PG220 : Programmation Orientée Objet

Aide au service événementiel de la patinoire de Mériadeck

Claire SAUVAGE (∼csauvage)
Martin LADURE (∼mladure)
Thomas MAILLARD (∼tmaillard)

Dans ce court rapport, nous allons présenter l'architecture, le fonctionnement global et l'implémentation de notre projet de programmation JAVA.

1 Architecture

Dans notre projet, nous avons séparé chaque type d'événement en associant une classe à chacun d'eux.
Notre projet comporte un seul package(*cli*), qui contient toutes les classes du programme. On appelle la classe mère *PM.java*, qui redirigera (en fonction des arguments) la classe à exécuter.
Détail et utilité de chaque fichier :

- Client.java : Classe qui permet d'implémenter la structure pour un client.
- Concert.java : Classe qui permet d'implémenter la structure pour un concert.
- Evenement.java : Classe qui permet en fonction des arguments transmis, d'ajouter au fichier XML des nouveaux évènements.
- Hockey.java : Classe qui permet d'implémenter la structure pour un match de hockey.
- Importer.java : Classe qui permet de parcourir le fichier CSV donné en argument, et de créer le fichier XML, dont le nom est également donné en argument.
- Manifestation.java: Classe qui permet l'ajout de tout type de manifestation. En fonction du deuxième argument, cela permet de choisir la bonne classe en fonction du type d'évènement.

- Patinage.java : Classe qui permet d'implémenter la structure pour un évènement de type Patinage.
- PM.java : Classe mère qui permet de sélectionner la classe à appliquer en fonction des arguments.
- Rapporter.java : Classe qui permet de parcourir le fichier XML donné en argument, et de générer un rapport en HTML, plus lisible, pour afficher les données du fichiers XML et aussi quelques statistiques.
- Resa.java : Classe qui permet d'implémenter la structure pour une réservation.
- Reserver.java : Classe qui permet de gérer les réservations. Elle prends en entrée le CSV ainsi que le fichier XML.

2 Gestion des Erreurs :

Pour la gestions des erreurs dans les fichiers réservations et ajout des événements, nous avons opté pour une recherche des erreurs de manière récursive. On parcourt le fichier XML à la recherche de l'existence d'un ou plusieurs doublons. Dès qu'un tel cas est détectée, on arrête la recherche, puis on ajoute le message d'erreur dans une *list*. Avant l'ajout dans le XML on vérifie le nombre d'erreur.

3 Difficultés rencontrées :

La compréhension du sujet fut difficile, car il y avait de nombreuses classes à implémenter. Nous avons dû réaliser une architecture pour coder le projet. Nous avons éprouvé, de grandes difficultés sur la gestions des erreurs. En effet, nous avons rencontré des problèmes concernant la vérification des erreurs. De plus l'implémentation des dates, qui nous permettait de gérer également les erreurs nous a demandé beaucoup de temps. Finalement, nous utilisons un format simple du type *DD/MM/YY*.
Toutefois, nous avons été ravis de pouvoir appréhender les bases de ce langage très utilisé aujourd'hui. De plus un point positif est que nous avons très correctement géré le partage en SVN de notre travail de groupe.