Travail de synthèse

« Trouver les faudeurs »

420-335-AL
Programmation concurrente et distribuée

Note: / 120

Remise dans LÉA le 20 décembre avant minuit

Directives importantes:

- Ce travail pratique doit être réalisé seul. ATTENTION : Les « clones » ou les copies trop semblables se verront attribuer la note de 0 (plagiat).
- Remise
  - ✓ Vous devez remettre le code source de votre programme Java ainsi que tous les fichiers nécessaires pour compiler et exécuter votre programme à l'aide d'IntelliJ.
  - ✓ Inclure votre fichier .sql pour créer votre base de données
  - ✓ Pour réduire l'espace, prenez soin d'enlever le répertoire build
  - ✓ Ce travail doit être remis électroniquement dans LÉA avant la date limite

- La **qualité du français** est importante, prenez le temps de vous relire et de corriger vos fautes. Jusqu'à 10% de pénalité sera appliqué pour une mauvaise utilisation du français dans l'interface ou dans les commentaires.
- Le travail doit être remis au plus tard à la date et à l'heure indiquée dans LÉA. Après cette date, **10% de pénalité par jour sera appliqué**, jusqu'à un maximum de 5 jours de retard. Après 5 jours, la note est 0.
- Un programme qui présente des erreurs de compilation ne sera pas corrigé

Mise en contexte

1986

Été 1986, Gaétan Fréchette. Un mécanicien de carrière se lance en affaires. Il s'achète un tracteur muni d'une souffleuse et une pelle de déneigement et se lance dans l'aventure du déneigement d'entrée de cour et lance l'entreprise Déneigement Gaétan inc. Pendant l'hiver, Gaétan travaille fort tout l'hiver et récolte beaucoup d'argent.

Durant l'année 1986, l'entreprise de Gaétan déclare 212,000$ de revenu. Il déduit environ 100,000$ de dépenses en fournissant une copie de toutes les factures qu'il a dû payer pour supporter ses opérations. On y compte des factures d'essence, d'entretien mécanique, de frais de téléphone, de nouveaux pneus de tracteur, etc.

Le gouvernement calcule l'impôt à payer sur le revenu net. Le revenu net est la différence entre le revenu brut et le total des dépenses durant l'année. Par exemple, le revenu net de Gaétan pour 1986 se calcule comme suit :

Déneigement Gaétan inc 1986	
Revenu brut	212,000
Dépenses :	
Facture de livraison d'essence, 3 janvier	20,000
Facture de livraison d'essence 20 février	40,000
Facture de réparation, 2 juin	12,000
Facture de réparation, 14 juin	16,000
Facture de comptable, 10 mars	12,000
Revenu net ($212,000 – toutes les dépenses)	112,000

Gaétan a donc payé de l'impôt pot sur un revenu net de 212,000$. Si l'impôt est de 30%, Gaétan a donc payé 33,600$ d'impôt.

1987

N'étant pas de nature à partager, Gaétan décide donc en 1987 de réduire son impôt considérablement. Il fonde donc une autre compagnie, au nom de son beau-frère. La compagnie s'appellera Déneigement la bordée. L'idée est de séparer les revenus bruts entre les 2 compagnies, les maisons paires avec Déneigement Gaétan inc. et les maisons impaires avec Déneigement la bordée. Il utilisa le même tracteur pour les 2 compagnies. Le stratagème est de dupliquer les factures de dépenses. Pour une même facture, Gaétan en utilise une copie pour Déneigement Gaétan inc. et une autre pour Déneigement la bordée. Par exemple, pour 1987, le revenu net de Gaétan se calcule comme suit :

Déneigement la bordée inc 1987	
Revenu brut	108,000
Dépenses :	
Facture de livraison d'essence, 3 janvier	23,000
Facture de livraison d'essence 20 février	37,000
Facture de réparation, 2 juin	13,000
Facture de réparation, 14 juin	16,000
Facture de comptable, 10 mars	11,000
Revenu net ($108,000 – toutes les dépenses)	8,000

Déneigement Gaétan inc 1987	
Revenu brut	106,000
Dépenses :	
Facture de livraison d'essence, 3 janvier	23,000
Facture de livraison d'essence 20 février	37,000
Facture de réparation, 2 juin	13,000
Facture de réparation, 14 juin	16,000
Facture de comptable, 10 mars	11,000
Revenu net ($108,000 – toutes les dépenses)	6,000

Grâce à la duplication de factures, Gaétan a donc réduit son revenu net à 14,000$. Si l'impôt est de 30%, Gaétan a donc payé seulement 4,200$ d'impôt, donc une réduction considérable de 29,400$.

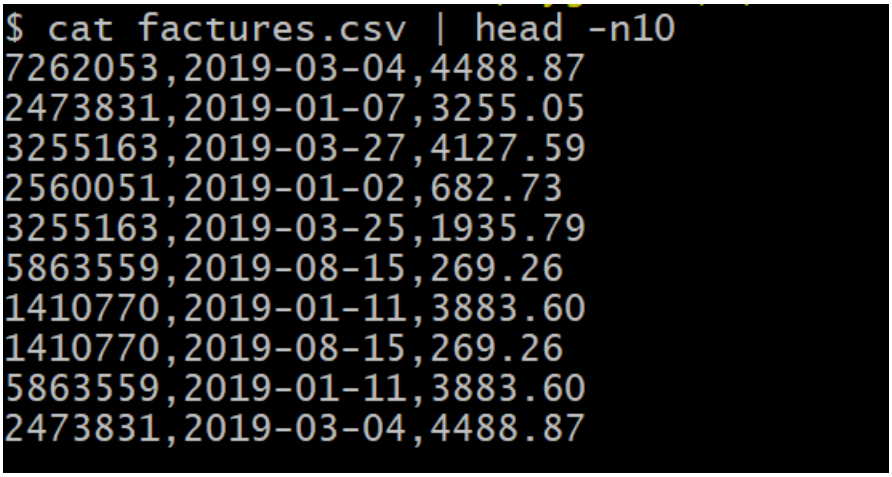
1988 et après

Gaétan a été reconnu coupable de fraude fiscale au cours de l'été 1988. Il servira désormais une sentence de 2 ans de travaux communautaires et a dû vendre son tracteur à son compétiteur pour rembourser au gouvernement l'impôt non perçu. La rumeur veut que Gaétan ait été balancé aux autorités par un étudiant du Cégep André Laurendeau. L'étudiant aurait mis à jour le stratagème de duplication de factures pendant qu'il s'occupait de la comptabilité de l'entreprise dans la cadre d'un stage d'été.

La duplication de factures est commune et le gouvernement vous donne le mandat de concevoir un programme pour découvrir les fraudeurs. On met donc à votre disposition toutes les factures déclarées par les entreprises du Québec et c'est à vous de faire le reste!

Question 1

Créez un programme Java qui va charger en mémoire le fichier de données de factures. Le fichier se nomme factures.csv et est inclus avec l'énoncé du travail. Le fichier contient plus de 1,000,000 de factures déclarées par plusieurs entreprises.



Les factures sont présentées sur 3 colonnes :

No d'entreprise	Numéro unique d'entreprise
Date	Date de la facture
Montant	Montant de la facture

Le programme va offrir un menu à l'usager. L'usager indique l'intention de charger le fichier factures.csv en mémoire en choisissant l'option 'c'. Le fichier se charge en mémoire de façon asynchrone. C'est-à-dire que le chargement doit se faire par un autre thread que le main thread. Le main thread doit normalement être utilisé pour afficher le menu et répondre aux demandes de l'usager sans jamais être bloqué. Pendant le chargement, l'usager peut donc choisir de faire d'autre opérations comme interrompre le chargement ou en voir le statut. Voir un exemple d'utilisation plus bas.

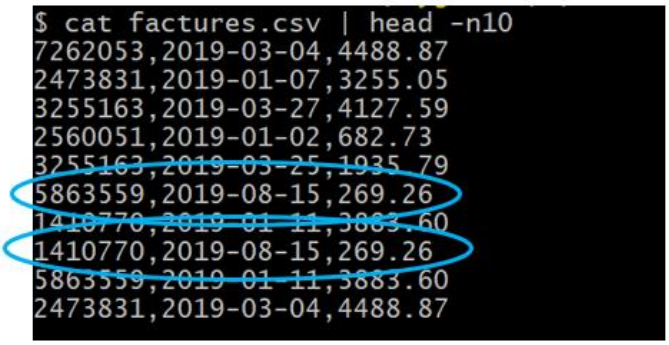
Pour répondre à cette question, vous devez **obligatoirement** :

- Charger les factures dans un TreeMap qui utilise une clée composée de la date et du montant. La clée devra déclarer l'override compareTo() comme démontré en classe. Voici le TreeMap désiré :

  TreeMap<FactureClee, List<String>>

  Le TreeMap devra contenir une liste de numéro d'entreprises qui ont utilisé une facture qui présente la même date et le même montant.
- Créer un projet **gradle**
- Utiliser **git** pour que le professeur puisse faire git log et voir le progrès

Le TreeMap contient la liste de toutes les factures. Chaque entrée du TreeMap représente toutes les factures qui présentent la même date et même montant. La liste de String contient tous les commerces qui ont utilisé une facture pour la même date et même montant.



L'image précédente représente une liste de factures. Deux d'entre elles ont la même date et le même montant. Ces deux factures seront insérées dans le TreeMap en utilisant la date et le montant comme clée et les deux commerces dans la liste comme suit:

Voici un exemple d'utiisation pour le chargement. Notez que le chargement se fait en arrière-plan et que le nombre de factures chargées augmente quand on demande au programme de donner le statut. On va faire l'implémentation du statut en Q2.

```
c: charger le fichier factures.csv
a: analyser les factures
f: afficher les fraudeurs
s: afficher le statut
i: interrompre
q: terminer et quitter
Entrez votre choix (c,a,f,s,i ou q) :c
Entrez votre choix (c,a,f,s,i ou q) :s
    ---------------------------
    Status
    ---------------------------
    Etat: Chargement
    Factures chargees: 577210
    Fraudeurs trouves: 0
    ---------------------------
Entrez votre choix (c,a,f,s,i ou q) :s
    ---------------------------
    Status
    ---------------------------
    Etat: Pret
    Factures chargees: 1001238
    Fraudeurs trouves: 0
    ---------------------------
Entrez votre choix (c,a,f,s,i ou q) :
```

Barème pour la question 1

Élément	Pondération
Question 1	
Utilisation d'un thread	5%
Utilisation de NIO pour charger le fichier	5%
Utilisation d'un TreeMap selon l'énoncé	5%
Application fonctionne de façon asynchrone comme démontré dans l'exemple	5%
Total	20%

Question 2

Le programme devra éventuellement traiter plusieurs millions de factures et les traitements prendront beaucoup de temps. On veut offrir la possibilité à l'usager de connaître l'état d'une opération. Par exemple, au cours du chargement, on veut donner à l'usager le nombre de factures chargées jusqu'à présent en mémoire. On veut également l'informer de l'opération actuellement en cours. Pour un exemple de d'utilisation, voir l'exemple en question 1.

Pour répondre à cette question, vous devez **obligatoirement** :

- Utiliser synchronize pour s'assurer qu'un seul thread à la fois peut lire ou écrire les données partagées entre plusieurs threads. Les données partagées sont : l'État, le nombre de factures chargées ainsi que le nombre de fraudeurs trouvés.

Barème pour la question 2

Élément	Pondération
Question 2	
Utilisation de la synchronisation pour les données partagées entre 2 threads	10%
Application affiche correctement les données comme présenté dans l'exemple	10%
Total	20%

Question 3

Une fois les données correctement chargées, on va procéder à l'analyse des données. Quand l'usager est prêt à lancer l'analyse, il va utiliser l'option 'a' du menu. Comme le chargement, l'analyse devra se faire en arrière-plan par l'utilisation d'un thread.

L'analyse consiste à trouver les faudeurs. **Un commerce qui a soumis plus de 10% de factures dupliquées est automatiquement soupçonné de commettre de la fraude.** Votre travail consiste donc à trouver ces commerces. Une fois que vous avez trouvé les commerce soupçonnés de fraude, le département des enquêtes de chargera de comparer manuellement les factures et de déterminer si il y a matière à poursuite.

Vous devez donc créer une liste de commerces qui contient les statistiques de factures, c'est-à-dire les le nombre de factures ainsi que le nombre de factures possiblement dupliqués. Une facture est considérée comme étant possiblement dupliquée quand plus de un commerce l'a utilisée. Donc si il y a plus d'un commerce pour une facture dans le TreeMap.

Voici un exemple d'exécution d'analyse :

```
    ----------------------------
    Status
    ----------------------------
    Etat: Pret
    Factures chargees: 1001238
    Fraudeurs trouves: 0
    ----------------------------
Entrez votre choix (c,a,f,s,i ou q) :a
Entrez votre choix (c,a,f,s,i ou q) :s
    ----------------------------
    Status
    ----------------------------
    Etat: Analyse
    Factures chargees: 1001238
    Fraudeurs trouves: 33
    ----------------------------
Entrez votre choix (c,a,f,s,i ou q) :s
    ----------------------------
    Status
    ----------------------------
    Etat: Pret
    Factures chargees: 1001238
    Fraudeurs trouves: 42
    ----------------------------
Entrez votre choix (c,a,f,s,i ou q) :
```

Tout comme le chargement, l'analyse doit se faire en arrière plan par l'utilisation d'un thread. L'usager peut voir la progression en affichant le statut. Le nombre de fraudeur trouvé devra donc augmenter à mesure que l'analyse progresse.

Barème pour la question 3

Élément	Pondération
Question 3	
Analyse et trouver les fraudeurs	10%
Utilisation d'un thread	10%
Total	20%

Question 4

Pendant l'analyse, à mesure qu'on trouve les fraudeurs, on doit insérer l'information dans la base de données. Vous devez créer la table Fraudeurs qui va contenir le numéro de commerce, le nombre de factures ainsi que le nombre de factures possiblement dupliquées. Vous devez sauvegarder les données de fraudeurs avec l'utilisation de JDBC. Vous devez créer un CRUD pour permettre l'ajout, la suppression, ainsi que la recherche par numéro de commerce.

Barème pour la question 4

Élément	Pondération
Question 3	
Ajout des fraudeurs dans la base de données pendant l'analyse	10%
CRUD pour les fraudeurs	10%
Total	20%

Question 5

Ajoutez une autre option pour afficher tous les fraudeurs qui ont été trouvés jusqu'à présent. On veut que l'usager puisse choisir l'option 'f'. Votre programme devra alors obtenir tous les fraudeurs de la base de données et de les afficher à l'écran.

```
Entrez votre choix (c,a,f,s,i ou q) :f
    ---------------------------------
    Faudeurs
    ---------------------------------
    5143322 28% de factures dupliquees
    5400375 89% de factures dupliquees
    5481629 31% de factures dupliquees
    5582698 43% de factures dupliquees
    5587211 44% de factures dupliquees
    5671456 35% de factures dupliquees
    5973628 46% de factures dupliquees
    6447459 12% de factures dupliquees
    ---------------------------------
Entrez votre choix (c,a,f,s,i ou q) :s
```

Ajoutez le code pour interrompre et quitter. Interrompre permet d'arrêter le chargement ou l'analyse sans quitter l'application. Quitter permet de compléter le traitement courant et de quitter.

De plus, assurez vous d'empêcher l'usager de lancer deux tâches en même temps. Si par exemple l'usager essaie de lancer une analyse pendant que le programme est en cours de chargement, on ne

doit pas lancer l'opération. Dans ce cas, on affiche un message à l'usager. Voici un exemple de message quand l'usager relance le chargement pendant qu'on est déjà en chargement :

```
c: charger le fichier factures.csv
a: analyser les factures
f: afficher les fraudeurs
s: afficher le statut
i: interrompre
q: terminer et quitter
Entrez votre choix (c,a,f,s,i ou q) :c
Entrez votre choix (c,a,f,s,i ou q) :c
Patientez, une operation est deja en cours: Chargement
Entrez votre choix (c,a,f,s,i ou q) :s
    ----------------------------
    Status
    ----------------------------
    Etat: Pret
    Factures chargees: 1001238
    Fraudeurs trouves: 0
    ----------------------------
Entrez votre choix (c,a,f,s,i ou q) :
```

Barème pour la question 5

Élément	Pondération
Question 6	
Affichage des fraudeurs avec l'option p	5%
Interrompre	5%
Quitter	5%
Total	15%

Question 6

Les JUnits suivants sont à créer :

1. Chargement dans le treeMap de deux factures présentant la même date et le même montant
2. Chargement dans le treeMap de deux factures présentant des dates et montants différents
3. Analyse de deux factures présentant la même date et le même montant
4. Analyse de deux factures présentant des dates et montants différents
5. Insertion dans la table Fraudeurs
6. Suppression de la table Fraudeurs par numéro de commerce
7. Recherche par numéro de commerce
8. Obtenir tous les fraudeurs de la bd
9. JUnit de votre choix

10. JUnit de votre choix

Les JUnits doivent respecter la norme FIRST.

Barème pour le travail

Élément	Pondération
Question 1	20%
Question 2	20%
Question 3	20%
Question 4	20%
Question 5	15%
Question 6	20%
Utilisation de GIT et Gradle	5%

Critères de correction

- Des points seront enlevés pour du code mal écrit, mal indenté ou qui ne respecte pas les normes de programmation.
- Des points seront enlevés si le code utilise une solution qui est différente de celle qui a été montrée en classe et qui est moins efficace ou qui présente des désavantages
- Des points seront enlevés si le code pourrait être significativement plus efficace ou plus propre

Important :
Un programme qui présente des erreurs de compilation ne sera pas corrigé