TP Gestion Agendas

## 1/ Diagrammes de classes

Ce schéma représente le diagramme de classe initialement prévu à la suite de la lecture du cas. Nous nous sommes basé sur ce dernier pour commencer le travail et pour nous attribuer les tâches.

Il a ensuite évolué et s’est complexifié au fur et à mesure de l’avancement du TP, c’est pourquoi nous joignons aussi le diagramme de classe final.

Macintosh HD:Users:thomasgaillard:Desktop:DiagrammeDeClasse.pdf

Voici donc le diagramme de classe final, obtenu par le procédé inverse : une génération automatique depuis le code source. Nous avons pu remarquer qu’il est très difficile de tout prévoir en amont dans une phase de conception isolée.

## Macintosh HD:Users:thomasgaillard:Desktop:diagrammeClasse.png2/ Explications

**Environnement et technique de développement**

Nous avons utilisé l’IDE Eclipse pour ce projet et nous avons mis en place dès le début un SVN : Git, utilisé grâce au client lourd officiel du service d’hébergement Github. (<https://github.com/thomasgaillard/gestionAgendas>)

Ceci nous a pris un certain temps au départ pour nous organiser mais nous avons pu en gagner tout au long du projet en travaillant plus efficacement à deux sans échanges récurrents de fichiers pour avoir une version commune.

Nous avons pu découper les tâches et nous les répartir grâce au diagramme de classes initial.

**Choix de conception**

Nous avons choisi de concevoir le programme avec 3 classes « principales », c’est à dire la classe Evenement qui représente un événement avec toutes ses informations. Ensuite la classe Agenda qui est une composition d’événements et qui permet de regrouper les événements à travers plusieurs agendas pour les différencier. Le tout compose une classe Calendrier qui contient donc les différents agendas. On a également choisi de mettre à part la classe App.

Nous avons défini qu’un événement avait un attribut agenda (il connaît donc son agenda) car on devait pouvoir changer l’événement d’agenda et cela imposait alors que l’événement sache dans quel agenda il est contenu. Mais on avait aussi besoin d’une liste d’événements dans Agenda pour pouvoir réaliser les opérations de tri et de filtrage notamment.

Pour la gestion des événements en mode graphique nous utilisons les événements et les « listeners ». Quand l’utilisateur va cliquer, le listener va appeler une fonction qui exécutera le code nécessaire.

**IHM – lancement du programme**

Pour lancer le programme il faut exécuter la classe App. Il est également possible de créer une instance de cette même classe.

À son lancement, le programme va récupérer les fichiers ics Pro et Perso afin de restaurer les agendas déjà créés. On va donc avoir une sauvegarde des événements puisqu’à la fermeture de l’application il y aura un export des agendas.

Il y a également la possibilité de lancer la classe Test afin de lancer divers tests sur chacune des méthodes pour avoir des résultats en Console et pas sur une interface graphique.

**Fonctionnalités et limites**

Nous avons pu réaliser l’ensemble des fonctionnalités des parties 1 et 2 : les fonctionnalités de bases, l’import et export de fichiers « .ics» ainsi qu’une interface graphique.

La recherche ou le filtre des évènements peut être réalisé en fonction du nom, du lieu, de la date et de l’agenda.

Pour les tris nous avons utilisé différentes classes de comparaison pour utiliser la méthode « sort » d’une collection en fonction de multiples critères. (Redéfinition de la méthode « compare »).

Concernant l’export de fichiers « .ics», nous avons regardé la structure d’un tel fichier et nous avons généré un fichier par agenda de notre calendrier en respectant cette structure pour nos différents évènements.

L’import, de la même façon, va parcourir un fichier « .ics » pour enregistrer différents événements d’un même agenda dans notre calendrier.

## 3/ Annexes - Programmes

Voir les fichiers sources joints pour plus de lisibilité.