

INF 731 - Travail pratique no. 1

Énoncé du problème

Dans le cadre de ce travail pratique vous devrez concevoir une application qui permettra de produire la facture d'un client d'une succursale d'une chaîne de magasins. Naturellement, votre programme ne sait pas au point de départ combien il y a d'articles à facturer; votre programme devra donc faire le nécessaire pour réserver l'espace requis pour conserver durant le cours de l'exécution tous les articles devant être facturés.

Les données des articles à facturer au client seront inscrites dans un fichier de type texte¹. Votre programme devra lire en entier le fichier d'entrée où se trouvent les données et produire ensuite la facture attendue dans un fichier de sortie. Les éléments devant être indiqués sur la facture seront mentionnés plus loin.

Réalisation du travail

Le programme que vous devez réaliser doit être écrit en Microsoft Visual C# en utilisant au mieux les concepts de la programmation orientée objet que nous avons vu et verrons dans le courant des prochaines semaines. Ce travail fait appel à de nombreuses notions que nous voyons durant les huit premières rencontres du cours – création d'une classe, création d'instances, attributs, propriétés, méthodes, construction, composition, création de tableaux ou utilisation d'une collection, gestion des exceptions, etc.

Il y a aussi la question de la lecture et de l'écriture dans des fichiers, mais cette difficulté est assez minime et une courte explication en classe suffira à vous informer de la manière de procéder.

Format des données à lire

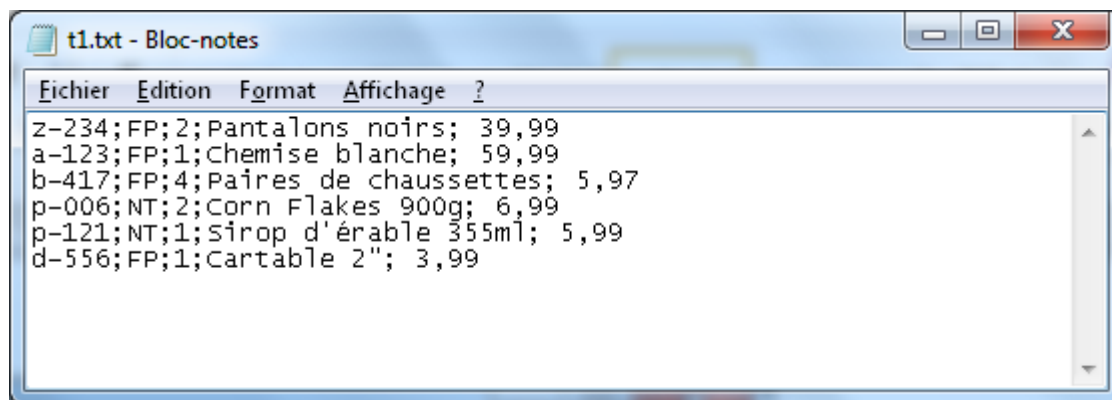
Cinq données décriront dans le fichier d'entrée un article qui sera facturé au client :

- Un numéro d'article, qui sera une chaîne de caractères sans espace (par exemple, on pourrait avoir la chaîne A-12345 pour décrire un numéro d'article);

¹ Dans un système réel, on aurait obtenu les infos des articles en interrogeant une base de données.

- Une catégorie, qui sera représentée par une chaîne de caractères pouvant avoir deux valeurs : NT ou FP. NT indiquera un article **n**on **t**axable et FP indiquera un article taxable au **f**édéral et au **p**rovincial. Les pourcentages de taxe qui s'appliquent sont ceux de la TPS et de la TVQ courantes. Chaque taxe s'applique sur le montant original de l'article, si cet article est taxable bien entendu;
- Une quantité, qui sera un entier positif ou égal à 0. Une quantité négative aurait pu être acceptable et signifier un article qu'on retourne au commerçant, mais il a été décidé de ne pas gérer cette possibilité pour le moment;
- Une description, qui sera une chaîne de caractères pouvant comporter des espaces;
- Un prix unitaire, dont le format dans le fichier sera celui d'un float ou d'un double.

Toutes ces données sont séparées dans le fichier d'entrée par des points-virgules. Afin de lire un fichier ainsi constitué pour produire une facture, votre programme doit demander à l'utilisateur au début de l'exécution le nom d'un fichier qui contient les articles à facturer au client. Voici par exemple le fichier t1.txt pouvant représenter des données à traiter :



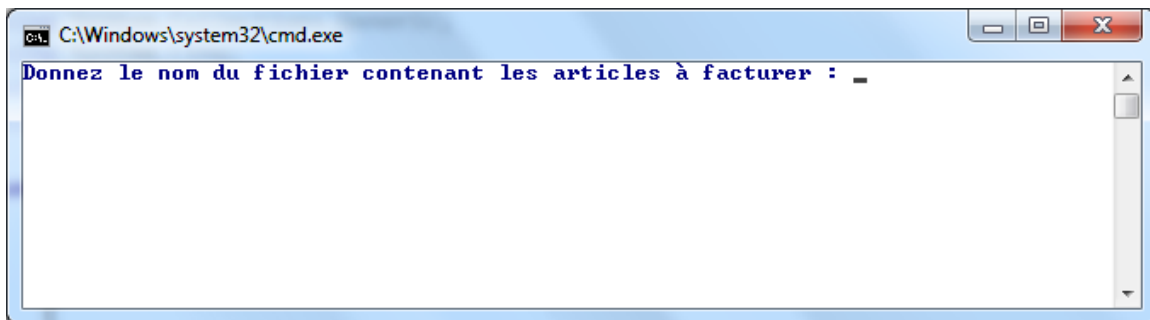
Vous avez l'assurance que dans le fichier,

1. Un numéro d'article est présent et ne comporte pas d'espace;
2. Le deuxième champ est nécessairement **nt** ou **fp**, en minuscules ou en majuscules. Il n'y a pas d'erreur sur ce champ mais il serait néanmoins prudent de le valider dans votre programmation puisqu'une entrée différente de **nt** ou **fp** signifierait qu'on a mal constitué le fichier;

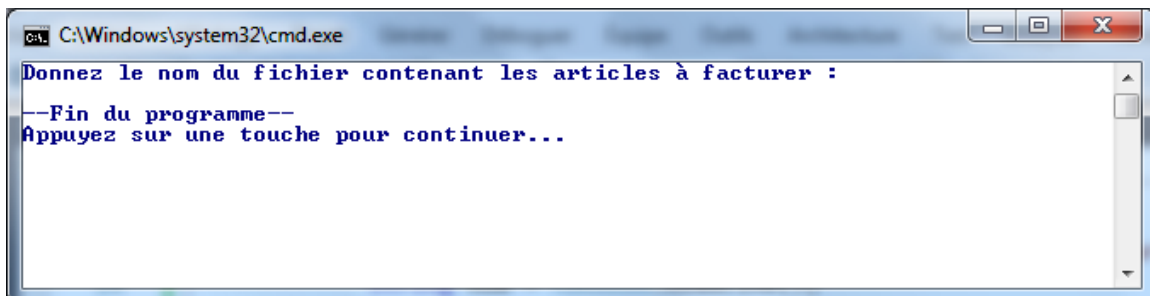
3. La quantité d'articles est un entier correctement formé; donc la lecture d'un entier ne posera pas de problème et se fera correctement dans le flux d'entrée; par contre, vous devrez vérifier que cet entier n'est pas négatif;
4. Il y a nécessairement une description mais il est impossible de la valider en tant que tel; le plus que vous pourrez faire, c'est de valider qu'elle n'est pas vide ou nulle;
5. Le montant indiqué en fin de ligne sera dans un format compatible avec un réel à virgule flottante. Il sera donc lisible comme un float ou un double; vous n'avez pas l'assurance que ce montant est positif, ce qui doit être vérifié².

Votre programme doit constituer une facture qu'il enverra dans un fichier de sortie. Votre programme créera un fichier dont le nom sera « **Facture -** » suivi du nom du fichier lu. Ainsi, si votre programme a obtenu de l'utilisateur qu'il doit traiter le fichier de données « **t1.txt** », le résultat produit par votre programme sera un fichier nommé « **Facture – t1.txt** ».

Lorsque votre programme se mettra en marche, il demandera à l'utilisateur le nom du fichier contenant les articles à facturer. Par exemple :

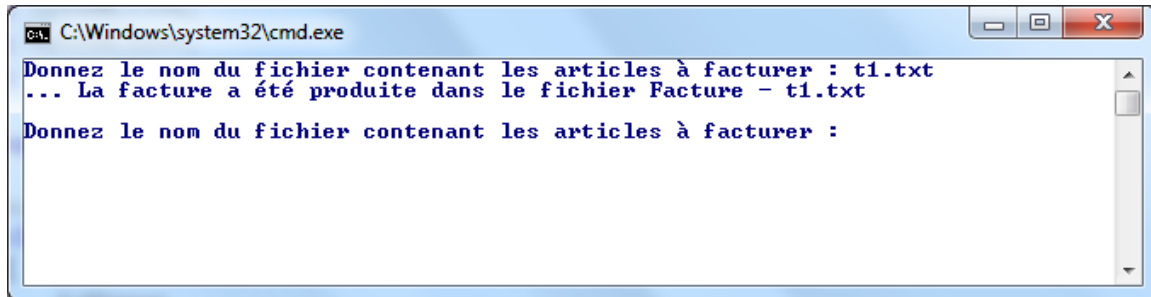


Si l'utilisateur entre un nom qui est vide ou qui n'existe pas, ceci met fin au programme.



² Notons qu'un montant négatif pourrait signifier un remboursement mais nous n'introduirons pas cet élément de complexité dans ce travail. Seuls les montants positifs seront acceptés.

Si le fichier existe, le programme le traite, constitue le fichier contenant la facture à produire (le résultat recherché) et redemande ensuite à l'utilisateur le nom du prochain fichier à traiter. Par exemple :

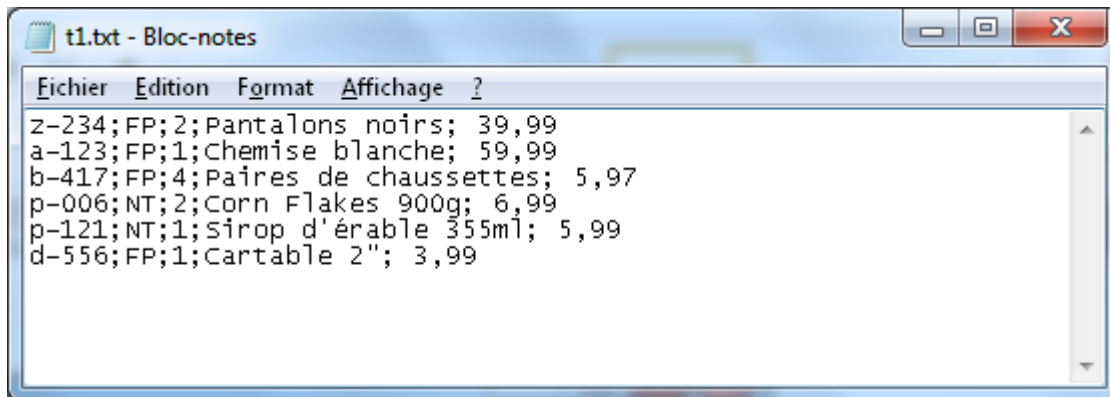


```
C:\Windows\system32\cmd.exe
Donnez le nom du fichier contenant les articles à facturer : t1.txt
... La facture a été produite dans le fichier Facture - t1.txt
Donnez le nom du fichier contenant les articles à facturer :
```

Résultat à produire : la facture

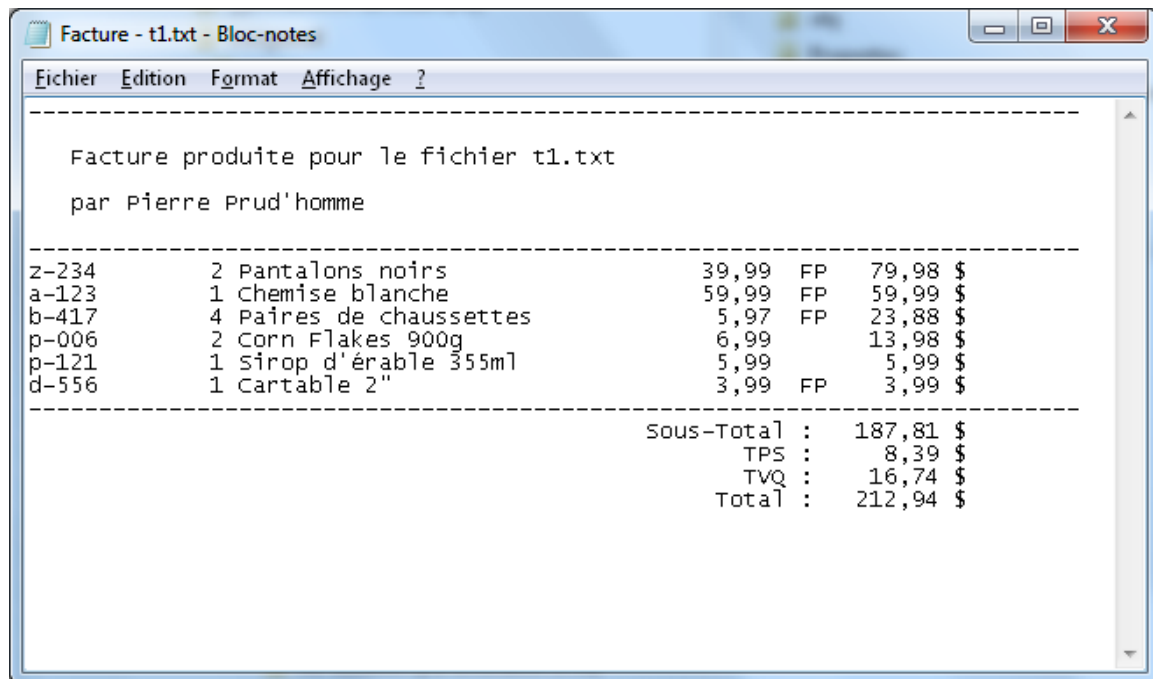
La facture à produire sera écrite par votre programme dans un fichier de sortie. Dans ce fichier, il doit d'abord y avoir un entête où se trouvent un titre et le nom du ou des auteurs du programme. Puis, il doit y avoir la liste des articles une ligne à la fois, le sous-total, les taxes puis enfin le grand total des articles.

Par exemple, pour le fichier d'entrée déjà présenté :



```
t1.txt - Bloc-notes
Fichier  Edition  Format  Affichage  ?
z-234;FP;2;Pantalons noirs; 39,99
a-123;FP;1;Chemise blanche; 59,99
b-417;FP;4;Paires de chaussettes; 5,97
p-006;NT;2;Corn Flakes 900g; 6,99
p-121;NT;1;Sirop d'érable 355ml; 5,99
d-556;FP;1;Cartable 2"; 3,99
```

La facture pourrait prendre la forme suivante :



```

Facture produite pour le fichier t1.txt
par Pierre Prud'homme

-----
z-234      2 Pantalons noirs          39,99  FP   79,98 $
a-123      1 Chemise blanche          59,99  FP   59,99 $
b-417      4 Paires de chaussettes      5,97  FP   23,88 $
p-006      2 Corn Flakes 900g          6,99          13,98 $
p-121      1 Sirop d'érable 355ml          5,99          5,99 $
d-556      1 Cartable 2"                    3,99  FP    3,99 $
-----
Sous-Total :    187,81 $
TPS :           8,39 $
TVQ :          16,74 $
Total :        212,94 $

```

Notez que dans l'exemple, la première colonne de prix correspond au prix unitaire de l'article et la deuxième colonne de prix, le prix des articles en fonction de leur nombre.

Bien sûr, la qualité de la présentation dans le fichier de sortie représentant la facture est importante. La facture doit être aisément lisible et la liste des articles facile à comprendre. Vous pouvez vous inspirer du mode de présentation que je vous suggère; il est possible de faire mieux, bien sûr.

Par contre, avant de vous soucier de la perfection de la présentation, vous devrez vous assurer que le traitement est correct et les résultats valables en toutes circonstances. Un résultat bien présenté mais incorrect ne vaudra ... pas grand chose.

Contraintes exigées de votre solution

Pour représenter chaque article de la facture, vous devez créer dans votre projet une classe `Article`. Chaque instance d'`Article` comportera des attributs pour son numéro, un attribut indiquant si c'est un article taxable, une description, la quantité achetée de cet article ainsi que le prix unitaire de l'article.

On voudra retrouver, parmi les opérations possibles sur un article, **que vous utilisiez ou non** chacune d'entre elles pour ce programme³:

- un constructeur paramétrique; il recevra en paramètre, dans cet ordre, une chaîne représentant le numéro de l'article, la chaîne lue du fichier d'entrée qui indique si cet article est taxable ou pas, un entier représentant la quantité d'articles à facturer, une chaîne représentant la description de l'article et un float ou un double qui représentera le prix unitaire de cet article;
- un constructeur de copie;
- des propriétés permettant de modifier les attributs et de faire connaître la valeur des attributs d'un `Article`; à vous de déterminer leur degré de visibilité correct;
- une surcharge de la méthode **`ToString()`** afin de simplifier la production de la facture.

Autres consignes relatives à la conception et à l'implantation

1. Il est convenu que nous ne savons d'aucune manière combien d'articles seront compris dans la facture. Un tableau dynamique ou un conteneur standard⁴ sera donc à utiliser dans le cadre de ce programme, sans doute dans une classe spécialement conçue pour conserver l'ensemble des instances d'`Article` par composition;
2. Durant la période de développement, vous pouvez utiliser un très grand tableau automatique pour simplifier votre programme; plus tard, quand l'ensemble de la logique fonctionnera, vous pourrez intégrer la notion de conteneur à l'intérieur de votre classe puisque ce détail d'implantation a normalement été encapsulé, ce qui le rend facilement modifiable. Le programme qui utilise les services de l'objet ne devrait donc pas être affecté par une modification de la représentation des attributs;
3. Pensez aux quelques cas de validation des données : vous devriez identifier dès le début les cas à rejeter et lancer / traiter des exceptions si ces cas se produisent.

³ On ne développe pas une classe juste pour les besoins du moment, bien que ceux-ci orientent nécessairement notre réflexion en période de design ou de développement.

⁴ Nous verrons l'idée d'un tableau 'dynamique' et du conteneur standard 'List' dans une rencontre prochaine

Critères de correction

Ce travail est corrigé sur 100 mais pondéré ensuite selon ce qui est annoncé au plan de cours. **Vous devez le faire seul ou par équipe de deux personnes.** Le barème de correction appliqué à ce programme sera :

- 50% des points pour le bon fonctionnement lors des tests
- 50% des points pour les autres critères
 - respect des demandes
 - bon découpage du problème
 - approche de programmation
 - élégance et esthétisme du code
 - usage de constantes lorsque la situation s'y prête
 - bonne utilisation des concepts de la POO
 - commentaires
 - etc.

Les commentaires devront au moins inclure :

- Un prologue au programme indiquant
 - le nom du ou des auteurs
 - le nom du fichier
 - le nom du projet
 - la date de création
 - la description de ce que fait ce programme
- Un prologue à chaque sous-programme indiquant
 - la description de ce que fait ce sous-programme
 - une description des intrants
 - une description des extrants et valeur de retour
- Chaque partie de code dont la seule lecture n'indique pas clairement la raison d'être devrait également faire l'objet d'un commentaire

Important

Vous devez me fournir lors de la remise votre projet en entier accompagné d'un fichier **Creation.txt** contenant des données qui vous ont permis de tester le bon fonctionnement de votre programme. L'étudiant avisé s'assurera du bon fonctionnement de son programme avec le fichier **Creation.txt** fourni, ce qui

l'assure d'une note meilleure lors des tests de fonctionnement à la correction du travail. Il est à noter que je ne mets pas de contrainte sur le contenu de ce fichier sauf qu'il doit comporter **au moins 2 articles** pour produire une facture.

Bien entendu, il est très prévisible que je mette au point d'autres fichiers de tests pour valider le bon fonctionnement de vos programmes.

Remise

Vous devez remettre une version électronique et une version papier de votre programme. La version papier devra être imprimée en **mode paysage**. Le tout doit naturellement être broché. La date de remise est fixée à la 10^e séance de cours, c'est-à-dire le

Mardi 27 octobre 2020 avant le début du cours

Vous pourrez remettre la version électronique de votre travail en me le faisant parvenir en pièce jointe à un message de courriel à l'adresse **p.prudhomme@gmail.com**, la date et l'heure du message faisant foi du moment de la remise. Prévoyez quelques minutes pour l'acheminement du message.

IMPORTANT : pour me rendre la gestion de tout cela plus facile, je vous demande de nommer votre dossier contenant le projet aux noms et prénoms des équipiers avant de le zipper; ainsi, le travail de Luc Robitaille et Wayne Gretzky me serait expédié en pièce jointe sous le nom RobitailleLuc_GretzkyWayne.zip.

N'oubliez pas non plus de supprimer le dossier caché **.vs** ainsi que les dossiers **bin** et **debug** avant de produire l'archive **sinon la pièce jointe sera détectée par le serveur de courriel comme pouvant contenir un virus et elle ne sera pas acheminée.**

Bon travail.