Projet Long TOB
Rapport Général 3

Groupe EF-5

CHEGGAF Ahmed
CHEVALLEREAU Adrien
CONTET Clément
CROUZET Sylvain
GUILHE LA COMBE DE VILLERS Pierre-Louis
DAI Guohao
DJILALI Célia

Département Sciences du Numérique - Première année
2020-2021

Table des matières

1 Introduction

Le projet a pour objectif de proposer un outil afin de construire un système routier, de le simuler et enfin de l'optimiser. Après avoir construit le système routier à étudier, l'utilisateur entre les données de simulation, l'interface propose alors une représentation graphique afin de visualiser la simulation et présente des données analysées.

2 Les principales fonctionnalités

L'application finale propose trois étapes principales à l'utilisateur :
— Création du système routier
— Simulation du système routier
— Analyse des données générées par la simulation

Lors de ces 3 itérations, nous avons pu développer l'étape de création et une partie de l'étape de simulation.

Lors de l'étape de création, l'utilisateur à la possibilité de créer un système routier sur une carte située dans l'onglet "Création" de l'application. Il peut alors placer les éléments du système routier qu'il souhaite simuler. Cette fonctionnalité a commencé à être développée lors de la 1ère itération et a été terminée lors de la 3ème itération. Cependant, il ne peut pas encore lancer et visualiser la simulation.

3 Principaux choix réalisés

Pour développer l'application, nous avons tout d'abord choisi de diviser l'implantation en 3 packages :
— Véhicules : implantation des voitures circulant dans le système
— Réseau : implantation des axes routiers ainsi que les intersections et rond-points
— Vue : implantation de l'interface graphique

Nous avons ensuite décidé que l'utilisateur ne pouvait créer des axes routiers qu'en ligne droite pour le moment. Ce choix nous a facilité le développement des packages Réseau et Vue.

4 Implantation des véhicules

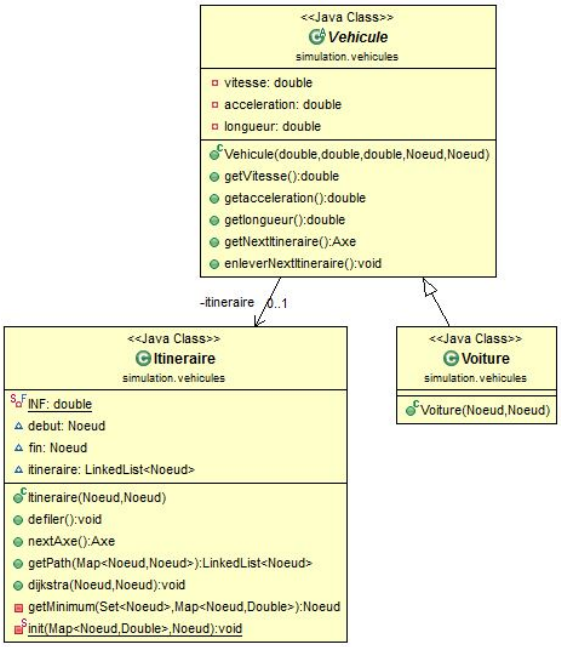

FIGURE 1 – Diagramme de classe du package véhicule

Le package véhicule implémente donc les véhicules (Cf figure 1) qui, dans la simulation, parcourront le réseau. Pour l'instant seul une voiture standard est implanté, mais par la suite nous voudrions implanter des camions, des motos etc.

La seconde partie du package se compose de `Itinéraire.java` qui va jouer le rôle d'un GPS. En effet en lui donnant une entrée et une sortie sur le réseau alors itinéraire va trouver le chemin le plus court par l'algorithme de Dijsktra et ainsi le véhicule parcourra cet itinéraire jusqu'à sa destination finale.

5 Implantation du réseau

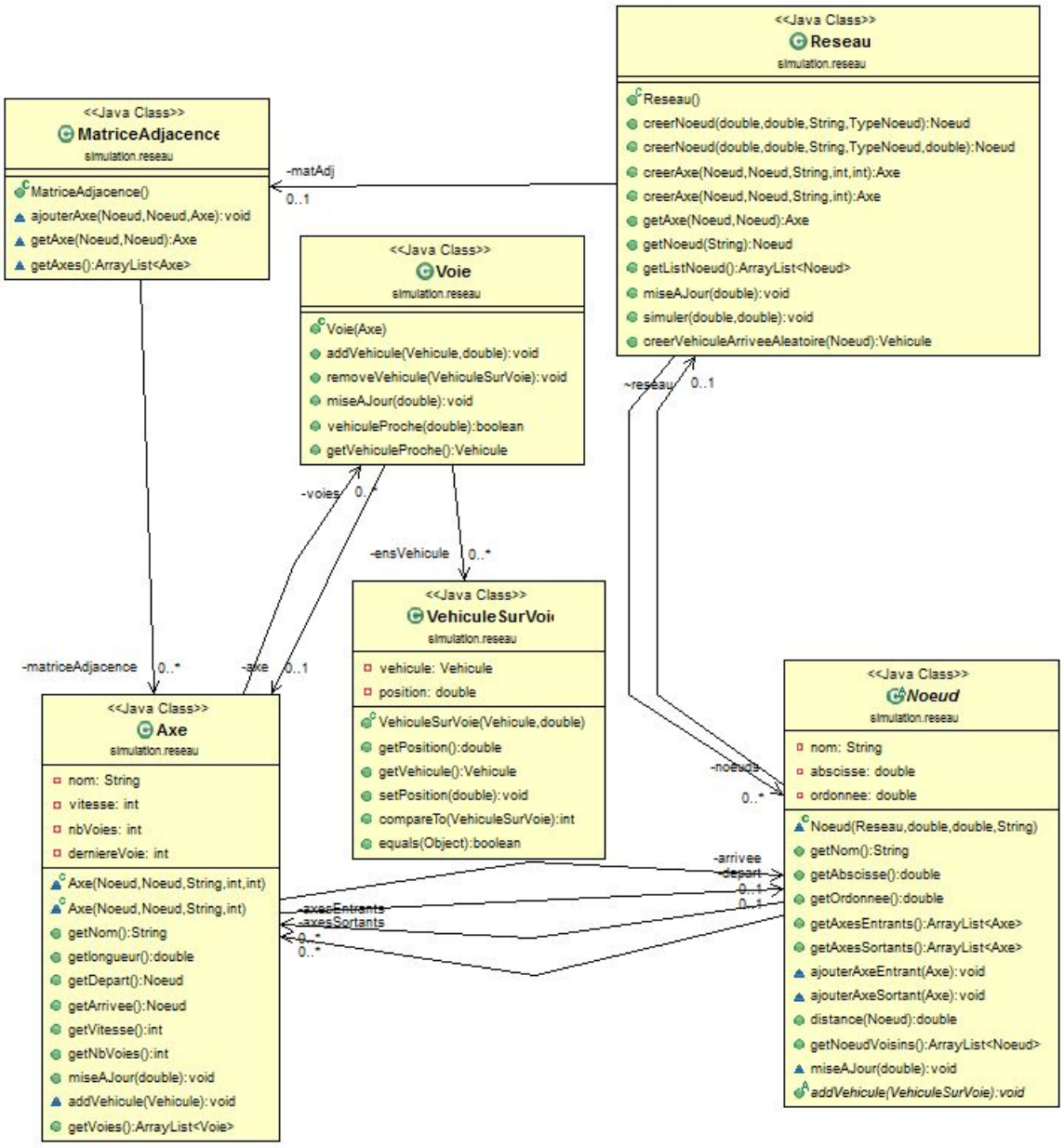

FIGURE 2 – Diagramme de classe du package réseau

Dans le package réseau (Cf figure 2) se trouve le coeur de la simulation. En effet c'est ici que les différents objets routiers sont créés et stockés ainsi que toutes les méthodes essentielles à la simulation. Tout le réseau est découpé en deux objets : les Noeuds et les Axes.

5.1 Noeuds

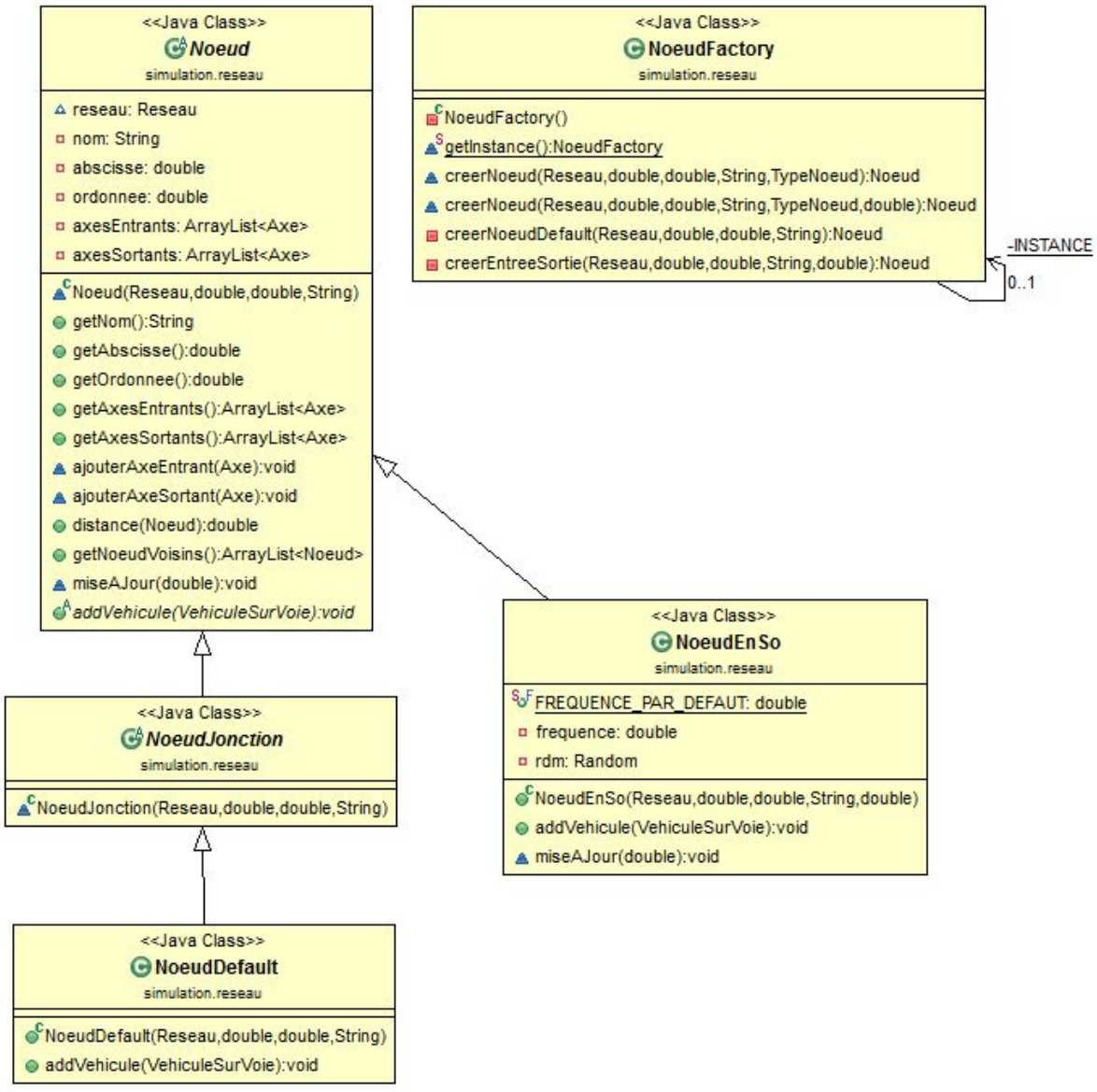

FIGURE 3 – Diagramme de classe du package réseau

Les Noeuds (Cf figure 3) représentent les Entrées/Sorties (section 5.1.1) et toutes les intersections à savoir `Intersection.java` qui représente le carrefour de base (section 5.1.2). Puis les autres intersections plus complexes comme les Rond-Points qui sont seulement une ensemble de carrefours reliés par un axe à sens unique avec des priorités particulières.

5.1.1 Entrées/Sorties

Dans notre modèle les entrées et les sorties représentent les endroits où les voitures sont créées et détruites. On suppose ainsi que c'est les bords de notre réseau de simulation. Ainsi ce sont des

noeuds mais reliés à maximum deux axes (un vers et un depuis ce noeud). On peut ainsi définir, au niveau de cet objet, un flux de véhicules.

5.1.2 Intersection

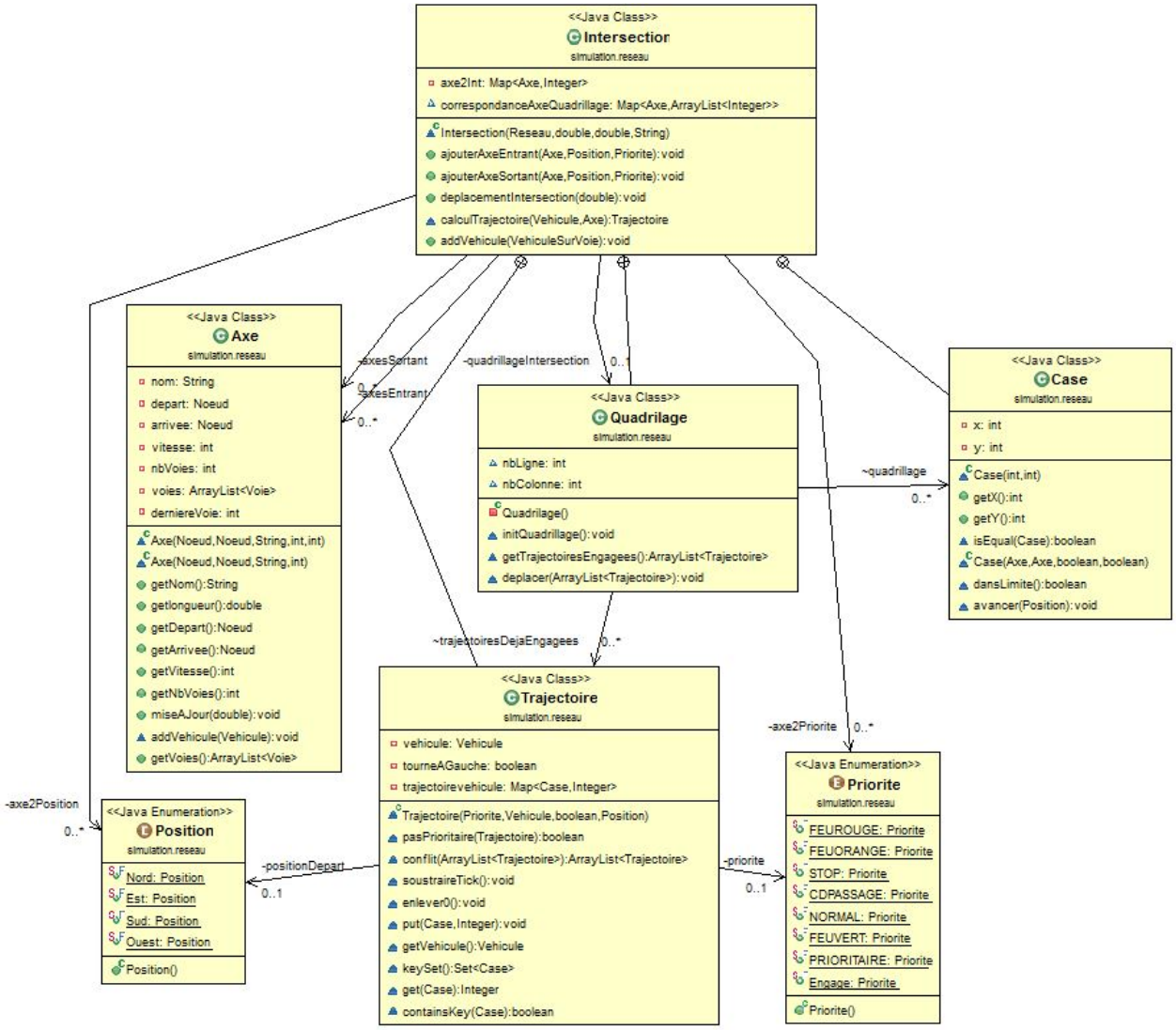

FIGURE 4 – Diagramme de classe du package réseau

`Intersection.java` est une sous classe de `Noeud.java` et représente une intersection en forme de croix, entre différentes routes. Premièrement chaque axe routier que gère `Intersection.java` est associé à une priorité (voir diagramme de classe figure 4), ce mécanisme permet de traiter avec la même structure des feux de circulation et des intersections basées sur des cédez-le-passage, et feu. L'algorithme calcule les trajectoires de tous les véhicules qui sont à portée de pouvoir s'engager dans l'intersection, puis pour chaque conflit entre 2 trajectoires, il élimine celui qui a une priorité plus basse.

Ainsi on a désormais un ensemble de trajectoires qui ne se coupent pas avec les priorités les plus importantes en premier. Pour des véhicules qui ont une priorité égale, on applique la priorité à droite, le véhicule qui se trouve dans un axe vers la droite est donc prioritaire. Enfin

l'intersection garde en mémoire les véhicules qui sont en train de le traverser, et leur donne une priorité supérieure à toutes celles des axes de l'intersection.

5.2 Axes

Un `Axe.java` est représenté dans le monde réel par une route à plusieurs voies à sens unique avec une vitesse prédéfinie. Un Axe relie obligatoirement deux noeuds entre eux. Pour l'instant seul les axes en lignes droite sont implantés pour faciliter la création mais à terme nous aurions voulus implanter les axes courbés. Actuellement les véhicules ne peuvent pas changer de voie.

5.3 Voies

Comme dit précédemment, une voie est l'élément qui constitue les axes. Dans le programme, la voie conserve en mémoire des `VehiculeSurVoie.java` qui sont globalement des `Vehicule.java` auxquels on a ajouté un réel représentant la position du véhicule qui décroit depuis le début de la voie pour atteindre zéro au bout de la voie. Ces véhicules sur voie sont stockés en mémoire par un TreeSet.

Les TreeSet sont des ensembles sans doublon qui stockent les éléments de manière ordonnée sous la forme d'un arbre en les comparant entre eux. Ils permettent d'insérer des éléments dans n'importe quel ordre et de les restituer et parcourir dans un ordre précis efficacement. Toutes ces propriétés en font la classe idéale pour stocker des véhicules qui ne peuvent se superposer, qui sont dans un ordre précis et qui pourraient s'insérer n'importe où grâce à un changement de voie (pas implémenté). De plus, on veut pouvoir parcourir l'ensemble dans un ordre précis (on veut faire avancer un véhicule si le véhicule devant lui a déjà avancé) et vérifier s'il y a des véhicules à une certaine portée, ce que permettent facilement les TreeSet.

6 Implantation de l'interface graphique

6.1 Structure de la fenêtre principale

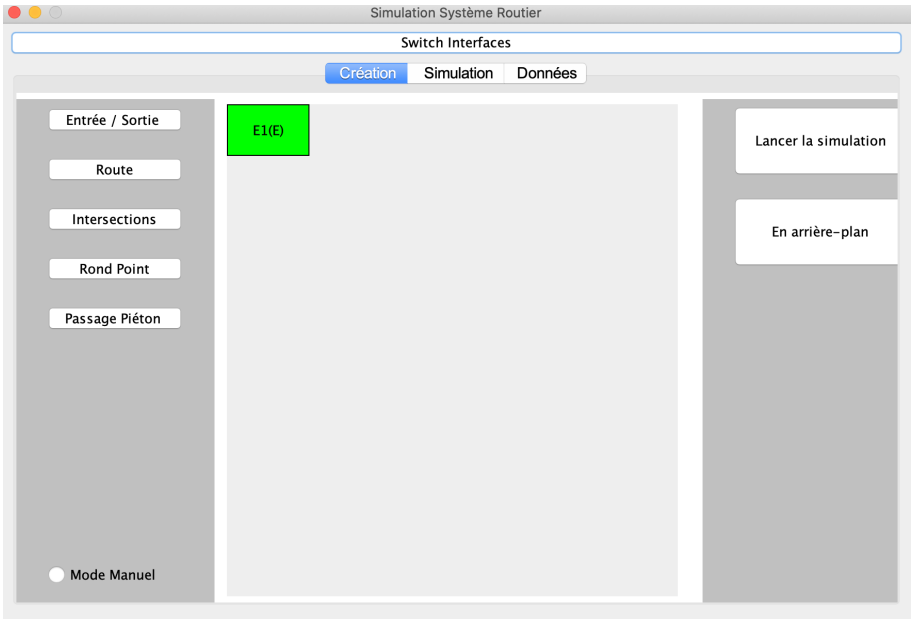

FIGURE 5 – 1ère version de l'interface graphique

Pour l'implantation de la structure de la fenêtre principale, l'outil WindowBuilder disponible sur Eclipse a été utilisé. En effet, l'interface mise à disposition par ce programme rend la création d'éléments Java Swing plus simple et permet une meilleure visualisation du résultat final.

La fenêtre principale correspond à la classe `Windows.java`.

La fenêtre a donc été divisée en trois onglets correspondant chacun à une étape : Création, Simulation, Résultat. Cette fenêtre est présentée sur la figure 5.

6.1.1 Onglet "Création"

Dans cet onglet, l'utilisateur est invité à créer un système routier grâce à différents éléments (route, intersection...).

Nous avons choisi d'utilisation des JButton pour chaque élément à créer.

6.1.2 Onglet "Simulation"

Un élément JTabbedPane contenant les 2 onglets "Rapide", "Détaillé" sont ajoutés pour choisir le mode de simulation.

Un JPanel a été créé pour l'affichage de la simulation.

6.1.3 Onglet "Résultats"

C'est dans cet onglet que seront visibles les statistiques de la simulation.

6.2 Création du système routier par entrée de coordonnées

Nous avons commencé par implémenter une version de l'application où l'utilisateur peut ajouter des éléments grâce à des boîtes de dialogue.

Pour ajouter des éléments à son réseau, il faut que l'utilisateur entre les paramètres de l'objet à créer. Pour cela une classe est créée pour chaque boîte de dialogue : `CreerRoute.java`, `CreerInter.java`, `CreerRondP.java`, `CreerEntreeSortie.java`. Un exemple de boîte de dialogue est présenté sur la figure 6.

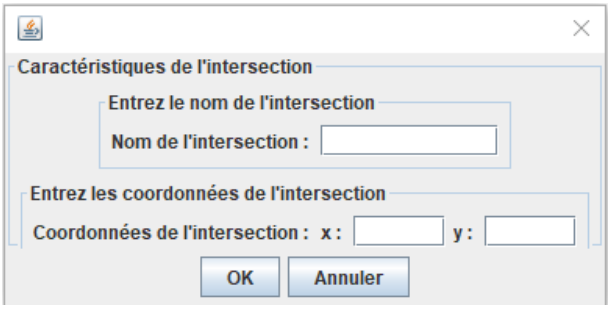

FIGURE 6 – Exemple de boîte de dialogue (création d'une intersection)

6.2.1 Gestion des évènements

Pour que les paramètres entrés dans les boites de dialogue permettent de créer les objets du système, les méthodes créatrices sont appelées dans la méthode `valider`, listener du bouton "OK".

Pour que la map de la fenêtre principale soit modifiée lors de cet évènement, la fenêtre principale est mise en attribut des classes `CreerRoute.java`, `CreerInter.java`, `CreerRondP.java` et `CreerEntreeSortie.java`. De plus, une méthode `getMap()` qui renvoie le JPanel de la map a été définie dans la classe `Windows.java` afin d'ajouter des éléments à ce JPanel.

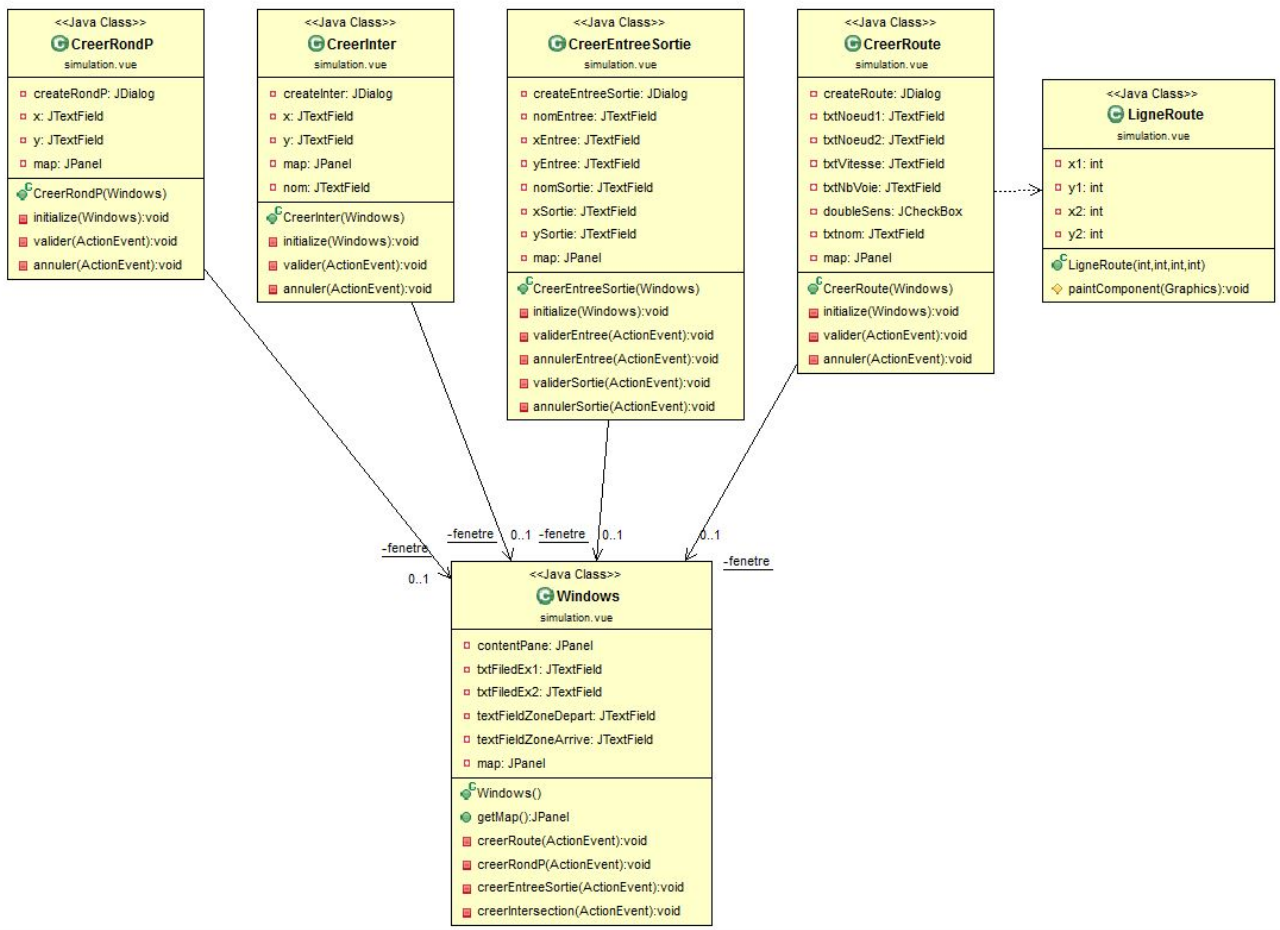

FIGURE 7 – Diagramme de classe de la première version de l'interface graphique (onglet création)

Pour afficher les rond-points et intersections des images sont utilisées. Pour les entrée/sortie, se sont des JLabels qui sont affichées. Pour afficher les axes, une classe `LigneRoute.java` a été ajoutée pour dessiner une ligne entre deux points. La classe Graphics2D de Java est donc utilisée pour cela.

6.2.2 Diagramme de classes

Le diagramme de classes de la figure 9 permet de visualiser les liens entre les différentes classes de cette première version.

6.3 Création du système routier par clic

Pour la création du système par clic, nous avons implémenté un JPanel contenant l'intégralité des éléments pour pouvoir l'intégrer facilement dans l'architecture. Ce JPanel utilise deux listeners pour suivre les mouvements et les clics de la souris.

Les éléments tels que "Route", "RondPoint", "Source et "Intersection" sont des JComponent et ils remplacent la méthode `paintComponent` pour qu'ils puissent avoir leur propre forme.

6.3.1 L'architecture globale du JPanel

Le JPanel principale contient trois JPanel, le premier contenant les buttons qui permettent à l'utilisateur de choisir le type d'élément à dessiner (route, rondpoint, intersection ou bien une source). Le deuxième JPanel représente le PanelDrawing qui est un sous type de JPanel, cette classe override la méthode de paint pour dessiner les éléments enregistrer dans les liste (car après la création de chaque JComponent, on les stocke dans leurs listes respectifs). Et le dernier JPanel représente les paramètres de chaque type, tous les buttons et les JTextfields ont leurs propres `ActionListener` et sont connectées avec le panel principale.

FIGURE 8 – Vue de la interface interactive

6.3.2 Gestion des clics

Lorsque l'utilisateur clique sur un endroit de l'écran, la gestion dépend principalement du type sélectionné.

— Si on souhaite insérer une route, il faut que la souris pointe sur un noeud d'un coté et un autre de l'autre sinon on ne sera pas capable de dessiner la route, après avoir dessiner la route on ajoute un axe au réseau principale.
— Si on souhaite ajouter une source ou bien un rond point, l'utilisateur peut positionner son élément où il veut dans l'écran, le seul point à respecter est de ne pas super positionner deux éléments du même type.

6.3.3 Gestion du comportement des JComponent

Chaque JComponent possède une fonctionnalité qui se traduit par un changement de couleur quand l'utilisateur pointe la souris sur l'élément, cela est fait en utilisant une méthode qui parcourent les listes et applique une méthode intitulée `isIn` qui est relativement la même car elle crée un rectangle de même dimension que le JComponent et renvoie si les coordonnées sont dedans, sauf pour le rond point où on utilise la position de la souris par rapport au centre pour pouvoir récupérer l'angle entre la souris et le rond point, de telle manière on pourra sélectionner un des quatre cotés du rond point.

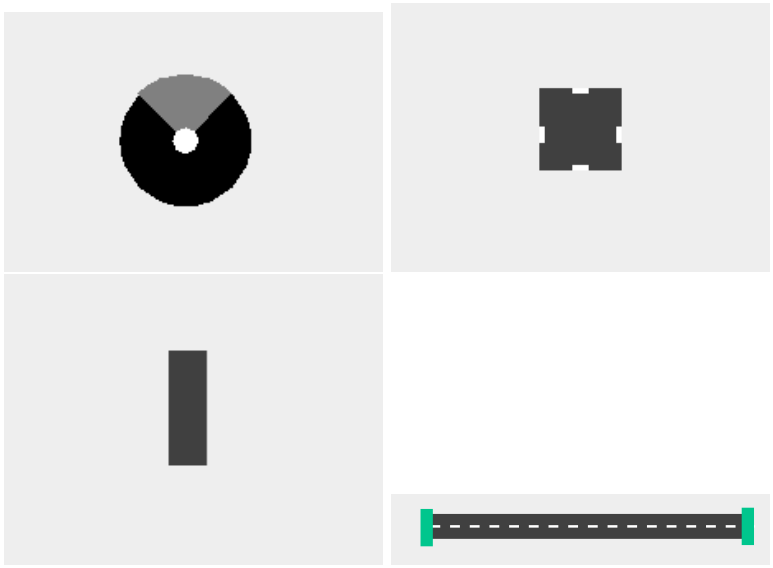

FIGURE 9 – Comportement des éléments quand ils sont sélectionnés

6.3.4 Diagramme de classes

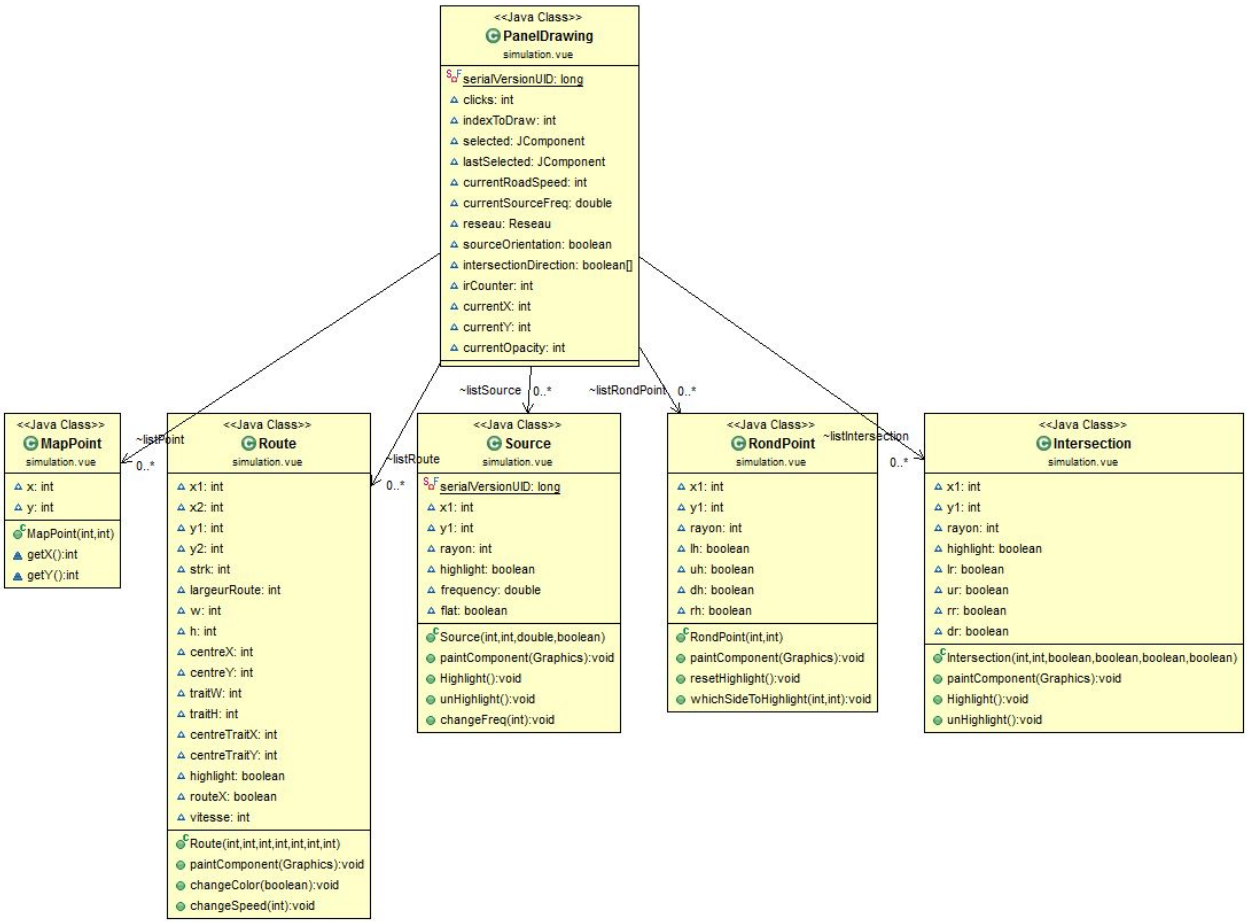

FIGURE 10 – Diagramme de classe de la deuxième version de l'interface graphique.

Le diagramme de classes de la figure 10 permet de visualiser les liens entre les différentes classes de cette deuxième version.

7 Organisation de l'équipe grâce aux méthodes agiles

Une fois le sujet du projet long fixé, nous avons effectué une réunions pour découper le projet en plusieurs épics afin d'établir une stratégie globale.

Nous avons notamment décidé d'opter pour un travail en sous-groupes composés de 2 à 3 personnes afin de pouvoir mettre en place, dans la mesure du possible avec le travail à distance, du pair programming. C'est pourquoi nous avons mis en place les trois pôles interface, réseau et véhicule. Cela permet aux membres d'un pôle de se concentrer pleinement sur la partie qu'ils sont en train de développer à un instant t. En contrepartie, cela nécessite de porter une attention particulière à la communication au sein du groupe afin de s'assurer qu'à chaque itération tout le monde soit à jour.

Au cours de la première itération, nous avons passé un temps conséquent à planifier à la fois à long terme, quelle direction va prendre le projet, quel est notre product backlog et à court terme, qu'allons nous réaliser dans cette première itération, quel est notre sprint backlog. Pour nous aider

nous avons utilisé le planning poker afin d'établir une estimation consensuelle du travail à réaliser assez rapidement en répartissant les points d'efforts parmi les différentes fonctionnalités et avons mis en place un task board en ligne pour que tout le monde puisse suivre l'avancement dans les détails du sprint en cours.

À la fin de la première itération, nous avons effectuer un sprint review pour déterminer les points sur lesquels nous pouvions encore nous améliorer du point de vu organisationnel et faire le point sur l'avancement de l'intégralité du projet. Nous avons aussi pu établir notre vélocité en nous basant sur la quantité totale de travail effectuée.

Notre fonctionnement lors des deux itérations suivantes est resté globalement le même, on peut cependant noter que la porosité des trois pôles est devenue de plus en plus importante à mesure que l'on a développé le projet, le travail nécessitant une vision de plus en plus globale. À chaque itération, nous avons donc effectué un sprint review pour faire le point suivit de l'établissement d'un sprint backlog afin de planifier les prochaines tâches sur lesquelles se concentrer.

8 Conclusion

Pour conclure, le projet que nous avons choisi était un projet très ambitieux avec beaucoup de travail à réaliser. En effet, nous n'avions aucune d'expérience sur la réalisation d'un tel projet du point de vue technique et il a donc fallut tout construire depuis zéro. En revanche, c'est cela qui fait la richesse et l'intérêt du sujet.

La méthode agile nous a permis de nous organiser de façons efficace malgré la taille du projet, aucun d'entre nous n'avaient travaillé avant sur un projet avec plus de membre. Ce projet nous à même permis de prendre pleinement conscience de l'intérêt de chacun des outils vus en cours.

Nous avons aussi eu la chance de collaborer au cours de ce projet avec Guohao, un élève d'origine chinoise qui a rejoins l'N7 au second semestre.