

Projet Programmation Orientée Objets 2013-2014

Informations générales

Ce projet est à effectuer par groupe de 2 étudiants. Un groupe de 3 pourra être formé en cas de problème de parité. Vous disposez des 4 séances de 2h40 avec votre encadrant pour travailler sur ce projet.

Contexte

Le service événementiel de la patinoire de Mériadeck a subi de gros problèmes informatiques récemment et a perdu toute son installation informatique. Nous allons les aider à surmonter cela en leur proposant une application simple et légère qui va leur permettre de temporairement gérer les manifestations ainsi que les réservations.

Nous distinguons 3 types de manifestations, et pour chacune nous allons devoir stocker un ensemble d'informations :

- Les rencontres de Hockey sur glace
 - Deux équipes
 - Un arbitre
 - Une compétition (Championnat, Coupe de France)
 - Une date
 - Une heure de début
 - Une heure de fin
 - Le prix d'une place
 - Le nombre total de places en vente
 - Un numéro d'évènement
- Les concerts de musique
 - Le nom d'un artiste
 - Un style de musique
 - Une date
 - Une heure de début
 - Une heure de fin
 - Le prix d'une place
 - Le nombre total de places en vente
 - Un numéro d'évènement
- Les créneaux de patinage "libre"
 - Une date
 - Une heure de début
 - Une heure de fin
 - Le prix d'une place

Par chance, le service informatique a pu restaurer sous forme de fichier CSV un extrait d'une liste de manifestations (Ce format n'est pas optimal comme vous pouvez le constater) :

```
Ref;Type;Date;HeureDebut;HeureFin;Prix;Places;Artiste;Style;Equipe1;Equipe2;Competeur;Arbitre
1;Hockey;25/10/13;17h00;20h00;5;2000;;;Bordeaux;Amiens;Championnat;Ricardo
2;Concert;26/10/13;21h00;23h30;50;3000;Dave;Chanson Française;;;;;
;Patinage;27/10/13;18h00;23h30;10;;;;;;;
```

Un exemple de fichier CSV est fourni dans l'archive du projet.

Travail demandé

Mise en place d'une base de données en XML

Le format CSV pour stocker les informations n'est pas très pratique. Nous allons stocker les manifestations ainsi que les réservations sous format XML dans un même fichier, nommé `db.xml`. La structure de ce document XML vous est laissée libre.

Votre programme devra être capable de prendre en entrée les deux fichiers CSV dans les formats décrits ci-dessus, et de produire en sortie un fichier XML : `db.xml` comportant les mêmes informations dans le format XML que vous avez élaboré. Pour réaliser cette opération vous devez écrire une classe `PM` qui prend des paramètres en ligne de commande du shell. Voici la ligne de commande qui permet de réaliser l'opération décrite ci-dessus :

```
java cli.PM importer /chemin/vers/fichier_manifestations.csv /chemin/vers/db.xml
```

Le code ci-dessus signifie que la ligne de commande est programmée à l'aide du classe `PM` qui se trouve dans un package `cli`. La réalisation de cette classe est à votre charge, elle doit réaliser les commandes qui sont décrites dans les différentes sections du projet. Vous devez aussi assurer le traitement et report des erreurs concernant les paramètres et les noms de fichiers.

Ajout de réservations

On doit pouvoir rajouter des réservations exprimées dans un fichier CSV qui contient les informations suivantes :

- Référence Client
- Référence de l'évènement
- Nom
- Prénom
- Âge
- Réduction (ex : -20%)

Ces réservations **doivent être elles aussi stockées dans le fichier `db.xml`**. Ce fichier contient plusieurs demandes de réservations qui doivent être traitées ligne par ligne. Pour qu'une réservation soit correcte il faut que :

- Que la manifestation concernée existe bien (la référence est correcte) : erreur de type R1
- Que sa date n'est pas antérieure à la date actuelle : erreur de type R2

- Que le nombre actuel de réservations ne dépasse pas le maximum autorisé : erreur de type R3

Si tout est bon, on ajoute la réservation à la liste, et donc on met à jour db.xml. Si ce n'est pas le cas on écrit sur la sortie standard le code de l'erreur suivi d'une tabulation puis de la ligne qui a provoqué l'erreur. On passe ensuite à la ligne suivante.

Voici un exemple de fichier CSV pour les réservations, fourni dans l'archive du projet :

```
Ref_evt;Ref_client;Nom;Prenom;Age;Reduction
1;1;Robert;Martin;45;0
2;2;John;Doe;20;-20%
```

Voici la ligne de commande qui permet de réaliser l'opération décrite ci-dessus :

```
java cli.PM reserver /chemin/vers/fichier_reservations.csv /chemin/vers/db.xml
```

Ajout de manifestations

A partir d'une ou plusieurs demandes de créations d'évènement exprimées dans un fichier CSV, il faut mettre à jour le fichier db.xml. Pour qu'une manifestation soit valide il faut que :

- La référence de la manifestation n'existe pas déjà : erreur de type M1
- La date et l'heure soient valides (le jour n'existe pas, l'heure non plus) : erreur de type M2
- Le créneau ne soit pas déjà réservé, on précisera l'évènement en question dans le message d'erreur : erreur de type M3
- Le prix de la place soit valide (cela doit être un entier positif ou égal à 0) : erreur de type M4
- L'artiste ou le nom des équipes de hockey sont manquantes : erreur de type M5

Si tout est bon, on ajoute la manifestation à la liste, et donc on met à jour db.xml. Si ce n'est pas le cas on écrit sur la sortie standard le code de l'erreur suivi d'une tabulation puis de la ligne qui a provoqué l'erreur. On passe ensuite à la ligne suivante.

Voici un exemple de fichier CSV pour les concerts:

```
Ref;Date;HeureDebut;HeureFin;Prix;Places;Artiste;Style
342;26/10/13;21h00;23h30;50;3000;Dave;Chanson Française
631;30/10/13;21h00;23h30;50;3000;Britney Spears;Pop
```

Voici un exemple de fichier CSV pour les matchs de hockey :

```
Ref;Date;HeureDebut;HeureFin;Prix;Places;Equipe1;Equipe2;Compet;Arbitre
432;25/10/13;17h00;20h00;5;2000;Bordeaux;Amiens;Championnat;Robert Ricardo
136;30/10/13;17h00;20h00;5;2000;Bordeaux;Paris;Championnat;Bob Burnquist
```

Les deux exemples précédents sont fournis dans l'archive du projet. Voici les lignes de commandes qui permettent de réaliser les opérations décrites ci-dessus :

```
java cli.PM manifestation concert /chemin/vers/fichier_concerts.csv
chemin/vers/db.xml
```

```
java cli.PM manifestation hockey /chemin/vers/fichier_hockey.csv  
/chemin/vers/db.xml
```

État des réservations

Le service événementiel aimerait être informé des réservations actuelles pour l'ensemble des manifestations. Pour l'aider à avoir un aperçu, nous vous demandons de produire un rapport sous format HTML qui doit faire apparaître les informations suivantes :

- La liste des manifestations, avec pour chacune :
 - Les informations de base sur la manifestation
 - Le nombre de réservations en cours
 - Le nombre de places encore disponibles
 - La recette prévisionnelle (en prenant en compte les réductions!)
 - La répartition des réservations par tranche d'âge [<25][25-40][>40]
- La recette prévisionnelle totale
- La recette prévisionnelle par type de manifestation, et le pourcentage que cela représente par rapport au total

Attention, ces informations sont calculées pour les **manifestations qui sont postérieures à la date actuelle**.

Voici les ligne de commandes qui permettent de réaliser les opérations décrites ci-dessus :

```
java cli.PM rapport /chemin/vers/db.xml /chemin/vers/rapport.html
```

L'exécution de votre programme doit permettre la génération d'un tel rapport au format. Nous attendons que les informations souhaitées apparaissent, le style graphique du rapport est laissé libre.

Bonus

Si vous avez fini tous les points de ce projet, vous pouvez passer à l'étape suivante : on aimerait intégrer un back-end base de données relationnelle type MySQL pour persister les manifestations et réservations. Pour l'instant notre base de données est constituée d'un fichier XML, et on aimerait offrir la possibilité de stocker sous les deux formats. Attention ! On ne veut pas altérer les fonctionnalités du programme.

Le système de données utilisé (base de données ou XML) doit être spécifié au lancement du programme. Il vous faudra modifier l'architecture de classes pour rendre **transparent** le type de système utilisé. C'est donc principalement ici un exercice de conception. On pensera donc à créer une interface `AccessDonnees` pour uniformiser l'accès et la mise à jour des données. La priorité ici sera d'avoir une architecture fonctionnelle prête à accueillir l'écriture de code Java qui utilise du SQL, etc... Un ensemble fonctionnel à 100% sera forcément un plus.

Mise en oeuvre

Introduction à XML

Afin de devenir plus familier avec le langage de représentations de données XML, il vous sera demandé dans un premier temps de prendre connaissances des liens suivants :

<http://haypo.developpez.com/tutoriel/xml/introduction/>

<http://erwy.developpez.com/tutoriels/xml/modelisation-xml-choix-structures-pour-representation-donnees/>

Librairies Tierces

Pour vous aider dans la partie lecture/écriture de fichiers CSV et XML, nous allons utiliser ce que l'on nomme couramment des librairies tierces. Ce sont des ensembles de classes testées et fonctionnelles qui ont pour vocation d'être ré-utilisées. Le principal objectif pour des personnes comme nous est d'éviter de perdre du temps à ré-inventer la roue. Comme cette pratique est courante dans le développement logiciel et notamment en Java, nous allons vous faire découvrir un peu cet aspect.

Ainsi, pour la partie XML il faut utiliser la librairie JDOM (<http://www.jdom.org>), et pour la partie CSV la librairie OpenCSV (<http://opencsv.sourceforge.net/>). A vous de vous renseigner sur les documentations fournis par les sites internet, et des résultats fournis par Google.

Voici une liste de ressources concernant ces librairies :

- <http://javadevtips.blogspot.fr/2011/10/opencsv-easy-csv-parser.html> (opencsv)
- <http://www.mkyong.com/java/how-to-read-xml-file-in-java-jdom-example/> (jdom)

Architecture

La réalisation de ce projet va vous permettre de définir un ensemble de classes et de méthodes. Une attention sera portée sur la pertinence de vos choix. Lors de la soutenance finale, vous devrez être capable de justifier vos choix. Gardez bien en tête que le programme que vous allez écrire doit être en mesure d'évoluer (si jamais de nouveaux types de spectacles sont intégrés, ou par exemple différents types de clients, ou même de salles). Essayer donc de définir une architecture de classes qui facilitera des futures opérations de maintenance.

Livrables

Il sera demandé aux étudiants de fournir une archive contenant :

- Le code source du programme dans un dossier src
- Un rapport nommé README, à la racine de l'archive, de maximum 2 pages qui présente l'architecture de votre programme, la manière dont vous avez implémenté les différentes fonctionnalités, ainsi que les difficultés que vous rencontrées.

Le schéma de nommage de l'archive **doit** être le suivant (pénalité de 1 point en cas de non respect) : pg220-2013-LOGINS.zip, avec LOGINS à remplacer par la liste (classée par ordre alphabétique) des logins des membres du groupe séparés par des underscore (_).

L'archive est à envoyer par mail à l'adresse falleri@labri.fr **avant le 13 Janvier 2013 à minuit.**

Vous devez absolument respecter les contraintes suivantes. Une pénalité de 1 points sera appliquée sur chaque projet qui ne les respecte pas.

- Votre projet doit contenir un ou plusieurs packages
- Les noms de packages sont intégralement en minuscules
- Les noms de classes et interfaces doivent commencer par une majuscule
- Les noms de méthodes, d'attributs et de variables doivent commencer par une minuscule
- Utilisez le camel case pour les noms des classes, interfaces, méthodes et attributs.
- Les attributs doivent avoir une visibilité. Par défaut, les attributs sont *private*. Des méthodes de type *get/set* doivent permettre de gérer l'accès depuis l'extérieur à ces variables. Attention, définissez ces méthodes pour chaque attribut, seulement si elles sont utiles.
- Définissez des constructeurs que lorsqu'ils sont nécessaires.
- Commentez au maximum votre code. Expliquez ce que font vos méthodes quand elles sont complexes. N'expliquez pas tout et n'importe quoi. Soyez concis et allez à l'essentiel, indiquez ce qui est réellement utile à la compréhension.

Annexe

Utilisation des librairies externes

Récupérez les fichiers JAR des librairies nécessaires. Notez que le format JAR est un format de type "Archive" pour du code java compilé.

Pour intégrer les librairies dans votre projet, suivez la procédure suivante :

1. Créer un dossier lib à la racine de votre projet (clic droit sur le projet, new-> folder)
2. Copier/Coller les fichiers JAR des librairies, faites un F5 pour rafraîchir l'affichage
3. Clic droit sur les fichiers JAR -> Build Path -> Add to Build Path