BA Demba
ZIHOUNE Bilal

Rapport de SAE (2.01 et 2.02)

Sommaire :

1. Introduction.
2. Qualité de développement.
3. Conception générale.
4. Conclusion
5. Annexes

    5.1. Lien vers notre dépôt git.
    5.2. Manuel d'utilisation.
    5.3. Maquette
    5.4. Diagramme de classe général.
    5.5. Diagramme de classe par composant.

1. Introduction.

Dans le cadre des SAE 2.01, 2.02 et 2.05, nous avons eu l'occasion de nous mettre dans la peau de vrais développeurs ayant pour objectif de mener à bien leur projet. Celui-ci était principalement centré autour de la résolution algorithmique d'un problème faisant appel au graphes et autour de la réalisation d'une application performante et accessible. En effet, le but était d'aider notre client à établir des itinéraires à partir de transactions fournies.

Dans notre application, il est possible d'ajouter de nouveaux scénarios, d'obtenir tous les itinéraires liés à un itinéraire et d'obtenir le plus rapide d'entre eux. Il est aussi possible de créer son propre itinéraire étape par étape, d'obtenir des informations sur la domiciliation des membres (leur ville d'origine) et de les regrouper par ville.

2. Qualité de développement.

2.1. L'organisation du travail.

L'organisation du travail pour ces SAE à été plutôt spéciale. En effet, j'ai (Demba) commencé la SAE tout seul. J'avais donc bien commencé. En raison de réorganisations des binômes, nous nous sommes retrouvés ensemble. Cela a rendu difficile l'adaptation et la répartition du travail.
Concrètement, je me suis occupé du codage de la partie modèle, contrôleur et d'une partie de la partie vue. Bilal a quant à lui participé à l'écriture des fonctions test, au codage de la partie vue et à l'UML.

2.2. Les outils utilisés

Tout au long de cette SAE, nous avons utilisé des outils comme git (avec bitbucket) et JUnit pour les tests. J'ai également essayé d'implémenter une API pour afficher une carte, une idée que j'ai bien vite abandonné.

2.3. Notre méthode de développement.

Nous avons essayé de mettre le plus possible en pratique le cours de Qualité de développement et ses enseignements sur le développement par les tests. C'est pour cela notamment que vous pouvez trouver un package test dans notre projet.

Cependant, pour être totalement honnête, cela n'a pas été le cas pour tous le développement, certaines méthodes ne possédant pas de tests et ces derniers pourraient être plus exhaustifs.

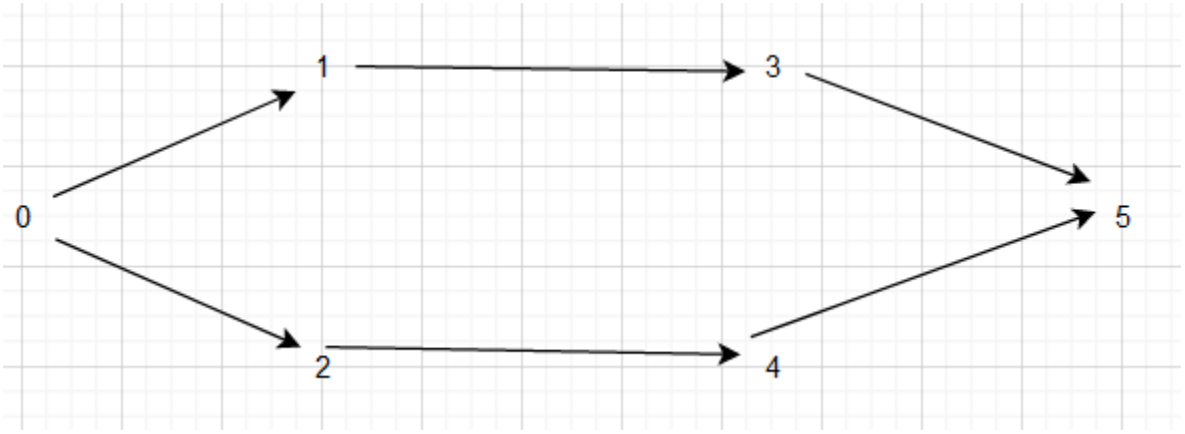
3. Conception générale.

Veuillez trouver les diagrammes de classe en Annexe.

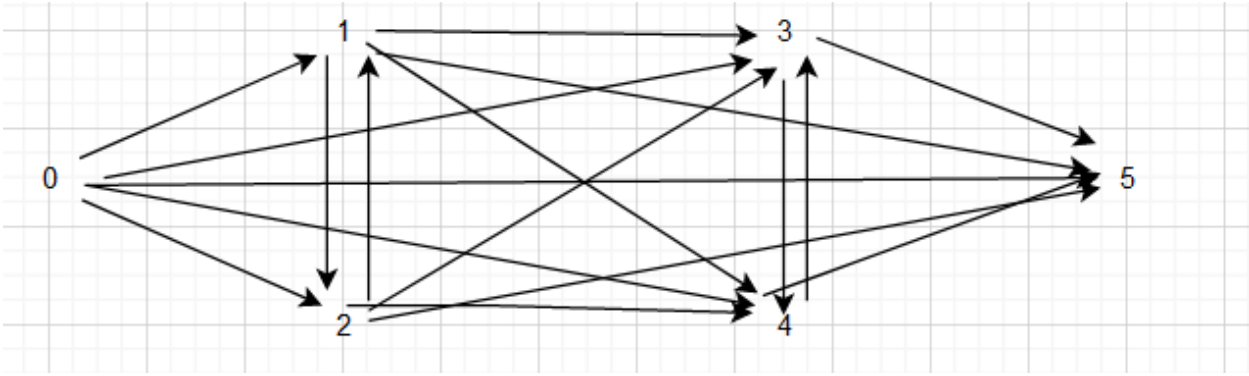
3.1. Structures et stratégies algorithmiques.

Le premier gros problème qui s'est présenté à nous à été la recherche de tous les chemins. La recherche des sources ayant été donnée, nous sommes partis du principe qu'il existait des niveaux de sources, c'est-à- dire des moment où il y avait plus d'un choix pour continuer sa route. Nous avons donc effectué des permutations entre les membres d'un niveau ce qui nous donnait un résultat en apparence correct mais incomplet.

Pour palier à ce problème, nous avons choisi la méthode brute force améliorée. Elle consiste à ajouter des arc entre les membres mais que vers l'avant. Par exemple :



Devient :



La condition pour qu'un arc entre i et j soit empruntable est que tous les sommets entrants de j soient présent dans le chemin courant. Il sera donc impossible de faire d'avoir un chemin commençant comme ceci : 0,3,... car si on part du chemin ne contenant que 0, il manque le sommet 1 pour accéder à 3.

Toutes ces opérations intermédiaires ne sont pas négligeables car pour le scénario 2_2, mon programme met environ 13 secondes à obtenir tous les chemins.
Les structures de données utilisées pour représenter un graphe sont 2 dictionnaires d'adjacence ayant pour clef un sommet (un membre) et pour valeur respectivement, la liste des voisins entrants et la liste des voisins sortants.

4. Conclusion

A mon avis, nous sommes allés plutôt loin dans le développement de notre application en ajoutant plusieurs fonctions supplémentaires. Cependant ce n'est bien évidemment pas parfait. L'un des gros défaut est le temps que le programme prend avec le dernier scénario. De plus, c'est vraiment une perspective d'amélioration car pour certains de mes camarades, l'obtention de tous les chemins du scénario 2_2 ne prend qu'environ 5 secondes.
Quelques autres voies d'amélioration sont possibles comme l'ajout d'une page permettant de créer soi-même un scénario.

5. Annexes

5.1. Lien vers notre dépôt git.

Veuillez le trouver ici .
https://bitbucket.org/bademba09/sae2_02/src/master/ .
Le mot de passe pour pouvoir le cloner : JrpYw4fUqe6tFzt6zPFg
La branche à cloner est master

5.2. Manuel d'utilisation.

Veuillez le trouver ici .
https://drive.google.com/file/d/1B-BQgFLQil_T7BR-d9Epk7XntoGG3bdg/view?usp=sharing

5.3. Maquette

Veuillez la trouver ici .
https://drive.google.com/file/d/1avs7IKNGEFU4cWt7hwPsgCFLvXku627Z/view?usp=sharing

5.4. Diagramme de classe général.

Veuillez trouver le diagramme général ici .
https://drive.google.com/file/d/1gUqL8zaDris6LoWyDspPW3uSjM3hzWWP/view?usp=sharing

5.5. Diagramme de classe par composant.

Veuillez trouver le diagramme du composant contrôleur ici .
https://drive.google.com/file/d/1EMbwOnuEeZ8OsyPU7L_dtLVFplOJatlp/view?usp=sharing

Veuillez trouver le diagramme du composant modèle ici.
https://drive.google.com/file/d/1TyFiH7TG78Rp9JATdXfToK7XAvj6eTRz/view?usp=sharing

Veuillez trouver le diagramme de la composante vue ici .
https://drive.google.com/file/d/14kRiDrbXD-ubo1ewnTuA-oQsA3N31p5d/view?usp=sharing

A noter que pour ce dernier diagramme, en raison d'un manque d'espace, les relations d'extension (l'héritage) ne sont pas représentées. Ainsi, les classes commençant par 'VBox' héritent d'une VBox, celles commençant par 'HBox' d'une HBox, celles commençant par 'GridPane' d'une GridPane et celles commençant par 'ComboBox' d'une ComboBox.
Aussi, la classe 'PageMain' hérite de Application.