Modèles de conception réutilisables

Projet de groupe

-- Circulation Mediator --

Raphaël Racine, Yassin Kammoun,
Valentin Minder, Paul Ntawuruhunga

18 juin 2015

pour

Le professeur Pier Donini
L'assistant Sébastien Rosat

Table des matières

Introduction

Ce mini-projet porte sur le design pattern Médiateur et il constitue à la réalisation d'un petit logiciel de simulation routière simplifié, qui s'appuie sur ce modèle.

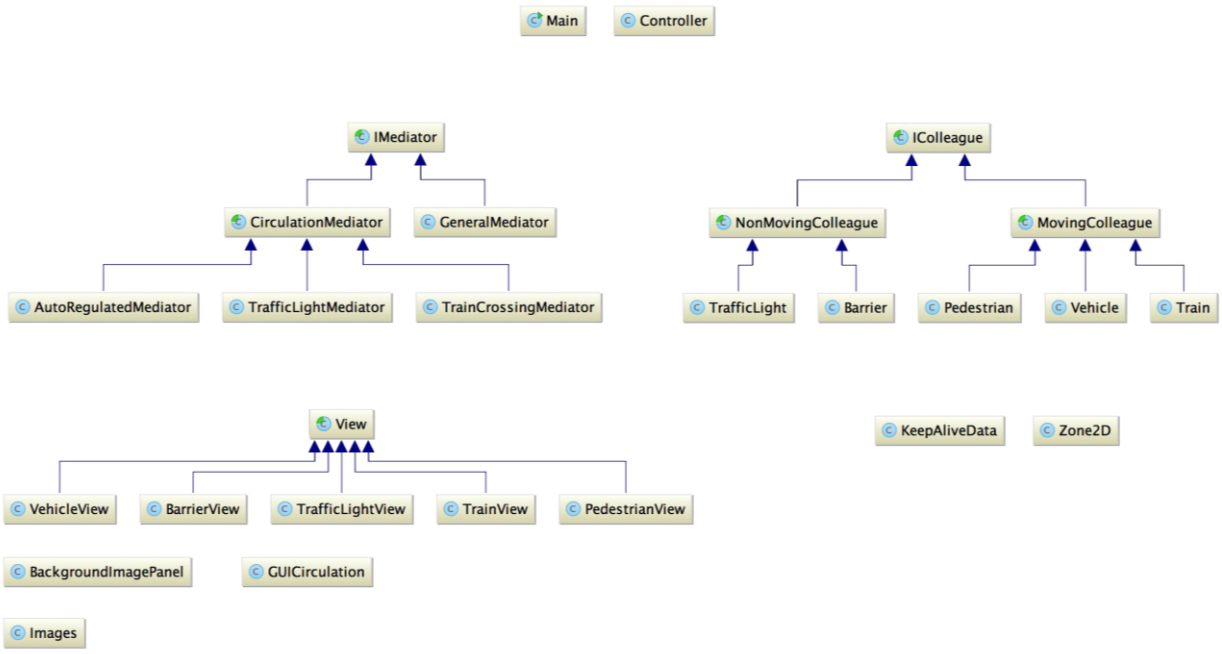
Il consistera principalement à faire communiquer des voitures, des trams et des piétons afin que chacun respecte les priorités, et subiront des accidents s'ils ne font pas attention à ce qu'ils font.

Choix d'implémentation

La plupart du temps, le pattern `Mediator` est expliqué avec des exemples assez simples tels que tchats, interface utilisateur, etc… Dans ces cas, le rôle du médiateur est très bien défini : Transmettre un message d'un objet à un ou plusieurs autres. Les destinataires sont clairement définis. Cependant, en développant cette application de simulation de circulation routière, nous avons dû faire face la problématique suivante :

Il y a un grand nombre de collègues de type distincts. Ils transmettent des messages qui ne concernent qu'une catégorie d'objets. Par exemple, les véhicules et piétons communiquent pour ne pas entrer en collision, les feux de circulation communiquent pour être synchronisé, les véhicules communiquent avec les feux de circulation afin de respecter des règles. Les trains communiquent avec les barrières, qui elles communiquent avec les véhicules, etc…

La première approche était d'utiliser un seul médiateur. Mais au vu la situation, son implémentation aurait été d'une grande complexité. Ce qui augmente le risque d'erreurs. La méthode que nous avons adoptée, que nous détaillerons plus bas, est d'utiliser plusieurs médiateurs. Les collègues gardent toujours une référence sur un seul médiateur, mais qui peut changer dynamiquement selon leur position.



Packages

Notre application est divisée en 4 packages.

1. Le package Mediator
   Définit les différents médiateurs de l'application. Comme nous le verrons plus bas, il y a plusieurs médiateurs organisés en hiérarchie.

2. Le package Colleague
   Définit les différents types de collègue

3. Le package Protocol
   Celui-ci permet de standardiser les différents messages qui circulent afin d'aider les médiateurs à réagir aux messages reçus.

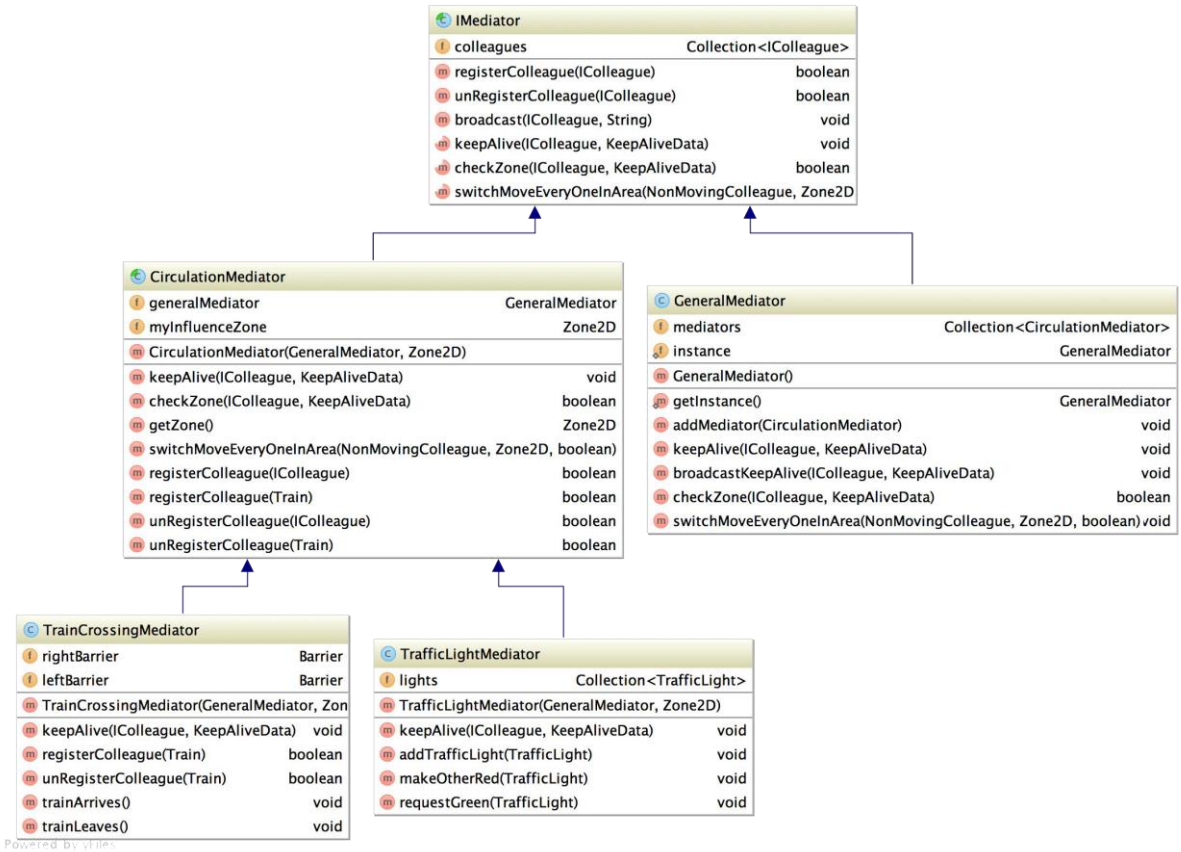
4. Le package GUI
   Les classes s'occupant de l'affichage. Ce package utilise le modèle Observateur pour observer le déplacement des collègues.

Mediators

Au lieu de déléguer à un unique médiateur la redirection des messages et la logique de l'application, on définit plusieurs types de médiateurs qui seront spécialisés pour la gestion d'un type de message précis. Il est ainsi plus simple de définir leur comportement. Comme il y plusieurs médiateurs, et que chaque collègue doit avoir une référence sur un seul médiateur, Il est nécessaire d'avoir un `GeneralMediator` qui pourra selon la position d'un collègue lui spécifier son médiateur direct. On dit que le `GeneralMediator` enregistre le collègue au près d'un autre médiateur

A leur création, chaque collègue possède une référence sur le `GeneralMediator`. Pendant que l'objet se déplace, il envoi un message de type `keepAlive` « voici ma position » destiné à tous ses collègues. Le `GeneralMediateur`, avant de transmettre ce message vérifiera la position du collègue puis lui définira un nouveau médiateur s'il se trouve dans sa zone. Par exemple, lorsqu' une voiture s'approche d'un feu rouge, son ancien médiateur sera remplacé par le `TrafficLightMediator` de la zone correspondante. Même si le médiateur d'un objet change, certains messages continuent à être redirigé vers le `GeneralMediator` par le biais de son médiateur courant. Ainsi le `GeneralMediator` aura toujours le contrôle sur certains types de messages (les « `KeepAlive` » par exemple).

Ainsi, le premier rôle du `GeneralMediator` est de vérifier la position d'un collègue et de lui attacher un nouveau médiateur si besoin. Puis son second rôle et de rediriger des messages à tout les collègue (Broadcast) car c'est le seul à tous les connaitre.



Colleague

On distingue deux types de collègue. Le MovingColleague bénéficiant d'un changement dynamique de médiateur et le NonMonvingColleague pour lequel le médiateur est fixe. Les véhicules, trains et piétons sont donc des MovingColleague. En revanche les feux rouges et barrières sont des NonMovingColleague.

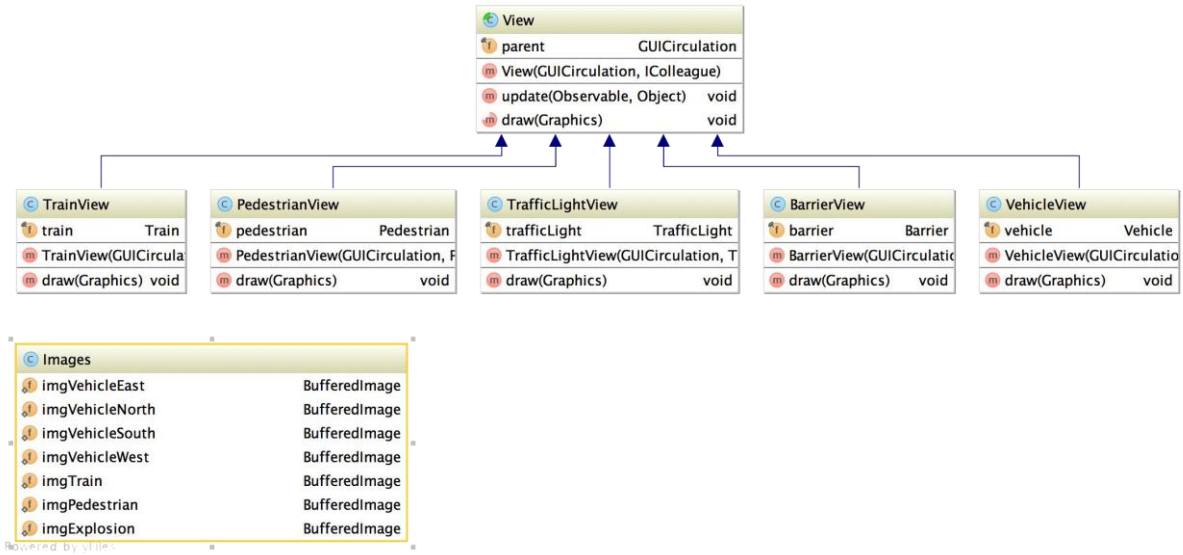
La classe Zone défini la place occupée par un collègue ou la zone d'influence d'un médiateur. Un Collègue envoie ses messages de type KeepAlive, servant à indiquer sa position, en passant en paramètre la zone qu'il occupe. Les médiateurs se servent de cette zone pour réagir. Le TrafficLightMediator comprendra ce message comme « Bonjour, je suis tel collègue et j'aimerais que tu passe au vert ». Le generalMediator comprendras « .. vérifie que je m'adresse au bon mediateur ».

Les TrafficLight ou Barrier sont aussi des collègues et ont leur propre type de message destinés à tous les movingColleague de leur zone d'influence. Lorsqu'un feu ou une barrière change d'état il/elle souhaite le communiquer à ses collègues par le biais d'un message tels que « Que tous ceux qui sont dans ma zone s'arrêtent » par la methode switchMoveEveryOneInArea().

Comme les keepAlive sont des messages de type broadcast, un MovingCollegue recevant un tel message s'en servira pour vérifier s'il peut continuer à avancer ou s'arrêter afin d'éviter un accident. Ceci montre que les médiateurs ne sont pas des controlleurs mais servent simplement à transmettre des messages. Ensuite, libre choix au colleague de traiter ce message comme bon lui semble. C'est la raison pour laquelle nous leur avons ajouté l'attribut isSignalisationAware.

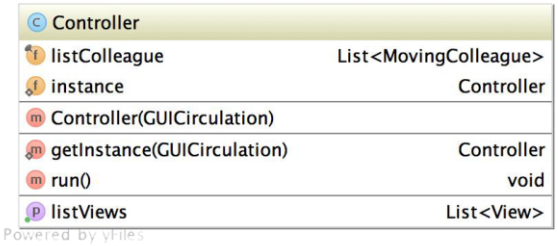
GUI

Concernant l'affichage, nous utilisons le pattern Observer. Chaque vue du package views possède une référence vers son objet observé (un colleague). Afin de s'afficher correctement, chaque vues héritant de View redéfinissent respectivement la méthode draw.



Controller

Le contrôleur a pour tâche d'instancier tous les collègues, leurs vues et les médiateurs. Il fait avancer les collègues dans sa boucle principale.



Conclusion

Nous aurions souhaité rendre le contrôleur plus générique afin de positionner aléatoirement les éléments sur la carte routière. Cependant, étant focalisé sur l'implémentation du pattern Mediator nous avons dû mettre de côté cette fonctionnalité. Notre carte reste donc fixe et les éléments ont donc du être positionné à la main. De plus nous aurions voulu rendre cette application plus interactive en permettant à l'utilisateur de jouer à un jeu plutôt que d'observer. Le temps manquant étant notre plus grand ennemi, nous avons décidé de perfectionner au maximum la simulation.

Ce mini-projet nous a permis de mieux comprendre le principe du médiateur et nous avons réussi à l'appliquer à notre cahier des charges de départ.