UTBM : Anthony Dury - Kadir Ercin - Romain Henry

Bases fondamentales de la programmation orientée objet - Pr. GECHTER

Analyse UML

Table des matières

Introduction

L'exercice a pour but de nous initier aux principaux concepts de la programmation orientée objet. Plus particulièrement de comprendre et de savoir mettre en oeuvre Java, ainsi que d'acquérir une méthode d'analyse retranscrite dans des diagrammes UML.

Il s'agissait ici d'implémenter soit un jeu de plateau, soit le nouveau jeu de steam "Mini Metro". Nous avons choisi d'adapter le jeu Mini Metro qui nous paraissait plus "user friendly", une adaptation qui ne sera donc pas personnalisé afin de préserver le gameplay.

Le principe de ce jeu est donc de construire un réseau de métro au fur et à mesure de la demande. Il faut donc dessiner des lignes entre les stations afin que les voyageurs puissent se rendre à leur destination. Mais attention, la ville se développe et de plus en plus de stations et d'utilisateurs apparaîtront à des positions aléatoires. La partie prend fin lorsqu'une station atteint son quota maximum de voyageur. De ce fait, nous pouvons et devons repenser notre réseau à chaque instant afin de maximiser son efficacité. Au fil du temps, des trains, des lignes, des ponts vous sont offerts pour vous aider. L'objectif est donc de répondre à un maximum de demandes.

Nous avons donc pris le jeu en main avant de procéder à son analyse en reprenant la norme UML. Ce document fait donc l'objet de description de cette analyse.

I. Cahier des charges

A. Choix des technologies

Dans un premier temps nous avons donc réfléchis sur l'organisation et la sémantique du projet. De ce fait nous avons créé un dépôt privé sur GitHub afin de faciliter le travail de groupe et de suivre aisément l'évolution du projet, point essentiel sûr pour mener à bien tout projet de développement.

L'implémentation de l'application sera réalisée en JAVA sous l'IDE IntelliJ IDEA.

Nous avons décidé d'utiliser la bibliothèque JavaFX pour gérer l'interface graphique utilisateur afin de découvrir et d'apprendre une autre bibliothèque différente de Swing qui devient obsolète.

Concernant la modélisation UML, nous avons réalisé les diagrammes avec le logiciel StarUML de MKLab.

B. Les fonctionnalités

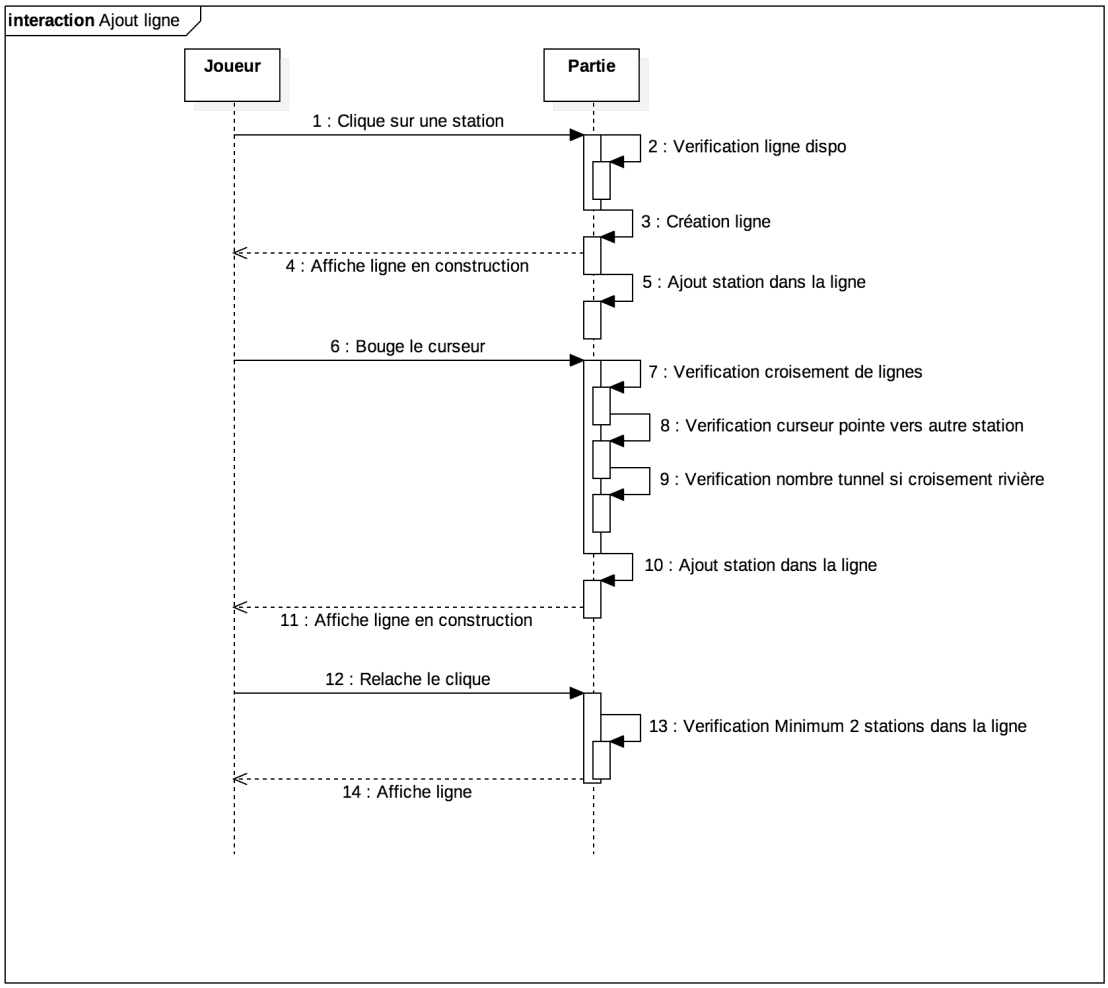
Les actions possibles de l'utilisateur peuvent être résumés par le diagramme de cas d'utilisation qui suit.

Nous nous concentrerons dans un premier temps sur l'implémentation des mécaniques de bases :

- Gestion des retraits et ajouts des lignes et des stations.
- Déplacements des clients et des trains.
- Apparition des clients et des stations.
- Utilisation d'une carte fixe et unique avec rivière impliquant la gestion des ponts.
- Gestion du temps.

Nous avons pris le choix de ne spécifier uniquement le fonctionnement des utilisations les plus complexes, à l'aide de diagrammes de séquence qui suivent.

- Ajouter une ligne :

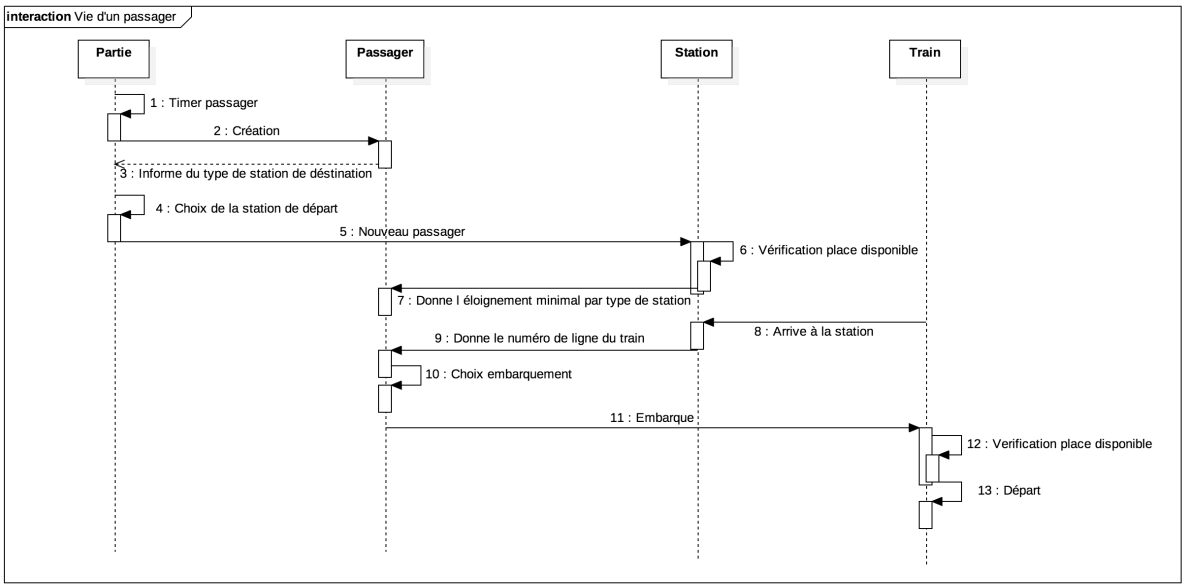


Nous allons utiliser le mécanisme de gestion des exceptions de JAVA afin d'identifier et traiter les erreurs.

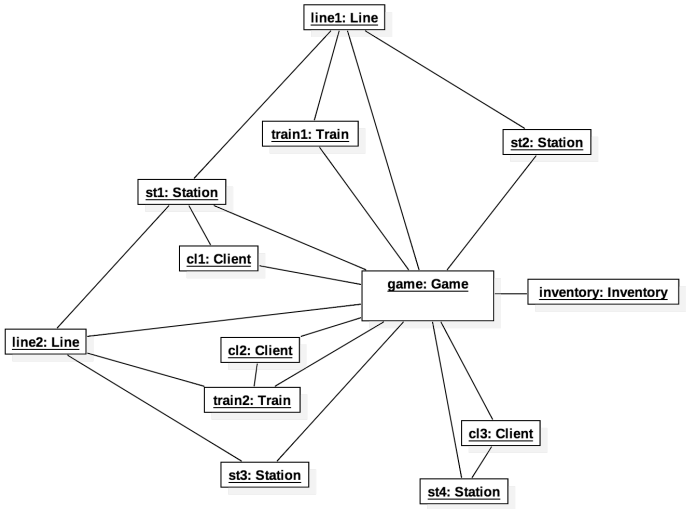
En fonction de notre avancement, voici une liste non exhaustive des fonctionnalités envisagées :

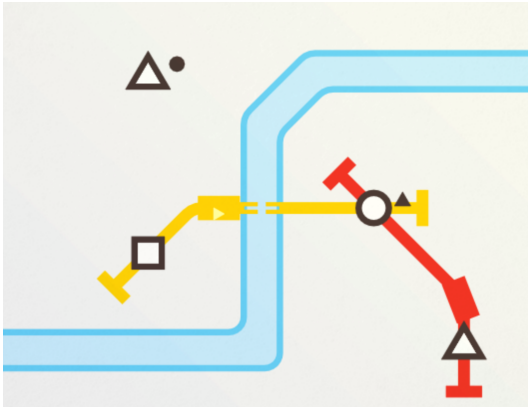
- Pouvoir jouer sur plusieurs cartes.
- Gérer le dézoom sur la carte lorsque le nombre de stations est élevé.
- Sauvegarder/charger une partie.
- Ajouter des fonctionnalités non existantes …

Afin de comprendre l'interaction entre les éléments de notre système nous avons produit un diagramme dynamique permettant de mieux comprendre la “vie d’un passager”



Afin d’exprimer un contexte d'exécution et d’avoir une idée des instances des classes à un moment T, nous avons produit un diagramme d’objet représentant le jeu avec différents cas voir ci-dessous.





- La ligne rouge *(line1)* contient une station “triangle” vide *(st2)*, une station “rond” *(st1)* avec un client “triangle” *(cl1)* et un train vide *(train1)*
- La ligne jaune *(line2)* contient une station “carré“ vide *(st3)*, une station “rond”*(st1)* avec un client “triangle” *(cl1)* et un train *(train2)* avec un client “triangle” *(cl2)*
- La station “triangle” contient un client “rond”

II. Organisation du projet

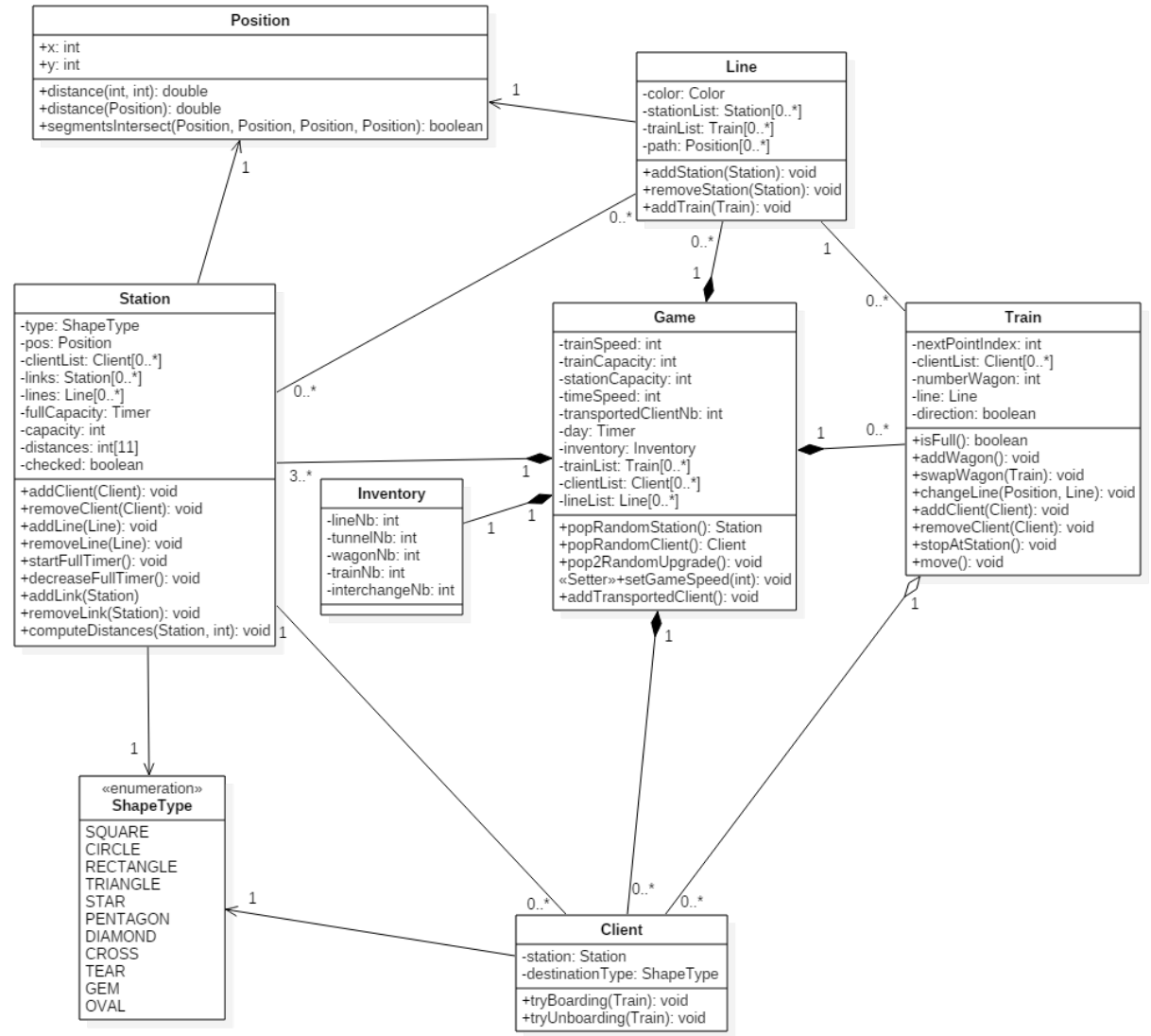
Nous avons décidé de créer des classes pour représenter les clients, les lignes, les trains, les stations.

Nous avons créé la classe Game qui permet de gérer le jeu, c'est à dire que cette classe contiendra les listes des stations, clients, lignes, trains et fera apparaître les clients, les stations et contiendra l'inventaire du joueur, qui est matérialisé par une classe.

Pour gérer le type des formes des clients et stations nous avons créé une énumération ShapeType.

Nous avons rajouté la classe position qui permettra de stocker des coordonnés entières (x, y) et qui gèrera tout ce qui concerne les intersections entre des droites, les distances etc …

Voici le diagramme de classe correspondant :

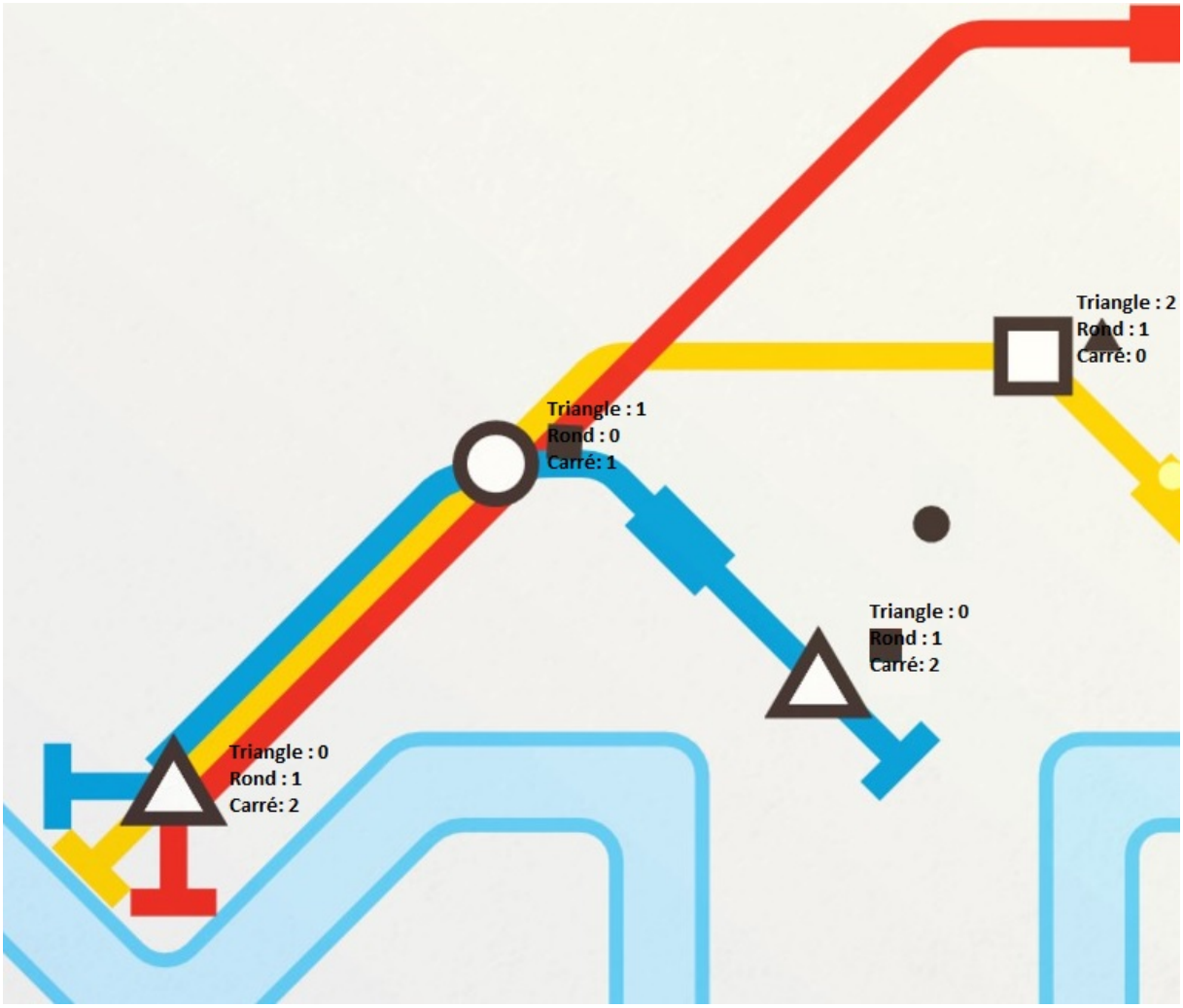


III. Déplacements des clients

Nous avons choisi une façon simple de gérer les déplacements des clients, lorsqu'un train arrive à une station, si la ligne du train contient une station du même type alors le client entre dans le train.

Sinon si le train s'approche (distance en termes de nombre de liens entre stations, pas de distance euclidienne) de la station la plus proche du type de station d'un client, alors ce client entre. Si aucun des deux cas n'est vérifié, le client n'entre pas, car il n'existe pas de chemin menant à une station de son type.

Voici un petit schéma facilitant la compréhension des distances utilisés pour notre projet :



Conclusion

L'utilisation des diagrammes UML pour modéliser le projet nous a permis dans un premier temps de définir un cahier des charges à l'aide du diagramme des cas d'utilisations. Cette étape a été importante pour la coordination de notre travail en groupe ainsi que pour poser les bases de notre application.

En s'appuyant sur ce diagramme, nous avons pu définir les interactions entre l'acteur principal (le joueur) et notre application sous la forme de plusieurs diagrammes de séquences. Ces diagrammes représentent les cas d'utilisations les plus complexes afin de nous guider lors de l'implémentation du projet.

Nous avons ensuite créé un diagramme de classe permettant de visualiser rapidement le modèle du projet afin de pouvoir réfléchir sur le fonctionnement (contrôleur) de notre application, dans le but d'éviter les problèmes de modélisations pouvant survenir durant l'implémentation. Nous avons aussi détaillé des aspects plus techniques, concernant la question de la gestion des déplacements des clients, qui semble être la difficulté du sujet.

La réalisation des diagrammes UML a donc été pour nous un outil essentiel pour organiser notre projet en groupe et visualiser la conception de notre application, ceci étant incontournable pour développer un projet s'appuyant sur la programmation orientée objet.