Demande de changement #1

Votre client vous demande de lui livrer ces fonctionnalités pour le 4 avril 2021, avant 23h55. La date de livraison n'est pas négociable.

Le contenu de ce document s'ajoute à la demande initiale du client.

Exigences fonctionnelles

Depuis la demande initiale, quelques changements ont été apportés aux besoins de l'entreprise et votre application devrait refléter cette nouvelle réalité.

- Dans le fichier d'entrée, les propriétés «client» et «contrat» n'existent plus. Elles ont été remplacées par une nouvelle propriété «dossier» qui contiendra la lettre du contrat suivi du numéro du client. Exemple :

```
{
    "dossier": "A100323",
    "mois": "2021-01",
    "reclamations": [
        {
            "soin": 100,
            "date": "2021-01-11",
            "montant": "234.00$"
        }
    ]
}
```

- Dans le fichier de sortie, la propriété «client» a également été remplacée par la propriété «dossier» qui doit contenir exactement la même valeur que dans le fichier d'entrée.

- Un nouveau type de contrat a été créé. Il possède la lettre «E». Voici les remboursements pour les soins déjà connus :

Numéro de soin	Catégorie de soin	E
0	Massothérapie	15%
100	Ostéopathie	25%
200	Psychologie individuelle	12%
[300..399]	Soins dentaires	60%
400	Naturopathie, acuponcture	25% max 15$
500	Chiropratie	30% max 20$
600	Physiothérapie	15%
700	Orthophonie, ergothérapie	22%

- Notre client rembourse maintenant un soin supplémentaire : «Kinésithérapie», qui possède le numéro de soin 150. Voici les remboursements pour chaque type de contrat :
  - A : 0%
  - B : 0%
  - C : 85%
  - D : 100% max 150$
  - E : 15%

- Notre client rembourse maintenant un soin supplémentaire : «Médecin généraliste privé», qui possède le numéro de soin 175. Voici les remboursements pour chaque type de contrat :
  - A : 50%
  - B : 75%
  - C : 90%
  - D : 95%
  - E : 25% max 20$

- La psychologie individuelle pour le contrat de type B est maintenant couverte à 100% sans montant maximal.

- La chiropratie pour le contrat de type D est maintenant couverte à 100% sans montant maximal.

- L'osthéopathie pour le contrat de type A est maintenant couverte à 35%.

- L'osthéopathie pour le contrat de type C est maintenant couverte à 95%.

- La physiothérapie pour le contrat de type C est maintenant couverte à 75%.

- Lorsqu'une erreur de validation survient, un élément «message» contenant la valeur «Données invalides» est généré dans le fichier de sortie. Il faut désormais remplacer la valeur «Données invalides» par un message d'erreur significatif décrivant le problème présent dans le fichier d'entrée. Nous arrêtons la validation à la première erreur rencontrée. Il n'est pas nécessaire de faire une liste de toutes les erreurs présentes dans le fichier d'entrée.

- Il faut désormais gérer le cas où une propriété JSON serait manquante dans le fichier d'entrée. Donc, si une propriété devrait être présente dans le fichier d'entrée, mais qu'elle n'y est pas, c'est un cas d'erreur et il faut générer le fichier de sortie avec l'élément «message» qui décrit quelle propriété est manquante.

- Tous les messages d'erreur doivent être des phrases complètes. Ces messages pourront être présentés directement à l'utilisateur.

- Dans le fichier de sortie, après la liste des remboursements, vous devez rajouter une propriété «total» qui contiendra la somme de tous les remboursements pour ce mois. Par exemple :

```
{
    "dossier": "C100323",
    "mois": "2021-01",
    "remboursements": [
        {
            "soin": 100,
            "date": "2021-01-11",
            "montant": "90.00$"
        },
        {
            "soin": 315,
            "date": "2021-01-22",
            "montant": "180.00$"
        }
    ],
    "total": "270.00$"
}
```


Exigences fonctionnelles supplémentaires

Maximum mensuel

Nous ajoutons un nouveau concept à notre logique d'affaire : le maximum mensuel. Certains types de soin ont un montant maximal mensuel de remboursement. Autrement dit, en plus de toutes les autres règles déjà connues, si la somme des réclamations pour un soin en particulier dépasse le maximum mensuel permis pour ce soin, on rembourse jusqu'au montant maximal mensuel et l'excédent n'est pas remboursé.

Supposons qu'il existe un montant mensuel maximal de 200$ pour le soin 175 (Médecin généraliste privé). En considérant le fichier d'entrée suivant :


```
{
    "dossier": "A100323",
    "mois": "2021-01",
    "reclamations": [
        {
            "soin": 175,
            "date": "2021-01-11",
            "montant": "130.00$"
        },
        {
            "soin": 175,
            "date": "2021-01-14",
            "montant": "130.00$"
        },
        {
            "soin": 175,
            "date": "2021-01-15",
            "montant": "130.00$"
        },
        {
            "soin": 175,
            "date": "2021-01-17",
            "montant": "130.00$"
        }
    ]
}
```

Nous obtiendrions le fichier de sortie suivant :

```
{
    "dossier": "A100323",
    "mois": "2021-01",
    "remboursements": [
        {
            "soin": 175,
            "date": "2021-01-11",
            "montant": "65.00$"
        },
        {
            "soin": 175,
            "date": "2021-01-14",
            "montant": "65.00$"
        },
        {
            "soin": 175,
            "date": "2021-01-15",
            "montant": "65.00$"
        },
        {
            "soin": 175,
            "date": "2021-01-17",
            "montant": "5.00$"
        }
    ],
    "total": "200.00$"
}
```

Voici les maximums mensuels pour les types de soin qui en possèdent :

Numéro de soin	**Catégorie de soin**	**Maximum mensuel**
100	Osthéopathie	250.00$
175	Médecin généraliste privé	200.00$
200	Psychologie individuelle	250.00$
500	Chiropratie	150.00$
600	Physiothérapie	300.00$

Manipulation monétaire sécuritaire

Il est généralement connu des développeurs qu'il ne faut jamais manipuler des valeurs monétaires avec des types de données réels (ex. float, double). Ces types sont des approximations qui permettent une grande précision pour les nombres fractionnaires mais qui représentent mal les valeurs monétaires. Quelques manipulations monétaires simples sur un type réel suffisent à faire apparaître ou disparaître quelques sous, ce qui n'est définitivement pas souhaitable pour un système qui manipule de l'argent.

Une façon courante de régler ce problème est de créer une classe «Dollar» qui contient toute la logique de manipulation de la monnaie. Souvent, la donnée interne est la valeur monétaire en cents. Ainsi, 120.21$ est représenté comme 12021 cents. Une division entière par 100 nous permet d'obtenir le nombre de dollars et un modulo 100 nous permet d'obtenir les cents. Comme la valeur monétaire est toujours représentée avec une valeur entière (les cents), nous n'avons plus les problèmes occasionnés par les points flottants.

Vous devrez donc créer une classe «Dollar» qui permettra d'effectuer toutes vos manipulations monétaires de façon sécuritaire. Chaque calcul fait sur une valeur monétaire devra passer par la classe «Dollar». Vous devrez également écrire des tests unitaires sur chacune des fonctionnalités de cette classe pour vous assurer qu'elle fonctionne parfaitement bien.

Statistiques

Le logiciel devra maintenant accumuler des statistiques lors de chaque exécution. Vous devrez conserver les statistiques dans un fichier. Le fichier peut être de la nature que vous voulez (texte, JSON, XML ou autre, tant que ça ne nécessite aucune installation).

À chaque exécution, vous devrez calculer les statistiques suivantes :

- le nombre de réclamations valides traitées;
- le nombre de demandes rejetées;
- le nombre de soins déclarés pour chaque type de soin.

Pour produire la liste des statistiques, l'utilisateur exécutera le logiciel en spécifiant, en paramètres, l'option "-S" et les statistiques seront affichées à la console. Par exemple :

```
java -jar Refund -S
```

L'option "-SR" doit réinitialiser les statistiques, c'est-à-dire les remettre à 0. Un message de confirmation doit être affiché à la console.

```
java -jar Refund -SR
```

Exigences non fonctionnelles

Voici quelques contraintes à respecter qui touchent votre code :

- Vous devez utiliser Maven dans votre projet pour gérer la construction du logiciel et pour gérer les dépendances de votre projet.

- Vous devez toujours respecter le «Single Responsibility Principle» au niveau des méthodes de classe.

- Chacune de vos classes doit également respecter le «Single Responsibility Principle» tel qu'indiqué dans le chapitre 10 du livre «Coder proprement.»

- Vous devez documenter dans un fichier nommé *workflow.md*, le "workflow" git (Gitflow, Feature Branch, Forking, ...) utilisé dans votre projet et vous assurer de bien l'appliquer.

- Vous devez rédiger suffisamment de «bons» tests unitaires afin d'acquérir une couverture de tests d'au moins 80% de tout votre projet. Vos tests doivent suivre les règles vues en classe et le code de vos tests doit également être propre.