INF 731 - Travail pratique no. 2

Énoncé du problème

Vous devrez concevoir une application qui permet de représenter virtuellement un convoi ferroviaire, ou si vous préférez un train. Chaque convoi est décrit dans un fichier de données de type texte. Votre application devra lire et interpréter ce fichier pour créer le convoi ferroviaire. La création du convoi s'accompagne de la production d'un « journal des opérations » (fichier *log*) dans lequel vous inscrirez, au fur et à mesure, le résultat de chaque opération effectuée sur le convoi.

L'objectif principal de votre travail consiste à développer un programme pour s'assurer de la validité d'un train. Un train sera valide si sa locomotive est capable de le tracter ou, si vous préférez, si le poids total du train n'est pas supérieur à la capacité de traction de la locomotive et si chacun des wagons du train respecte les limites imposées.

Information sur la réalisation du travail

Le programme que vous devez réaliser doit être écrit en C# en utilisant au mieux les concepts de la programmation orientée objet que nous avons vu et verrons dans le cours.

Il faudra encore lire et écrire dans des fichiers, mais vous devriez maintenant être familier avec cet aspect de la programmation.

Caractéristiques d'un convoi ferroviaire

Un convoi ferroviaire comportera toujours une locomotive en tête de train. Le convoi ne peut comporter qu'une seule locomotive ainsi qu'un nombre inconnu de wagons. Ces wagons pourront être de deux (2) types, soit des wagons de marchandises ou encore des wagons de passagers.

- **Une locomotive** sera définie à l'aide deux (2) données qui seront son poids exprimé en kilogrammes et sa capacité de traction qui s'exprimera en tonnes métriques. Évidemment, une erreur dans les données pourrait faire en sorte qu'on ait une locomotive dont la capacité de traction est négative ou encore inférieure à son poids. Il va de soi qu'une telle locomotive n'aurait aucune chance de pouvoir démarrer, encore moins de tirer un convoi. Un convoi qui aurait une telle locomotive à sa tête ne pourrait pas ajouter de wagons au train, bien entendu. L'ajout d'une telle locomotive à un convoi devrait être refusé et, puisque le train ne peut se constituer sans locomotive...

- **Un wagon de marchandises** possède un poids à vide constant de 2500 kilogrammes et peut contenir jusqu'à 12 000 kilogrammes de marchandises. Si on tente de dépasser cette charge maximum les essieux du wagon cèdent sous le poids des marchandises, le wagon devient inutilisable et il est donc impossible de l'attacher au convoi et de le tracter.

- **Un wagon de passagers** possède un poids à vide constant de 3000 kilogrammes et peut contenir jusqu'à cent (100) passagers. Pour les besoins de notre simulation, on considèrera que les passagers pèsent en moyenne quatre-vingt (80) kilogrammes chacun. Si on tente de dépasser la capacité du wagon en y logeant trop de passagers, il n'y aura pas suffisamment de sièges pour tout le monde ce qui provoquera une émeute durant laquelle un passager radical mettra le wagon hors d'usage. Il sera donc impossible d'attacher un tel wagon au convoi.

Description du fichier contenant les données de création du convoi

Dans le fichier texte contenant les données de création, le convoi est décrit selon le format suivant :

- Sur la toute première ligne, vous trouverez le poids en kilos de la locomotive de tête ainsi que sa capacité de traction exprimée en tonnes métriques. Les deux données seront séparées par des ';' dans le fichier. L'exemple ci-dessous permet de décrire une locomotive dont le poids est 4500 kilos et la capacité de traction de 150 tonnes métriques.

```
4500; 150
```

  Vous avez l'assurance que les deux premières données du fichier de création du convoi sont des entiers qui décrivent la locomotive de tête. De plus, dans un souci de simplification, nous dirons que si la capacité de traction est supérieure ou égale au poids total du convoi, la locomotive peut tirer le train[1].

- Sur les lignes suivantes, vous trouverez la description des opérations (une par ligne) permettant de modifier la composition du convoi ferroviaire. Il y a deux types d'opération acceptée possibles : l'ajout d'un wagon et le retrait d'un groupe de *n* wagons.

[1] Je sais que suite à des expériences, il a été démontré que la résistance suffisante pour que les roues de la locomotive ne patinent pas est égale à 1/6 du poids qu'elles supportent, c'est à dire le poids de la locomotive. Comme il faut que les roues de la locomotive ne patinent pas pour qu'elle avance, on pourrait prendre cette formule pour faire le calcul et dire, par exemple, qu'une locomotive de 10 tonnes métriques a une capacité de traction de 1667 kilos. De même, la force requise pour tirer un wagon à l'horizontale est de 1/233 du poids du wagon. C'est pour éviter d'avoir à traiter ces calculs, au demeurant intéressants, que la simplification est proposée.

- La ligne décrivant l'opération d'ajout peut prendre deux formes selon que l'on désire ajouter un wagon de marchandises ou un wagon de passagers. La ligne commencera par le code d'opération indiquant un ajout de wagon, le code d'opération « **A** » ou « **a** », suivi du type de l'ajout, la lettre « **M** » ou « **m** » signifiant wagon de marchandises ou la lettre « **P** » ou « **p** » signifiant wagon de passagers. Dans le cas où l'on désire ajouter un wagon de marchandises, un entier précisera le poids en kilos des marchandises contenues dans le wagon; dans le cas d'un wagon de passagers, un entier précisera le nombre de passagers. Les données décrivant l'opération seront séparées par des ';'.

  On ajoute toujours le wagon à la fin du convoi, à la suite des wagons déjà présents. S'il n'y a encore aucun wagon, l'ajout se fera (virtuellement) à la suite de la locomotive de tête.

  L'exemple ci-dessous permet de décrire une opération visant à ajouter un wagon de marchandises dont le contenu pèse *10500 kilogrammes*, et une opération visant à ajouter un wagon de passagers contenant *85 personnes*.

```
A;M;10500
A;P;85
```

  Il va de soi que l'ajout d'un wagon ajoute au poids total du convoi à tracter. Ce poids doit toujours être inférieur ou égal à la capacité de traction de la locomotive.

- La ligne décrivant l'opération de retrait d'un groupe de wagons du convoi commencera par le code d'opération « **S** » ou « **s** » et cette lettre sera suivie du nombre de wagons à retirer. On retire toujours les derniers wagons du convoi. L'exemple ci-dessous permet de décrire une opération visant à retirer les *cinq (5)* derniers wagons du convoi.

```
S;5
```

  Vous avez l'assurance que pour chaque ligne décrivant une opération, si la ligne commence par la lettre « **S** », cette lettre sera suivie d'un entier.

- Si la lettre n'est pas un « **S** » ni un « **A** », en minuscule ou majuscule, l'opération est invalide et devra être traitée comme tel. Des précisions sont données plus loin à ce sujet.

Création du journal d'interprétation

Lors du processus d'interprétation du fichier de création du convoi, vous devrez lire chaque ligne du fichier et instancier les différents wagons. Pour chacune des lignes que

vous lirez dans le fichier, vous devrez inscrire dans le « journal » le résultat de l'opération exécutée.

Au début du fichier texte contenant votre journal, vous créerez un en-tête indiquant le nom du fichier décrivant le convoi et les auteurs du programme. Ainsi, si les auteurs sont Sergey Brin, Larry Page, Paul Allen et William Gates III et que le fichier texte contenant le convoi est intitulé «orient-express.txt», l'en-tête devrait prendre la forme suivante :

```
------------------------------------------------------------
JOURNAL D'INTERPRÉTATION – ORIENT-EXPRESS.TXT
Par Sergey BRIN, Larry PAGE, Paul ALLEN, William GATES III
------------------------------------------------------------
```

À la suite de l'en-tête, vous inscrirez, pour chaque opération contenue dans le fichier de création, une indication du résultat de votre interprétation de l'opération. Cette indication doit être suivie du nombre de voitures que contient maintenant le convoi, du poids total du train ainsi que de la capacité de traction de la locomotive exprimée en kilos.

Ainsi, l'exemple ci-dessous permet d'indiquer que l'opération a été un succès, que le convoi comporte maintenant huit (8) voitures, que le poids total du train est de 62500 kilogrammes et que la capacité de traction de la ou des locomotives est de 125000 kilogrammes.

```
Opération acceptée   8    62500     125000
```

L'exemple plus bas permet, quant à lui, de montrer que l'opération demandée dans le fichier de création a été un échec, que le convoi comporte toujours huit (8) voitures, que le poids total du train est encore de 62500 kilogrammes et que la capacité de traction de la ou des locomotives est de 125000 kilogrammes.

```
Opération refusée    8    62500     125000
```

Une fois que toutes les opérations contenues dans le fichier de création du convoi auront été interprétées, vous devrez inscrire une représentation de votre train dans votre journal. Cette représentation indiquera d'abord les caractéristiques de la locomotive puis celles de chacun des wagons du convoi. En terminant, le poids total du convoi sera indiqué.

Par exemple, dans le cas d'un convoi se composant d'une locomotive ayant un poids de 5000 kilogrammes et pouvant tracter 150 tonnes métriques, d'un wagon passager contenant 85 personnes (poids total : 9800 kilogrammes), d'un wagon passager contenant 50 personnes (poids total : 7000 kilogrammes) et d'un wagon de marchandise contenant

10500 kilogrammes de fret (poids total 13000 kilogrammes), vous devriez inscrire à la fin de votre journal d'interprétation la mention suivante :

```
Locomotive[5000,150]->
  WagonPassager[9800]->
  WagonPassager[7000]->
  WagonMarchandise[13000]
POIDS TOTAL DU CONVOI : 34800 kg
```

Consignes relatives à la conception et à l'implantation

1. Lors de l'exécution, votre programme doit demander à l'utilisateur de fournir le nom du fichier contenant les informations relatives à la description du convoi. Dès que le nom du fichier est connu, vous êtes alors capable d'interpréter son contenu et de produire le journal d'interprétation. Le journal d'interprétation doit porter le nom de « **Journal – *nom du fichier original.txt* »** où le nom du fichier original est naturellement le nom de fichier fourni par l'utilisateur.

   Une fois cela terminé, votre programme devra demander un nouveau nom de fichier et recommencer le traitement tant et aussi longtemps que l'usager n'entrera pas un nom de fichier vide.

2. Gestion des situations d'erreur :
   a. Si vous déterminez qu'une opération (ajout – retrait) est incorrecte ou que l'appliquer rendrait le convoi ferroviaire invalide, vous devez ignorer cette opération mais continuer à interpréter le fichier des données. Utilisez à bon escient la notion d'exception. Indiquez dans un autre fichier de sortie (peut-être *erreurs.log*) la ligne en erreur afin qu'il soit possible de savoir exactement quelles lignes ont été rejetées à la fin de l'exécution.
   b. Dans le cas où vous déterminez que les données de la locomotive de tête sont invalides, vous n'avez pas à lire et interpréter la suite du fichier, vous devez seulement indiquer un message d'erreur dans votre journal d'interprétation en plus d'y inscrire l'en-tête.
   c. Une opération d'ajout peut être invalide dans deux cas :
      i. le poids des marchandises ou le nombre de passagers dépasse la capacité du wagon ;
      ii. l'ajout du wagon au convoi fait en sorte que le poids total du train dépasse la capacité de traction de la locomotive;

d. Une opération de retrait est invalide s'il n'est pas possible de la réaliser en totalité. Ainsi, dans le cas où on demande de retirer 5 wagons[2], d'un train qui n'en contient que 4 vous devez refuser l'ensemble de l'opération et laisser le train intact. On ne réalise pas partiellement une opération de retrait.

Contexte de réalisation

Ce programme est à réaliser en tirant au maximum profit des possibilités de la programmation orientée objet. **Pour ce travail, il s'agira de former des équipes de développement, chaque équipe ayant 4, 5 ou 6 étudiants[3],** cette flexibilité étant rendue nécessaire par le nombre d'étudiants de la cohorte. Il n'y a pas d'exception à cette règle. Le but visé est de vous amener à travailler sous des contraintes qui ressemblent à celles avec lesquelles un chef de projet et une équipe de développement doivent composer.

Il est clair que ce travail, surtout exécuté dans un tel contexte, présente une *face cachée* importante et que vous aurez à prendre des décisions à plusieurs niveaux. D'abord, au niveau organisationnel, déterminez au point de départ qui sera considéré comme ***l'intégrateur*** de votre équipe[4]. Ce dernier aura pour rôle de prendre les devants quand des problèmes à résoudre se présenteront. Les autres membres de l'équipe, qui devront interagir avec l'intégrateur et lui fournir leur partie du travail, devront quant à eux demeurer critiques quant à la manière dont se déroule ce processus. Vous êtes dans l'action mais vous devez également conserver une perspective sur l'action.

Chaque équipe devra me faire connaître sa **composition ainsi que son intégrateur d'ici une semaine.** L'intégrateur sera entre autre responsable de m'informer si l'évolution des travaux dans l'équipe ne se passe pas comme prévu. L'intégrateur ne peut être tenu responsable du résultat pour l'ensemble de l'équipe comme le serait un chef de projet. Toutefois, il a la responsabilité de me faire connaître rapidement les difficultés majeures rencontrées par son équipe qui pourraient empêcher de produire le résultat attendu dans le délai requis. ***Si une équipe n'arrive pas à nommer son intégrateur, je serai dans l'obligation d'en désigner un***[5].

De plus, il y aura sans doute des décisions à prendre quant à la manière de structurer les données et de constituer votre hiérarchie de classes. L'énoncé vous laisse de la latitude mais certains éléments de l'énoncé devraient orienter quelques-unes de vos décisions. De

[2] On s'entend qu'on ne peut jamais retirer la locomotive de tête d'un convoi ☺

[3] À ce moment-ci, je crois que nous sommes 10 ou 11 étudiants dans le groupe ce qui permettrait de créer une équipe de 5 étudiants et une équipe de 6 étudiants.

[4] Les travaux académiques présentent un cadre qui s'adapte mal à la notion de 'chargé de projet' où une personne est responsable d'une équipe de développement. Il serait donc inapproprié de vous nommer un 'chef de projet'. Toutefois, l'expérience démontre qu'il doit y avoir quelqu'un qui se charge de lier ensemble les diverses composantes que livrent les sous équipes, d'où la nécessité d'avoir un intégrateur du projet.

[5] Jusqu'à présent la chose n'est pas arrivée et j'espère ne pas avoir un précédent cette session-ci.

plus, des classes pourraient vous être utiles qui ne sont pas nécessairement décrites noir sur blanc dans l'énoncé du problème. Ceci est tout à fait normal dans le cadre d'un développement informatique : le client ne vous donne pratiquement jamais la description de son problème en fonction de la solution à venir puisqu'il n'y connaît généralement rien en informatique. En cas de doute, vous pouvez envoyer un courriel au client qui vous donnera les précisions requises[6].

Plus encore que cette *face cachée*, vous devez chercher à tenir compte des éléments que votre client bien souvent ignore ou omet involontairement. Par exemple, vous devez tenir compte de *l'avenir*, c'est à dire essayer d'élaborer votre design en fonction de ce qui vous sera - sans doute - demandé à l'étape suivante par votre client.

Ainsi, comment votre système s'adapte-t-il à la réalité d'une plus grande variété de wagons ? De la possibilité d'avoir plusieurs locomotives plutôt qu'une seule. Que la ou les locomotives comportent différentes caractéristiques ? De l'intégration d'un calcul plus réaliste de la capacité de traction – voir la note en bas de page à ce sujet ? Votre hiérarchie de classes et la structure de votre code devrait offrir assez de souplesse pour faciliter les ajouts et la réutilisation.

Réfléchissez à chaque classe importante de votre système, mettant l'accent sur ce que la classe en question doit être en mesure de faire. Réfléchissez aux cas où l'héritage et le polymorphisme peuvent s'avérer des outils intéressants à exploiter.

Pensez en termes de méthodes plutôt que d'attributs; ce sont surtout les membres publics d'une classe qui doivent survivre au passage du temps; ce sont ceux-là qui sont les plus importants au sens opérationnel. Ils forment l'interface de la classe, qui est un contrat que vous vous engagez à respecter formellement auprès des utilisateurs de la classe.

Appliquez strictement et systématiquement le principe d'encapsulation. Ceci vous amènera des économies importantes à moyen et à long terme, même si la tentation est parfois forte en début de développement de "*tourner les coins ronds*"… on finit toujours par en payer le prix.

Assurez-vous d'avoir bien analysé le travail à faire avant de commencer la programmation du système. Certains détails en apparence banals peuvent avoir des conséquences à long terme sur votre design. Vous devriez avoir le plan d'attaque le plus complet possible dès les premières étapes afin de ne pas vous retrouver avec une surprise plus tard dans l'élaboration du système. De telles erreurs de conception coûtent cher en temps et en ressources. Quand elles sont découvertes trop tard, elles peuvent même faire en sorte que le projet n'aboutisse jamais.

Comme dans un véritable projet de développement, vous pouvez poser des questions d'éclaircissement au client lorsque des passages de sa demande ne sont pas clairs[7].

Finalement, un énoncé comme celui-ci qui vous accorde une liberté d'action demande plus de réflexion de la part de l'équipe de développement. *La réussite (ou l'échec) d'un projet de développement dépend souvent de la qualité de l'analyse et du design initial; la*

[6] Dans le cadre du TP, votre humble serviteur joue aussi le rôle du client.

[7] Voir note précédente pour connaître l'identité du client.

livraison à temps dépend quant à elle souvent d'une subdivision efficace des tâches, d'où l'importance du rôle de l'intégrateur qui veille au grain.

Remise et correction

Vous devrez remettre, **sous forme électronique seulement**, les documents suivants :

- une version zippée du projet entier afin que je puisse le compiler et l'exécuter. Le fichier contenant la version zippé portera un nom composé des noms des équipiers afin de me faciliter la gestion des pièces reçues.

- inclure aussi des fichiers de tests permettant de vérifier le bon fonctionnement de votre application. Indiquez dans le journal de développement le nom de ces fichiers afin d'aider le correcteur à faire son travail plus aisément.

Vous devrez remettre, **sous forme papier et sous forme électronique**, le document suivant :

- un **journal de développement** de votre projet qui pourra comporter :
    - D'un point de vue technique :
        - un ou des schémas UML représentant vos diverses classes et les liens entre elles;
        - les fonctionnalités supplémentaires développées, s'il y en a;
        - les "bugs" connus de votre programme, s'il en reste;
        - le mode de fonctionnement du programme, s'il y a des particularités à remarquer;
        - tout autre élément technique pouvant valoir la peine d'être mentionné.
    - D'un point de vue processus de l'équipe :
        - Comment l'intégrateur a-t-il été désigné ?
        - Comment le leadership a-t-il été assumé dans l'équipe ?
        - Quelles ont été les sources de tensions et les conflits à régler ?
        - Quels ont été les éléments positifs de la démarche ?
        - Suggestions de choses à faire et à ne pas faire
        - Bilan et conclusions de cette expérience
    - Ce guide doit être écrit en format MS-Word.

La version électronique de ces documents peut me parvenir par courriel ou vous pouvez me remettre un CD contenant le tout.

Choisissez une police de caractères non proportionnelle (quelque chose comme Courier New ou `Lucida Console` par exemple) pour l'impression de votre programme. Ces polices conviennent bien à l'impression du code, clarifiant l'indentation appliquée.

Chaque étudiant doit remettre individuellement une version papier de son évaluation de ses pairs. Cette évaluation consiste pour chaque étudiant à évaluer **sa contribution** ainsi que **la contribution de chaque membre de son équipe** à la bonne marche du projet. Pour chacun, l'évaluation que vous faites doit suivre le barème suivant :

Note	Signification
A	Contribution exceptionnelle. Le projet aurait échoué sans cet apport remarquable.
B	Contribution généralement à la hauteur des attentes. La grande majorité des étudiants devraient avoir cette cote.
C	Contribution en deçà de ce qui était attendu ou de ce qu'il s'était engagé à livrer, quoique l'effort et la volonté y étaient. Il se peut même que certains livrables aient dû être pris en charge par d'autres étudiants de l'équipe.
D	Contribution minimale à la bonne marche du projet. L'étudiant a très peu contribué au produit final. Sa manière d'agir peut même avoir nui à la bonne marche du projet. Il devrait être *exceptionnel* d'attribuer un D à un membre de l'équipe.

L'évaluation de l'intégrateur de ses pairs sera davantage considérée en tenant compte de son rôle dans le projet. **L'intégrateur s'assurera d'avoir reçu, sous enveloppe cachetée[8], les évaluations de tous les membres de l'équipe afin de les remettre[9].**

8 Ces évaluations sont confidentielles.

9 S'il ne reçoit pas toutes les évaluations, l'intégrateur me le signale au moment de la remise finale afin que je puisse intervenir.

Le barème de correction qui sera appliqué à vos projets sera le suivant :

- 40 % pour le bon fonctionnement lors de tests, évalué selon la grille suivante :

Fonctionnement remarquable de qualité supérieure, travail professionnel, rien à redire	40
Très bon fonctionnement, travail d'excellente qualité	37
Bon fonctionnement en général, travail de bonne qualité	33
Fonctionnement acceptable mais il y a des problèmes encore à régler	30
Fonctionnement qui laisse beaucoup à désirer, résultats partiels, difficile à utiliser	25
Fonctionne faiblement, beaucoup de lacunes, résultats fragmentaires, peut même planter dans certaines circonstances	20
Fonctionnement très faible, peu de choses ont été complétées, produit un quelconque résultat; le programme peut même planter beaucoup	14
Le programme remis compile	8
Le programme remis ne compile pas	0

- 40 % pour la réalisation du code qui évaluera

• L'utilisation de l'héritage d'implémentation	4
• L'utilisation de l'héritage d'interface	4
• L'utilisation judicieuse de l'abstraction	4
• L'utilisation d'une ou de plusieurs collections	4
• Les choix algorithmiques, le découpage du problème, la conception de votre hiérarchie de classes, etc.	8
• L'utilisation judicieuse des notions vues durant la session : propriétés, qualificatifs d'accès, constructeurs, exceptions, lambdas, ToString, surcharge d'opérateurs, etc.	8
• La qualité générale de la programmation	8

- 10 % pour l'évaluation par les pairs
- 10 % pour le journal du développement.

Échéances

Dès que possible mais au plus tard d'ici une semaine, chaque équipe doit me faire connaître par écrit (à la prochaine séance) ou par courriel sa composition ainsi que son intégrateur. C'est l'intégrateur qui se charge de faire connaître son équipe de moi et m'indique le nom des étudiants qui en font partie.

L'ensemble du projet ainsi que le journal de développement doivent être remis au début de la dernière séance de cours le **mardi 18 avril 2017** au début de la séance à **18h45**[10], avant que ne commence l'examen final.

Aucun retard ne sera toléré puisqu'il s'agit bien évidemment de notre dernière rencontre de la session.

Bon travail.

Pierre P.

[10] Soyez ponctuel.