HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Gestion d'un carrefour à l'aide d'un réseau de Petri

Rapport du projet

Auteur : Bise Léonard
Professeur : Zysman Eytan
Filière : Informatique Embarquée (ISEC)
Cours : Programmation Concurrente (PConc)
Date : 10 Mai 2018

Table des matières

1 Introduction

Dans le cadre du cours Programmation Concurrente (PConc), il nous a été demandé de créer un programme qui simule le comportement d'un carrefour. Le carrefour est composé de deux flux, un Sud-Nord et l'autre Ouest-Est ou les voitures circulent dans un seul sens. A l'intersection des deux flux sont placé deux feux de signalisation, un par flux, ils permettent ou non au voiture de passer.

La gestion du système est fait grâce à un réseau de Petri qui réagit aux événement et qui produit des actions.

2 Cahier des Charges

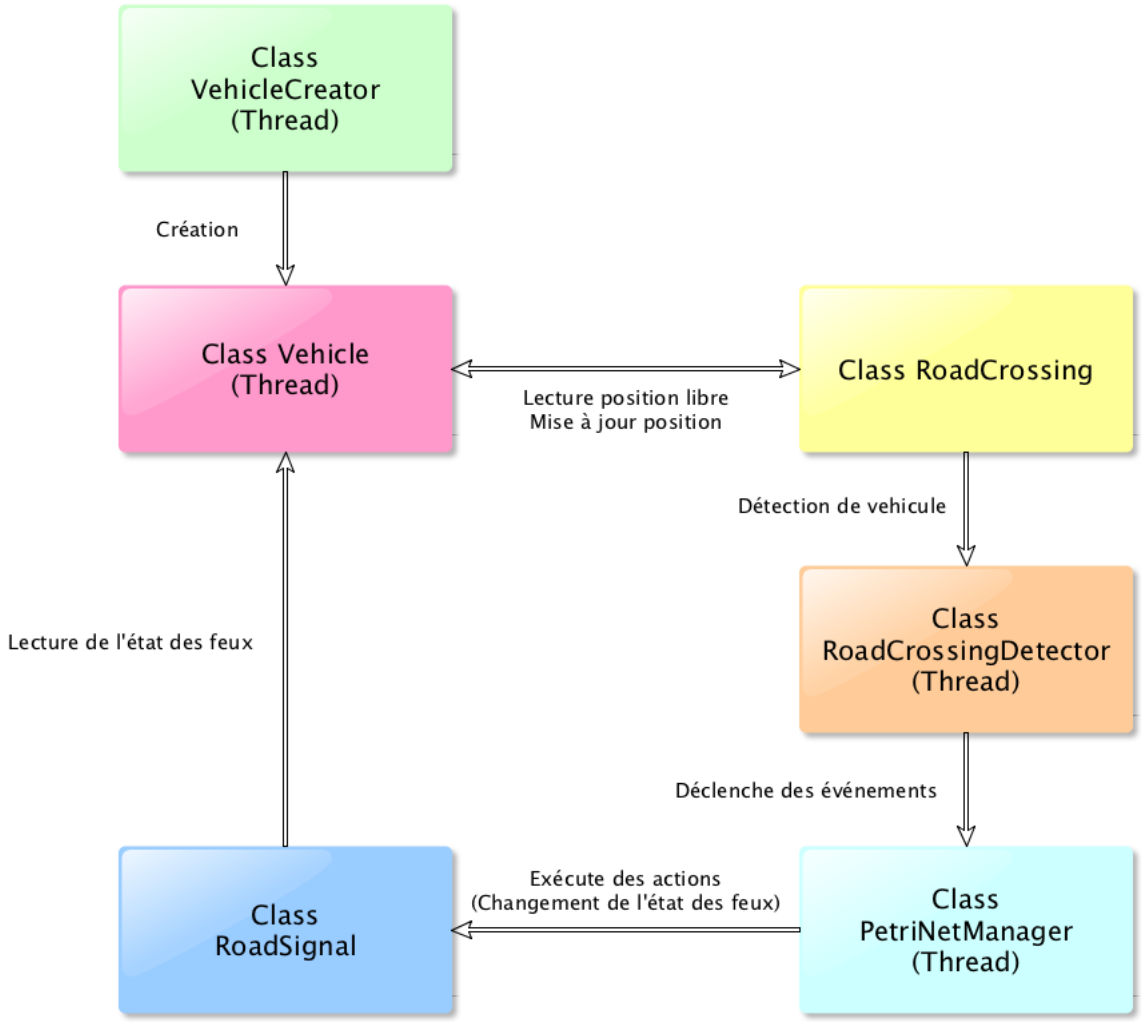
Dans le cadre du projet il nous est demandé de créer un projet codé en java comprenant les éléments suivants :

— Créer un gestionnaire de réseau de pétri auquel il est possible de donner un fichier de configuration pour créer un paysage (création des places, transition, arcs de pre et post incidence). Une fois le paysage créé on peut lancer la simulation du réseau de pétri. Lorsque des jetons arrivent dans des places, des actions sont lancé au travers d'objet qui implémente une interface PetriPlaceAction. Le système peut déclencher des événements liés au réseau de petri qui permettent de franchir des transitions.

— Utiliser le gestionnaire de réseau de pétri afin d'implémenter un réseau qui permet la gestion d'un carrefour avec deux flux perpendiculaire de voiture et qui est à sens unique. Des véhicules seront créé à des temps aléatoire par des threads dans les deux flux, ils avanceront en direction du carrefour, à leur arrivé devant le carrefour ils devront vérifier si le feu de signalisation est vert avant de pouvoir entrer dans le carrefour. Le RDP permettra, au travers d'actions liées à des places ainsi que d'événements, de changer la couleur des feux.

3 Architecture

L'image suivante montre l'architecture générale de l'application.



4 Le mécanisme du Réseau de Petri

La classe PetriNetManager implémente le gestionnaire de réseau de Petri. D'autres classes sont utilisées pour la gestion des réseaux, elles sont décrites ci-dessous.

- PetriNetManagerTest : Contient quelques tests du bon fonctionnement du gestionnaire de réseau
- PetriArc : Décrit les caractèristiques lié aux arcs de pré et post incidence comme le type (simple, test, inhibit) et leur poids
- PetriPlace : Une place dans un réseau de Petri ainsi que sont action liée si elle en as une
- PetriPlaceInterface : Une interface qui permet à la classe qui l'implémente d'être ensuite déclenchée par le gestionnaire de réseau lorsqu'un jeton entre dans la place qui y est liée.
- PetriTransition : Contient les informations relative à une transition faisant partie du réseau de Petri

La classe PetriNetManager est la classe centrale du gestionnaire de réseau de Petri. Après l'instanciation de la classe, un réseau de Petri peut être initialisé en utilisant la méthode loadFromTextFile avec un fichier de configuration. Ceci créera toute les places, transitions et arcs requis pour le réseau décrit.

Une fois le réseau initialisé l'utilisateur à la possibilité de faire les actions suivantes :

- Lier une place du réseau avec une classe implémentant l'interface PetriPlaceInterface avec la méthode setPlaceAction. Lorsque le gestionnaire du réseau détecte l'entrée d'un jeton dans la place, il déclenchera automatiquement l'exécution de l'action
- Lancer le gestionnaire de réseau de Petri avec la méthode start. Cette action lancera l'exécution du Thread de PetriNetManager qui après chaque tick effectuera la simulation du réseau
- Une fois le gestionnaire lancé, l'utilisateur doit déclencher des événements au moyen de la méthode fireTransition qui permettre à l'utilisateur de déclencher l'événement lié à une certaine transition ce qui permettra le franchissement de la dite transition si elle est préalablement sensibilisée

Le coeur de la simulation du réseau de Petri est effectuée par la méthode step. Cette méthode simule un pas de l'évolution du réseau de Petri, elle est cadencé par la durée d'un tick, c'est à dire un temps entre deux pas.
La liste suivante décrits les opérations effectuées par la méthode step, elle se découpe en quatre phases.

- Phase 0 : Les actions des places qui ont eu de nouveaux jetons sont exécutées
- Phase 1 : Le gestionnaire détermine les transitions qui sont sensibilisées, c'est à dire que les places qui y sont liées contiennent le nombre de jeton requis, et

les ajoute dans une liste de transition sensibilisée. La liste est ensuite mélangée pour que les transitions aient un ordre aléatoire

— Phase 2 : Pour chaque tranisition qui sont dans la liste des transitions sensibilisées, le gestionnaire vérifie si l'événment associé a été déclenché. Si c'est le cas, les jetons des places qui sont liées à la transition sont consommé et elle est ajoutée à la liste des transitions franchies. La liste des transitions sensibilisées est à nouveau parcourue pour vérifier si certaines transitions ne le sont plus suite à la consommation de jeton

— Phase 3 : Les jetons nécessaire sont produit dans les places liées aux transitions qui ont été franchies

5 Implémentation des acteurs

Le projet de gestion de carrefour demande la création de différents acteurs qui sont listé ci-dessous.

- RoadCrossing : La grille sur laquelle les voitures se déplace. Cette classe est passive, elle ne fait que d'être modifié par d'autre classe, elle n'as pas son propre Thread.
- RoadCrossingDetector : Les detecteurs utilisés pour déclencher des événements dans le réseau de Petri. Il y'en a deux par flux. Un placé juste avant le carrefour déclenchera l'événement lié à la détection de flux voulant entrer dans le carrefour, cela permet au réseau de Petri de faire passer le feu de l'autre flux au rouge et ainsi permettre au premier flux de pouvoir passer
- RoadSignal : Le feu routier, c'est un élément passif qui ne fait que d'avoir un état, soit rouge ou alors vert. Il y a un feu par flux, les véhicules lorsqu'ils arrivent devant le carrefour doivent veiller à vérifier que le feu est vert avant de pouvoir y rentrer. L'état des feux est modifiés au travers de deux actions du réseau de Petri.
- EventManager : C'est une classe qui permet de définir des événements réél et les associer a un événement du réseau de Petri en utilisant à l'interne des classe de type Event. C'est donc l'interface entre le monde réél et le réseau de Petri
- RoadCrossingEventManager : Utilise la classe EventManager pour définir les événements qui sont propres au problème du carrefour (par exemple Génération d'un nouveau véhicule ou arrivée devant le carrefour)
- SignalStateAction : Cette classe implémente l'interface PetriPlaceAction et défini donc l'action à effectuer lorsqu'un feu doit changer d'état. Elle directement appellé par le gestionnaire de réseau de Petri quand nécessaire et elle contient une réference au feu qu'elle doit gérer.
- Vehicle : Définit le comportement d'un véhicule, après avoir été créé il commence à avancer régulièrement le long de son flux en utilisant la grille défini par la classe RoadCrossing. Juste avant de rentrer dans le carrefour le couleur du feu sera vérifier pour voir si il est possible d'avancer. Lorsqu'elle arrive à la fin de la route, le véhicule a terminé sa vie et est détruit.
- VehicleCreator : Ce Thread est responsable de la création de véhicule dans un certain flux de manière régulière

6 Les échanges entre acteurs