# Demande initiale

Votre client vous demande de lui livrer ces fonctionnalités pour le 21 février 2021, avant 23h55. La date de livraison n'est pas négociable.

L'application à développer est un logiciel qui calculera des remboursements de réclamations d'assurances de soins de santé.

Dans ce domaine, il existe beaucoup de clauses, de cas d'exceptions et de particularités avec les contrats d'assurance, et les contrats de notre client ne font pas exception. Il a donc besoin d'un coup de main pour l'aider à se retrouver dans toutes ses règles afin d'assurer une croissance à son entreprise.

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 "back-end" de l'application.

## Fonctionnalités

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

```
{
    "client": "100323",
    "contrat": "A",
    "mois": "2021-01",
    "reclamations": [
        {
            "soin": 100,
            "date": "2021-01-11",
            "montant": "234.00$"
        },
        {
            "soin": 200,
            "date": "2021-01-13",
            "montant": "90.00$"
        },
        {
            "soin": 334,
            "date": "2021-01-23",
            "montant": "125.00$"
        }
    ]
}
```

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

```
{
    "client": "100323",
    "mois": "2021-01",
    "remboursements": [
        {
            "soin": 100,
            "date": "2021-01-11",
            "montant": "58.50$"
        },
        {
            "soin": 200,
            "date": "2021-01-13",
            "montant": "22.50$"
        },
        {
            "soin": 334,
            "date": "2021-01-23",
            "montant": "0.00$"
        }
    ]
}
```

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 Refund.jar inputfile.json refunds.json
```

Il existe 4 types de contrat différents : A, B, C, D. Voici les différentes catégories de soin en fonction de leur numéro et les remboursements pour chaque type de contrat :

| Numéro de soin | Catégorie de soin | A | B | C | D |
|---|---|---|---|---|---|
| 0 | Massothérapie | 25% | 50% max 40$ | 90% | 100% max 85$ |
| 100 | Ostéopathie | 25% | 50% max 50$ | 90% | 100% max 75$ |
| 200 | Psychologie individuelle | 25% | 100% max 70$ | 90% | 100% max 100$ |
| [300..399] | Soins dentaires | 0% | 50% | 90% | 100% |
| 400 | Naturopathie, acuponcture | 0% | 0% | 90% | 100% max 65$ |
| 500 | Chiropratie | 25% | 50% max 50$ | 90% | 100% max 75$ |
| 600 | Physiothérapie | 40% | 100% | 90% | 100% max 100$ |
| 700 | Orthophonie, ergothérapie | 0% | 70% | 90% | 100% max 90$ |

Voici également quelques validations à effectuer sur les données d'entrée :

- Le numéro de client doit obligatoirement être composé de 6 chiffres, aucun autre caractère que des chiffres ne doit se retrouver dans le numéro de client.
- Le contrat doit être une des quatre lettres suivantes : A, B, C, D. La lettre doit toujours être en majuscule.
- L'application recevra toutes les réclamations d'un client pour un mois dans le même document JSON, le mois est spécifié dans le champs mois sous le format AAAA-MM.
- Il faut vérifier que chacune des réclamations est bel et bien faite pour le mois qui est traité. Un

soin fait le 2 février 2021 ne doit pas être considéré dans une feuille de janvier 2021.
- Le signe de dollar ($) est toujours présent à la fin d'un montant.
- Tous les montants doivent supporter le point «.» et la virgule «,» pour délimiter les dollars et les cents dans le fichier d'entrée. Donc, peu importe si le montant provenant du fichier d'entrée utilise un point ou une virgule, ça devrait fonctionner. Par contre, le fichier de sortie ne doit toujours contenir que des points pour délimiter les dollars et les cents.
- Le numéro de soin doit être valide selon le tableau ci-haut.

Les données d'entrée doivent respecter toutes ces règles pour être considérées valides. Si les données d'entrée ne sont pas valides, l'application devra générer le fichier de sortie suivant :

```
{
    "message": "Données invalides"
}
```

### Contraintes technologiques

Voici les contraintes que vous devez respecter :
- Le logiciel doit être développé avec le langage de programmation Java (JDK11 ou plus).
- Il est impératif d'utiliser l'environnement de développement intégré IntelliJ.
- JUnit5 sera utilisé comme Framework de test.
- Les fichiers d'entrées et de sorties doivent être des documents JSON.
- Les sources doivent être entreposées dans un dépôt GIT sous GitLab.
- 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.
- Tous commentaires dans le code doivent être pertinents, 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 afin d'acquérir une couverture de tests d'au moins 40% sur tout votre projet. Vos tests doivent suivre les règles vues en classe et le code de vos tests doit également être propre.

### Autres exigences

Quelques exigences supplémentaires à considérer :

- Toutes les dates doivent être spécifié en format ISO 8601 (AAAA-MM-JJ).
- Vous devez inclure le fichier *equipe.md* contenant la composition et la charte d'équipe à la racine de votre dépôt.