#### PARTIE R Xavier

#### VOINIER Bruno 14/06/2015

**LO22 – P15 – ProjectCalendar**

Agenda électronique/Outil de gestion de projet

# INTRODUCTION

Le but de ce projet, réalisé dans le cadre de l’UV Lo21, est de développer une application permettant de mixer les fonctionnalités d’un agenda électronique (affichage dans un calendrier dynamique des différents évènements programmés dans la semaine), et celles d’un outil de gestion de projet (gestion, et création de projets à partir d’un ensemble de tâches de différents type). Cette application a été développée en C++, avec le framework Qt.

Dans ce rapport sera expliqué dans un premier temps le choix de l’architecture mise en place pour ce projet, puis il sera montré le caractère évolutif de notre architecture.

**LO22 – P15 – ProjectCalendar**

# Description de l’architecture

Afin de respecter les nombreuses fonctionnalités que doit posséder l’application, nous avons opté pour une architecture qui comme l’application est séparée en deux grandes parties (cf UML en annexe). Une permettant la gestion des projets, une autre partie concernant les évènements. La composition entre *tacheUnitaire* et *programmationTache* permet de faire la liaison entre ces deux parties.

### Partie gestion de projet :

Dans cette partie nous avons les classes *Projet*, *ProjetManager*, *Tache*, *TacheUnitaire*, *TacheComposite*, et *TacheManager*

Projet est une classe permettant de définir les caractéristiques d’un projet (titre, date de disponibilité, date d’échéance, et liste de taches). Elle s’occupe donc des taches qui la composent (ajout, suppression, recherche, etc…). Elle peut également mettre à jour ses différentes caractéristiques. Et elle comporte également une méthode qui lui permet d’afficher ses différentes caractéristiques. Cette classe est gérée par la classe *ProjetManager* (classe composée).

ProjetManager est donc une classe qui permet de gérer les différents projets de l’application. Elle a comme attribut une liste de projets et une instanceUnique sur elle-même. Cette classe permet donc d’ajouter, supprimer un projet de la liste, mais également de chercher un projet dans la liste. Elle possède également une méthode qui permet de retourner cette instance unique. On a défini au sein de cette class une classe Iterator qui permet de parcourir la liste des projets. Cette classe permet d’éviter d’avoir deux projets qui portent le même nom.

Tâche est une classe composante de la classe *Projet*. Elle permet de définir les caractéristiques d’une tache (identificateur, titre, dates de disponibilité et d’échéance). Cette classe est gérée par la classe *TacheManager*, et, est donc une classe composante de cette dernière. Nous avons décidé d’implémenter cette classe comme étant abstraite, car une tache peut être soit unitaire soit composite. Elle est donc héritée par les classes *TacheUnitaire* et *TacheComposite*. Cette classe permet d’avoir accès aux différents attributs et de les modifier facilement. Elle comporte également une méthode virtuelle pure qui permet d’afficher les caractéristiques de la tâche.

**LO22 – P15 – ProjectCalendar**

TacheUnitaire hérite donc de la classe *Tache*, et comporte deux attribut supplémentaire, la durée de la tâche est si la tâche est préemptable ou non. Ses méthodes permettent également d’avoir accès à ces nouveaux attributs et de les modifier. Elle définit également la méthode permettant d’afficher une tache.

TacheComposite est similaire à la classe *TacheUnitaire*, avec comme attribut supplémentaire une liste de taches qui représente la listes des taches composantes de la tache composite. Elle comporte les mêmes méthodes que la classe *TacheUnitaire* (mais adapté à ses attributs). Elle possède cependant deux constructeurs, un permettant de créer une tache composite sans tache composante, et un permettant de la créer avec ses tâches composantes.

TacheManager est donc une classe composé de la classe *Tache*. Elle permet ainsi de gérer celle-ci. Elle comme attribut un tableau d’objets de la classe Tache, mais également une instance unique sur elle-même. Elle permet, à l’image de la classe *ProjetManager*, d’ajouter, supprimer et récupérer des tâches dans le tableau de tâches. Elle peut également récupérer cette instance unique. Au sein de cette classe nous avons également implémenté une classe Iterator qui permet de parcourir l’ensemble des tâches.Par ailleurs, cette classe permet d’éviter d’avoir deux tâches ayant le même identifiant dans le tableau.

Un évènement pouvant être soit la programmation d’une activité, soit la programmation d’une tache unitaire, la classe *TacheUnitaire* permet de faire la liaison entre la partie gestion de projet, et la partie gestion d’évènements. La classe *TacheUnitaire* est donc une classe composante de la classe *ProgrammationActivité*.

### Partie gestion d’évènement :

Dans cette partie nous avons les classes *Evenement*, *EvenementManager*, *ProgrammationTache*, *ProgrammationActivite*, *EvtPj*, et *Evt1jDur*.

Evenement est une classe permettant de définir les caractéristiques d’un évènement (date, horaire de début, et horaire de fin). Nous avons choisi de rendre cette classe abstraite puisqu’un évènement est défini comme étant soit la programmation d’une activité soit la programmation d’une tache. Elle est donc héritée par les classes *ProgrammationTache* et *ProgrammationActivite*. Elle contient des méthodes permettant d’accéder à ses différents attributs, mais également une méthode qui retourne le type de l’évènement (tâche ou activité), cette méthode est virtuelle pure et rend ainsi la classe abstraite. Cette classe est gérée par la classe *EvenementManager* et est donc une classe composante de cette dernière.

**LO22 – P15 – ProjectCalendar**

EvenementManager est donc une classe qui permet de gérer les différents évènements de l’application. Elle comme attribut un tableau d’objets de la classe *Evenement*, mais également une instance unique sur elle-même. Elle comporte des méthodes qui permettent de chercher si une certaine tâche ou activité est déjà programmé dans la liste des évènements. Mais également deux méthodes (une pour les taches unitaires une pour les activités) qui permettent de vérifier que rien n’est programmé dans le calendrier à la date où l’on souhaite programmer cette tache unitaire/ activité, puisque comme précisé dans l’énoncé, deux évènements ne peuvent être programmés le même jour. Nous avons également dans cette classe, une classe Iterator qui permet de parcourir l’ensemble des évènements de l’application. Enfin elle permet d’ajouter la programmation de taches unitaires/ activité dans le calendrier en prenant en compte les méthodes précédemment explicitées afin de respecter les contraintes liée au sujet.

ProgrammationTâche est une classe qui hérite de la classe *Evenement*. Elle a comme unique attribut un pointeur sur un objet de la classe *TacheUnitaire*. Elle a une méthode qui permet de renvoyer la tache unitaire en question. Et une méthode qui définit le type de l’évènement en tâche.

ProgrammationActivite est une classe qui hérite également de la classe *Evenement.* Elle permet de définir une activité comme étant un évènement avec en plus un titre et un lieu. Ses méthodes permettent de retourner ces attributs, mais également de définir le type de la tâche en activité. Cette classe est héritée par *EvtPj* et *Evt1jDur*.

EvtPj est une classe qui hérite de la classe *ProgrammationActivite*. Elle a comme attribut supplémentaire la date de fin de l’évènement. Elle a des méthodes qui permettent de récupérer ses différents attributs.

Evt1jDur est une classe qui hérite également de la classe *ProgrammationActivite*. Elle a comme attribut supplémentaire la durée de l’évènement. Elle a à l’instar de la classe *EvtPj* des méthodes qui permettent de récupérer ses différents attributs.

**LO22 – P15 – Annexes**

## 

