INF2015 – Développement de logiciels dans un environnement Agile
Projet de session – Hiver 2017

Demande de changement #1

Votre client vous demande de lui livrer ces fonctionnalités pour le 9 mars 2017, avant 17h30. 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": "2017-01",
    "reclamations": [
        {
            "soin": 100,
            "date": "2017-01-11",
            "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.

- Jusqu'à date, tous les montants ont été présentés avec un point «.» pour délimiter les dollars et les cents. Il faut également supporter 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.

- 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": "2015-01",
    "remboursements": [
        {
            "soin": 100,
            "date": "2015-01-11",
            "montant": "90.00$"
        },
        {
            "soin": 315,
            "date": "2015-01-22",
            "montant": "180.00$"
        }
    ],
    "total": "270.00$"
}
```

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. 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.