Simulation d'un réseau routier

(À faire par groupes de 3 à 4 élèves)

Le but du TP est de réaliser un simulateur de réseau routier (on ne cherchera aucun réalisme par rapport à un vrai réseau). Le critère principal d'évaluation – en dehors du respect des normes de programmation Java communément admises (modularité, encapsulation, traitement d'exceptions) – sera **la clarté et l'évolutivité de votre conception en termes de hiérarchie de classes**, via l'identification de « schémas de conception » classique et leur bonne implantation.

Éléments du domaine à modéliser

1. Le réseau routier est constitué des « **segments de routes** » connectés par des « **jonctions** ». Un segment de route (toujours à deux voies de circulation) est caractérisé par sa longueur (en nombre d'une certaine unité) et peut être parcouru dans les deux sens. Deux segments ne sont jamais reliés directement mais par le biais d'une « jonction » de longueur une unité. Parmi les jonctions, on considère les « jonctions simples » qui relient exactement deux segments (par exemple un passage piéton protégé par des feux) et les « barrières » qui sont des jonctions à un seul segment servant à clore une route (une voiture ne peut donc pas avancer si elle est face à une barrière). Les carrefours sont des exemples plus complexes de jonctions : un carrefour connecte au moins 3 segments de routes. On peut avoir des carrefours en T, ou à 4 routes, à 6 routes, etc. Lorsqu'une voiture arrive à un carrefour, on tire au sort de façon uniforme parmi les autres segments de route celui sur lequel la voiture continuera son trajet.

2. Les voitures sont les éléments circulant sur le réseau. Une voiture a un identifiant et une longueur d'une unité. Elle a une vitesse maximale et un état courant (le segment de route et sa position sur ce segment, son sens de déplacement et sa vitesse courante). Les vitesses sont exprimées en unités de longueur parcourues par unité de temps. La vitesse est constante pendant une unité de temps et peut changer (instantanément !) entre deux unités de temps, par exemple pour s'arrêter à un feu. Une voiture roule au maximum de ses possibilités, selon les indications du réseau.

3. Les sémaphores, parmi lesquels les feux bicolores ou tricolores, et les panneaux « STOP » servent à réguler la circulation aux abords des carrefours ou des passages piétons. Un sémaphore est posé à une extrémité de segment de route et orienté vers l'intérieur de ce segment. Il est visible par toute voiture située sur ce segment et allant dans sa direction. Un feu bicolore peut prendre la couleur vert (la voiture qui voit le feu continue à sa vitesse courante) ou rouge (la voiture s'arrête instantanément). Quand un feu tricolore prend la couleur orange, la vitesse de la voiture est divisée par deux. Une voiture s'immobilise à un feu STOP pour au moins une unité de temps et passera ensuite si la voie est libre. On peut aussi imaginer d'autres sémaphores dont le rôle serait de limiter la vitesse à une valeur donnée.

4. Les capteurs comprennent notamment les capteurs de présence et les capteurs de vitesse. Un capteur est placé sur un tronçon de route et est associé à un « élément de régulation ». Un capteur de présence signale le passage d'une voiture pendant l'unité de temps courante. Un capteur de vitesse signale à son élément de régulation la vitesse de la voiture qui passe dessus.

5. Les « éléments de régulation » sont des éléments logiciels basés sur les capteurs et sémaphores et ils régulent la circulation aux carrefours. Un élément de régulation gère une configuration physique selon un algorithme donné. Différentes instances d'un même élément de régulation peuvent apparaître dans un réseau si on gère de la même façon des configurations identiques.

Un élément de régulation différent (pour une même configuration physique) peut apparaître sur une autre portion de réseau qui aurait besoin d'un algorithme différent, par exemple pour imposer des priorités entre voies. Il n'est pas prévu de changer dynamiquement l'algorithme d'un élément de régulation. Par exemple, un carrefour peut être équipé de feux de signalisation qui changent de couleur de façon périodique, indépendamment du trafic routier. Un autre carrefour peut être contrôlé par des feux qui ne changent de couleur que s'ils détectent qu'un véhicule est en attente sur une autre voie. Un dernier exemple serait celui d'un capteur de vitesse qui ferait passer un feu au rouge si une voiture va trop vite.

*Ce ne sont que des exemples. Il ne vous est demandé qu'un ou deux exemples simples qui montrent la flexibilité de votre simulateur. Votre projet ne sera **pas** évalué sur la correction ou la vraisemblance des algorithmes de régulation.*

L'écoulement du temps est **discret** (par opposition à « continu ») : on ne regarde pas ce qui se passe pendant une unité de temps. A chaque unité de temps, chaque voiture se déplace suivant son état et celui du réseau. L'ordre de traitement des voitures pendant l'unité de temps est arbitraire. À la fin de l'unité de temps, les éléments de régulation actionnent les sémaphores en fonction des informations fournies par les capteurs. Votre système doit signaler **les risques de collisions** : on considère qu'il y a **risque de collision** si pendant l'unité de temps écoulée, deux voitures sont passées sur un même capteur. On veut aussi être notifié automatiquement si une voiture essaye de franchir une barrière ou de passer un feu rouge.

Travail demandé

Modéliser et programmer les éléments présentés ci-dessus de manière à pouvoir simuler le trajet d'une ou plusieurs voitures sur un réseau routier précédemment construit :

1. Proposer une hiérarchie de classes Java, décrite en UML et utilisant de façon raisonnée les schémas de conception vus en TD et en cours. On ne fera **pas** apparaître les classes d'exceptions ou des classes « techniques » (entrées/sorties, comparateurs, etc.) De même, on ne fera apparaître que les principales méthodes pour comprendre la dynamique du système, à l'exclusion des constructeurs, de `toString` ou toute autre méthode auxiliaire.

2. Rédiger un document de 5 pages *maximum* pour commenter la hiérarchie proposée et décrire les tests **fonctionnels** réalisés. Indiquer la part réalisée par chacun.

Les interfaces graphiques sont interdites : consacrez-vous à l'essentiel ! Le réseau routier est défini au lancement de l'exécution et ne peut **pas** être modifié après coup. Il est construit par programme : il n'est **pas** demandé que le réseau soit lu dans un fichier ou construit interactivement. Il est recommandé de pouvoir construire des portions de réseau sans être obligé de revenir aux éléments de base. Vous pouvez ne pas considérer des configurations de réseaux ou de circulation trop complexes à modéliser (en les signalant dans le dossier remis).
dans le dossier remis)document)