignorée.",
            "Il manque 2 heures de formation pour compléter le cycle."
      ]
}
```

Le programme devra prendre le fichier d'entrée comme argument lors de l'exécution du logiciel dans une console (paramètre au main). Le fichier de sortie devra également être spécifié à la console. Exemple :

```
java -jar FormationContinue.jar declaration.json resultat.json
```

Voici la liste des catégories reconnues pour les activités de formation continue :

- cours
- atelier
- séminaire
- colloque
- conférence
- lecture dirigée
- présentation
- groupe de discussion
- projet de recherche
- rédaction professionnelle

Voici les règles d'affaires à valider :

- Dans un premier temps, uniquement le cycle "2023-2025" sera supporté. L'utilisation de tout autre cycle devra produire un message d'erreur dans le fichier de sortie.
- Toutes les activités déclarées pour le cycle 2023-2025 doivent avoir été complétées entre le 1er avril 2023 et le 1er avril 2025 inclusivement. Si une activité déclarée a été complétée à l'extérieur de cet intervalle, un message doit être produit dans le fichier de sortie et l'activité sera ignorée des calculs. Toutes les dates sont indiquées en format ISO 8601 (AAAA-MM-JJ).
- Le numéro de permis doit être une lettre majuscule suivie de 4 chiffres. La lettre doit être A, T, M ou K. Si le numéro de permis est invalide, un message doit être produit dans le fichier indiquant que le fichier d'entrée est invalide et que le cycle est incomplet et le programme s'arrête.
- Toutes les activités doivent appartenir à une des catégories reconnues mentionnées précédemment. Si une activité utilise une catégorie non reconnue, un message doit être produit dans le fichier de sortie et l'activité sera ignorée des calculs.
- Le champ "heures_transferees_du_cycle_precedent" contient des heures de formation en surplus qui ont été complétées lors du cycle précédent et qui peuvent être utilisées dans le cycle courant. Ce nombre ne peut pas être négatif et ne doit pas être supérieur à 7. S'il est supérieur à 7, un message doit être produit dans le fichier de sortie et uniquement 7 heures seront considérées lors des calculs. S'il est négatif un message doit être produit dans le fichier de sortie et la valeur est ignorée dans les calculs. Cette règle s'applique aux heures des différentes activités.
- Un minimum de 40 heures de formation doivent être déclarées dans le cycle. Il n'y a pas de maximum. En dessous du 40 heures, un message doit être produit dans le fichier de sortie.

- Un minimum de 17 heures doivent être déclarées pour les catégories suivantes : cours, atelier, séminaire, colloque, conférence, lecture dirigée. Autrement dit, la somme des heures des activités dont la catégorie appartient à la liste ci-dessus, doit être supérieure ou égale à 17 heures. En dessous du 17 heures, un message doit être produit dans le fichier de sortie. Les heures transférées du cycle précédent sont comptabilisées dans cette somme.
- Un maximum de 23 heures peuvent être déclarées dans la catégorie présentation. Au-delà de 23 heures, les heures supplémentaires sont ignorées des calculs mais aucun message n'est produit dans le fichier de sortie.
- Un maximum de 17 heures peuvent être déclarées dans la catégorie groupe de discussion. Au-delà de 17 heures, les heures supplémentaires sont ignorées des calculs mais aucun message n'est produit dans le fichier de sortie.
- Un maximum de 23 heures peuvent être déclarées dans la catégorie projet de recherche. Au-delà de 23 heures, les heures supplémentaires sont ignorées des calculs mais aucun message n'est produit dans le fichier de sortie.
- Un maximum de 17 heures peuvent être déclarées dans la catégorie rédaction professionnelle. Au-delà de 17 heures, les heures supplémentaires sont ignorées des calculs mais aucun message n'est produit dans le fichier de sortie.
- Les heures d'une activité doivent être supérieures ou égales à 1 et elles sont des valeurs entières. Si une activité possède une heure invalide, un message est produit dans le fichier de sortie et l'activité est ignorée des calculs.

Contraintes technologiques

Voici les contraintes que vous devez respecter :

- Le logiciel doit être développé avec le langage de programmation Java (JDK13 ou plus).
- Il est impératif d'utiliser l'environnement de développement intégré IntelliJ.
- Les fichiers d'entrées et de sorties doivent être des documents JSON.
- Les sources doivent être entreposées dans un dépôt GIT sous GitLab à l'UQAM.
- Les fichiers d'entrée et de sortie doivent être en UTF-8.

Exigences non fonctionnelles

Voici quelques contraintes à respecter qui touchent votre code :

- Votre équipe doit s'entendre sur un style uniforme à appliquer au code. Les particularités de votre style doivent être documentées dans un fichier `style.md`, rédigé en *markdown*, à la racine de votre projet. Il est permis de faire référence à un document déjà existant. Le style inclut également la langue utilisée pour nommer vos variables, méthodes ou classes ou pour rédiger vos commentaires dans le code.
- Une fois votre style défini, tout votre code doit être modifié pour le respecter.
- Tout commentaire dans le code doit être pertinent, c'est-à-dire qu'un commentaire doit documenter ce que le code ne décrit pas déjà de façon évidente. Votre approche face aux commentaires doit respecter le chapitre 4 du livre Coder proprement.
- Vos méthodes ne devraient pas dépasser 10 lignes de code et chaque méthode devrait respecter le "Principe de responsabilité unique" décrit dans le chapitre 3 du livre Coder proprement.
- Vous devez rédiger suffisamment de « bons » tests unitaires avec JUnit5 afin d'acquérir une couverture de tests d'au moins 50% sur votre projet (couverture de branches). Vos tests doivent suivre les règles vues en classe et le code de vos tests doit également être propre.

Exigences techniques (Pénalité 20%)

- Votre dépôt doit se nommer **exactement** *inf2050-a24-projet-equipe#*
- L'URL de votre dépôt doit être **exactement** *https://gitlab.info.uqam.ca/<utilisateur>/inf2050-a24-projet-equipe#* où <*utilisateur*> doit être remplacé par l'identifiant de votre représentent, et # par le numéro de votre équipe (ex : equipe2).
- Votre dépôt doit être **privé**
- Les usagers *@correcteurs* et *@dogny_g* doivent avoir accès à votre projet comme *Maintainer*

Remise

Le travail est automatiquement remis à la date de remise prévue. Vous n'avez rien de plus à faire. Assurez-vous d'avoir votre travail disponible sur votre **branche master ou main** qui sera considérée pour la correction. Tous les *commit*s après le 6 octobre 2024 à 23:55 ne seront pas considérés pour la correction.

Barème

Critère	Points
Fonctionnalité	/40
Respect du style et qualité du code	/15
Utilisation de git	/15
Qualité des tests et couverture	/20
Documentation	/10
Total	/100

Plus précisément, les éléments suivants seront pris en compte:

- **Fonctionnalité (40 points):** Le programme compile sans erreurs et affiche le résultat attendu.
- **Respect du style et qualité du code (15 points):** Les identifiants utilisés sont significatifs et ont une syntaxe uniforme, fonction courte, le code est bien indenté, il y a de l'aération autour des opérateurs et des parenthèses, le programme est simple et lisible. Pas de bout de code en commentaire ou de commentaires inutiles. Le code doit être bien factorisé (pas de redondance). La présentation est soignée.
- **Utilisation de Git (15 points):** Les modifications sont réparties en *commits* atomiques. Le fichier .gitignore est complet. Les messages de *commit* sont significatifs et uniformes.
- **Qualités des tests et couverture (20 points):** La qualité et la pertinence des cas de tests, l'atteinte de la couverture de test requise, la propreté du code de test.
- **Documentation (10 points):** Le fichier *equipe.md* contenant la charte et la composition de l'équipe, ainsi que les fichiers *style.md* et *README.md* sont complets et respectent le format *Markdown*. Le fichier README résume les technologies présentes et explique comment compiler et exécuter votre logiciel.

Clarification :

- Si le fichier d'entrée n'existe pas (chemin invalide) ou est invalide (JSON mal formé), un message significatif est affiché à la console et le programme s'arrête.
- Si le cycle est invalide, on produit un message d'erreur dans le fichier de sortie et on arrête le programme. Plus besoin de continuer la validation.
- Si une activité est déclarée en dehors de l'intervalle du cycle, on produit un message d'erreur dans le fichier de sortie et l'activité est ignorée dans le calcul. De même si la date d'une activité n'est pas valide (mauvais format, etc.), on produit un message d'erreur dans le fichier de sortie et l'activité est ignorée du calcul.
- Si une catégorie n'est pas reconnue, un message doit être produit dans le fichier de sortie et l'activité sera ignorée des calculs.