Rapport du projet 1 : *Parking Escape*
cours d'Algorithmique 2 : INFO-F-203

Verhelst Théo Petit Robin

18 décembre 2015

Table des matières

1 Introduction

Ce document est le rapport relatif au projet du cours d'algorithmique 2 (*INFOF-203*) : *Parking Escape*. Nous commencerons par introduire le projet avec l'objectif de l'énoncé ainsi que les objectifs visés par l'implémentation. Ensuite seront discutés les choix concernant l'implémentation et la modélisation du problème. Pour finir, l'algorithme et l'implémentation seront détaillés avant de conclure.

1.1 Résumé de l'énoncé

En bref, résumons la consigne de l'énoncé.

Soit un parking P admettant une et une seule sortie. Soient $(n+1)$ voitures $\{v_G, v_1, v_2, \ldots, v_n\}$. Toutes ces voitures ont une orientation qui leur est associée (soit horizontale, soit verticale). L'objectif est d'amener la voiture v_G (appelée voiture *Goal*) jusqu'à la sortie du parking en respectant les déplacements relatifs à l'orientation de chaque voiture, à savoir : une voiture verticale ne peut se déplacer que vers le haut ou vers le bas et une voiture horizontale ne peut se déplacer que vers la gauche ou vers la droite.

Les informations concernant la disposition du parking pour la résolution sont passées en paramètre au programme à l'aide d'un fichier d'*input*. De plus, la sortie attendue du programme doit se faire à la fois sur l'*output* standard et dans un fichier d'*output* dans le cas où aucune solution n'est trouvée (en expliquant brièvement pourquoi aucune solution n'est accessible).

1.2 But du projet

Les objectifs de ce projet sont à la fois de faire travailler le langage Java vu au cours de Langage de Programmation 2 (INFOF-202) et travailler l'implémentation des algorithmes de théorie des graphe vus au cours d'Algorithmique 2. De plus, le travail étant réalisé en binôme, un objectif (secondaire) de ce projet est d'entamer le principe le travail de groupe qui sera à appliquer pour le projet d'année.

Une consigne du projet était de réaliser ce dernier en faisant bon usage des concepts de la programmation orienté-objet, le langage obligeant (Java est **uniquement** OO : tout est objet).

2 Choix de représentation

Les seules consignes concernant l'implémentation étaient de réaliser le programme en Java et en orienté objet. Le choix des classes à implémenter était dès lors totalement laissé aux étudiants. Cette section a pour objectif d'expliquer et de détailler les choix faits dans le cadre de notre implémentation.

Le programme est représenté par un *package* contenant trois classes principales : **Graph**, **Situation**, et **IOManager** ainsi que deux classes périphériques : **Main** et **SolutionNotFoundException**.

Ces deux dernières ne servent qu'à lancer le programme pour le premier (ainsi que maintenir une batterie de tests) et la seconde est une exception définie dans le cadre du package. Il y a dès lors peu de contenu à l'intérieur de celles-ci. Les trois premières quant à elles détiennent le cœur du programme.

2.1 Centralisation dans la classe `Situation`

La classe `Situation` est la classe contenant toute la représentation interne du parking ainsi que tout le traitement relatif aux modifications de l'état de ce dernier : mouvements des voitures, gestion des permissions de mouvements, gestion des voitures bloquant les mouvements des autres, etc.

C'est la classe la plus importante du point de vue des proportions car c'est celle qui contient la codification utilisée par les autres.

2.2 Séparation des I/O dans une classe `IOManager`

La classe `IOManager` est une classe contenant exclusivement des méthodes statiques. En effet, le seul but de cette classe est de servir d'interface entre le programme chargé de résoudre le problème donné et les fichiers chargés de définir la situation initiale ou de décrire la situation finale. C'est donc au sein de cette classe que le parsing du fichier d'input est fait dans le but de créer la situation de départ décrite dans le fichier d'input. C'est également dans cette classe que sont réalisés les formatages pour l'écriture lors de l'achèvement du programme.

2.3 Classe de résolution algorithmique

La dernière classe, à savoir `Graph` contient la structure algorithmique relative au parcours du graphe afin de déterminer la situation sur base d'un algorithme décrit dans la section suivante.

2.4 Manipulation d'un *arbre* lors de la génération des solutions

Suite à une discussion avec M. Fortz, titulaire du cours INFOF-203, à propos du projet, il nous a été conseillé de ne générer le graphe que lorsque c'est nécessaire, donc de ne pas prendre en compte les nœuds ne nous intéressant pas dans la résolution. À savoir : il est inutile (*a priori*) de déplacer une voiture placée dans un coin si elle ne gène aucune autre voiture dans l'immédiat.

La structure de données que nous manipulons donc dans ce programme est donc un arbre qui est parcouru par niveaux (parcours en largeur d'un graphe) comme expliqué dans la section suivante.

3 Algorithme

3.1 Choix de l'algorithme

3.2 Explications