Travail pratique #2 : Gestion de stock d'une pharmacie

Objectifs

- Appliquer les arbres binaires de recherche à une problématique.
- Écrire un programme efficace en temps et en mémoire.

Problématique

Vous devez écrire un programme en Java nommé **`tp2`** pour la gestion de stock des médicaments dans une pharmacie. Votre programme reçoit en entrée un flux de transactions des types suivants :

- la réception d'une livraison de médicaments;
- une prescription: une demande d'une liste de médicaments pour un client;

- un affichage de stock de pharmacie
- établissement d'une date courante
- génération de commande des médicaments manquants lors d'un changement de la date courante

Selon les quantités des médicaments en stock, le programme doit déterminer s'il accepte la demande ou met la demande dans la commande. Lorsqu'une prescription est servie ou une livraison est effectuée, le stock des médicaments doit être ajusté. À chaque prescription, il faudra assigner un identificateur unique.

Chaque médicament possède une date d'expiration. Les médicaments expirés doivent être supprimés du stock.

Lorsqu'un client demande un médicament, il faudra fournir les exemplaires ayant la date d'expiration la plus proche, si cette date n'est pas supérieure à la date de fin de traitement médical complet, en considérant le nombre de renouvellements ainsi que la quantité de médicaments demandée dans la prescription.

Par exemple, si un client demande un médicament X pour un traitement d'une durée de 30 jours et avec 6 répétitions, pour satisfaire une demande, il faudra que le stock contienne la quantité suffisante de médicaments non expirés couvrant une période de 6*30 = 180 jours. Autrement dit, pour simplifier le traitement, on veut que la pharmacie fournisse la totalité de médicaments demandés non expirés qui devrait être utilisé durant la période de traitement. Si on n'a pas une quantité suffisante, la pharmacie doit commander la totalité de médicaments nécessaires à un fournisseur en générant un enregistrement correspondant dans la liste des commandes pour une date courante.

Pour chaque date de traitement courante des prescriptions, une seule liste des commandes doit être générée pour tous les médicaments avec les quantités insuffisantes du stock et pour ceux qui ne sont pas en stock. Les médicaments expirés doivent être retirés du stock.

Structure du programme

Syntaxe d'appel du programme tp2

Votre programme doit pouvoir être lancé en ligne de commande avec la syntaxe suivante :

```
java Tp2 nomfichier1.txt nomfichier2.txt
```

Les résultats dans le format spécifié produits par votre programme doivent être écrits dans le deuxième fichier **`nomfichier2.txt`** passé au programme comme deuxième argument (**`args[1]`**).

Format d'entrée et de sortie

Le flux d'entrée est une suite d'actions. Chaque action débute avec un mot-clé en majuscules spécifiant son type.

Action	Syntaxe et description
`DATE`	**`"DATE" date ";"`** Cette transaction spécifie la nouvelle date courante. La première date ne sera jamais avant le 2000-01-01. Les dates d'expirations ne vont pas dépasser 2020. La date courante est importante pour déterminer les médicaments expirés ou ceux qui s'approchent de la date d'expiration. Toutes les dates dans le TP2 sont spécifiées dans le format **`AAAA-MM-JJ`**. Il est nécessaire de valider les dates lues. Ainsi, vous devez gérer le nombre de jours par mois et les années bissextiles. Pour une date donnée, il faudra afficher une liste des médicaments commandés ultérieurement avec la quantité totale et ensuite, vider la liste. Si la liste est vide, on affiche la date courante et OK.

Exemple d'entrée:

```
DATE 2018-06-5 ;
```

Exemple 1 de sortie (liste est vide):

```
2018-06-5 OK
```

Exemple 2 de sortie (la liste n'est pas vide):

```
2017-10-27 COMMANDES :
Medicament2 40
Medicament3 7
Medicament6 3
Medicament8 4
Medicament9 4
```

PRESCRIPTION `"PRESCRIPTION" ":" (nom_médicament dose_traitement répétition)* ";"`

Cette transaction spécifie une demande d'une liste de médicaments pour un client. Chaque médicament est spécifié par son nom, la quantité de médicament d'un cycle de traitement (5 unités, une unité par jour pour le médicament 1) et le nombre de cycles (répétitions, 6 fois pour médicament 1).

Exemple d'entrée:

```
PRESCRIPTION  :
Medicament1    5       6
Medicament6    3       1
Medicament3    5       1
```

Exemple de sortie:

```
PRESCRIPTION 1
Medicament1 5 6  OK
```

```
Medicament6 3 1  COMMANDE
Medicament3 5 1  COMMANDE
```

APPROV `"APPROV" ":" (nom_médicament quantité expiration)* ";"`

La transaction spécifie un réapprovisionnement de médicaments.

Exemple d'entrée:

```
APPROV :
Medicament1 120 2018-05-29
Medicament5 10 2018-05-27
;
```

Exemple de sortie:

```
APPROV OK
```

Un médicament peut apparaître plusieurs fois, et ce, avec des dates d'expiration pouvant être différentes. Exemple:

```
APPROV :
Médicament14 12 2018-10-29
Médicament14  2 2018-06-03
Médicament10  8 2018-06-29
Médicament2   7 2018-06-29
;
```

STOCK `"STOCK" ";"`

La transaction affiche le stock courant. En sortie, on affiche **STOCK** suivi par une date courante, suivi par une liste de médicaments. Chaque médicament est affiché sur une nouvelle ligne. Les médicaments en rupture de stock ou expirés ne devraient pas être présents dans le stocke et, comme conséquence, affichés.

```
STOCK ;
```

Exemple de sortie:

```
STOCK 2017-10-27
Medicament1 80 2018-05-29
Medicament4 8 2017-10-30
Medicament5 10 2018-05-27

STOCK 2005-04-27
…
Medicament33 82 2009-09-01
Medicament33 96 2010-04-01
...
```

Les transactions doivent être traitées dans leur ordre d'arrivée (ordre d'apparition dans le fichier d'entrée). Une fois une transaction lue, il faut la traiter immédiatement afin d'afficher son résultat avant de pouvoir lire la prochaine.

Pour faciliter le ***parsing***, il y a au moins un espace blanc (espace, tabulation ou retour de ligne) après chaque chaîne de caractère ou nombre. Le nom de médicament ne contient jamais d'espace blanc.

Exemples d'utilisation

Exemple 1

```
java Tp2 exemple1.txt exemple1+.txt
```

exemple1.txt

```
APPROV :
Medicament1 120 2018-05-29
Medicament5 10 2018-05-27
;
DATE 2017-10-26 ;
STOCK ;
PRESCRIPTION  :
Medicament1    5       6
Medicament6    3       1
Medicament3    5       1
;
PRESCRIPTION :
Medicament1    3       2
```

exemple1+.txt

```
APPROV OK
2017-10-26 OK

STOCK 2017-10-26
Medicament1 120 2018-05-29
Medicament5 10 2018-05-27

PRESCRIPTION 1
Medicament1 5 6  OK
Medicament6 3 1  COMMANDE
Medicament3 5 1  COMMANDE

PRESCRIPTION 2
```

```
Medicament9    4       1
Medicament2    4       1
;
PRESCRIPTION :
Medicament3    2       1
;
PRESCRIPTION :
Medicament8    4       1
Medicament1    4       1
Medicament2    3       12
;
DATE 2017-10-27 ;
APPROV :
Medicament4 8 2017-10-30
;
STOCK
;
```

```
Medicament1 3 2  OK
Medicament9 4 1  COMMANDE
Medicament2 4 1  COMMANDE

PRESCRIPTION 3
Medicament3 2 1  COMMANDE

PRESCRIPTION 4
Medicament8 4 1  COMMANDE
Medicament1 4 1  OK
Medicament2 3 12  COMMANDE

2017-10-27 COMMANDES :
Medicament2 40
Medicament3 7
Medicament6 3
Medicament8 4
Medicament9 4

APPROV OK
STOCK 2017-10-27
Medicament1 80 2018-05-29
Medicament4 8 2017-10-30
Medicament5 10 2018-05-27
```

Autres exemples

Voir d'autres exemples.

Remise

La date de remise: 10 juillet à 23:55 au plus tard.

Rapport

1. Autoévaluation
2. Analyse de la complexité temporelle en notation grand O de chacune des transactions suivantes:
    - PRESCRIPTION;
    - APPROV;
    - DATE;

Les complexités temporelles devraient être exprimées en fonction de :

- *n* - indique nombre de types de médicaments différents;
- *m* - indique nombre d'items sur la prescription
- *k* - indique nombre d'items sur la liste de demande;
- et de tout autre variable que vous jugeriez pertinent.

Évaluation

- Ce travail pratique vaut 10% de la note finale.

Grille d'évaluation

Critère	Description	Pondération
A.	**Respect des directives pour la remise**	/ 1
B.	• Appréciation générale • Structure du programme + Qualité du code	/ 1
C.	**Fonctionnement**	/ 5
D.	**Efficacité** Votre programme doit utiliser judicieusement les arbres binaires de recherche.	/ 1.5
E.	**Analyse des algorithmes** Complexité temporelle des fonctions et justification.	/ 1.5
	Total :	/ 10