POLYTECHNIQUE MONTRÉAL

LE GÉNIE EN PREMIÈRE CLASSE

Log2810 : Structures discrètes

TP2 : Automates et langages

Trimestre : Automne 2019 Équipe 61

Équipier 1 : Kevin Leumassi - 1892002

Équipier 2 : Merwane Chalabi - 1889346

Équipier 3 : Gabriel Thibaud – 1888604

Date de remise : 3 décembre 2019

Polytechnique Montréal

Introduction :

Dans ce TP, il nous est demandé d'implémenter une méthode plus précise pour aller chercher des objets particuliers dans le contexte suivant : on attribuera à chaque objet un nom, un type (A, B ou C) et un code identificateur (hexadécimal à 6 caractères). Le code identificateur est unique aux objets, alors que plusieurs objets peuvent avoir le même nom ou le même type. Chaque objet aura aussi un poids de la même manière que lors du premier TP. Par la suite, on implémentera un programme qui permettra au personnel de facilement pouvoir faire l'inventaire de leur stock en plus de pouvoir effectuer une commande d'objets aux robots.

Présentations des travaux et notre solution :

Notre travail est organisé de la sorte : nous avons 7 classes pour gérer les différents éléments.

Les classes Interface, Button et Menu servent pour la création et l'utilisation de l'interface.

La classe Objet correspond aux objets que l'on trouve sur le fichier « texte.txt », nous avons donc décidé d'instancier des objets Objets lors de la lecture de ce fichier afin de pouvoir les manipuler au mieux.

La classe Node permet d'instancier des nœuds (« nodes ») que nous utilisons dans la fonction de suggestion, en effet chaque nœud correspond à une lettre et plus l'utilisateur entre des lettres pour former un mot, plus il avance profondément dans l'arbre (il s'éloigne de la racine). Les nœuds sont composés d'une liste de noeudsAdjacents qui correspondent aux nœuds-enfants, d'une liste d'objets qui correspondent à tous les objets que le programme peut suggérer rendu au nœud en question.

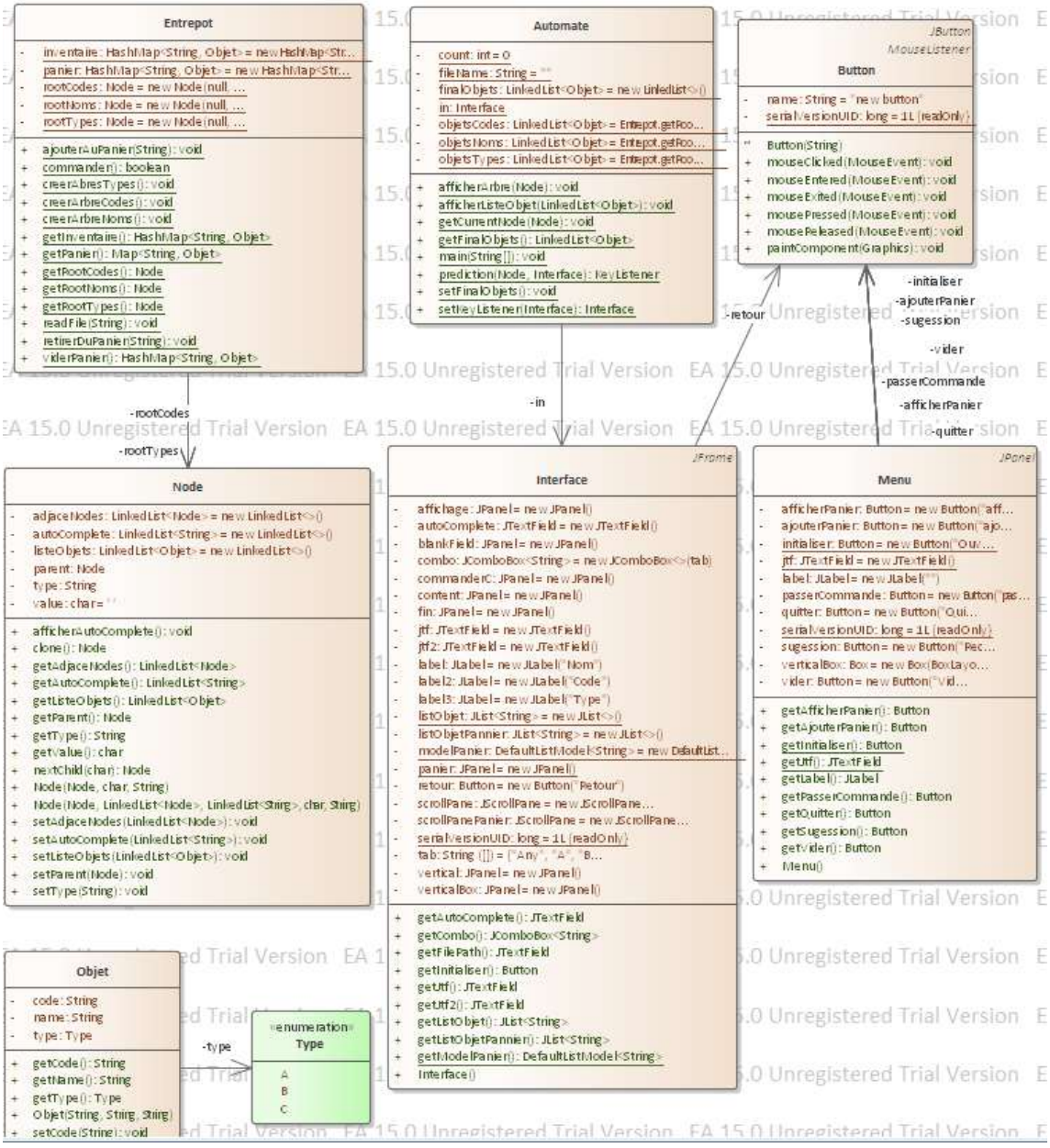
La classe Entrepôt est une classe principale qui, comme son nom l'indique, représente l'entrepôt du TP, la lecture du fichier, la création des arbres y est également effectuée. Cette classe possède des attributs tels que panier et

inventaire qui correspondent respectivement au panier qui contient les objets qu'un client souhaite acheter et l'inventaire que l'entrepôt possède et qui répertorie leur stock. Puisque nous avons le panier dans cette classe, nous avons donc les fonctions en lien avec une commande, en lien avec les ajouts et les retraits du panier.

La classe Automate regroupe les fonctions et attributs en rapport avec la fonctionnalité de suggestion, l'autocomplétions y est aussi implémentée.

Pour l'initialisation, nous lisons le fichier .txt et, lors de la lecture, nous créons des objets (qui correspondent aux objets du fichier) et nous les ajoutons dans notre conteneur « inventaire ». Ensuite nous créons 3 arbres (composés de nœuds) qui correspondent aux manières de rechercher dans la fonction de suggestion (par nom, par code et par type).

Diagramme de classes :



Difficultés rencontrées et nos éventuelles solutions :

La principale difficulté rencontrée lors de ce TP a été la fonction de suggestion. Nous avons opté pour une recherche via un arbre de nœuds. Chaque nœud correspond à une lettre (pour la recherche par nom), à un caractère du code (pour la recherche par code hexadécimal) ou à un type d'objet (pour la recherche par types). Nous avons une racine(root) qui correspond à un nœud initial qui a pour nœuds-enfants les différentes premières lettres des objets (par exemple dans inventaire.txt : il y a 5 premières lettres différentes A, B, C, D et H) et ainsi de suite, le nœud A possède autant de nœuds-enfants qu'il y a de 2^e lettres différentes pour les mots commençant par A. De cette manière, plus l'utilisateur entre de caractères plus la recherche se précise et moins de possibilité d'objets s'offrent à lui (la recherche s'affine). Si l'utilisateur arrête de taper, le programme lui propose tous les objets qui commencent avec ses lettres (s'il s'agit d'une recherche par nom). Si on applique ça à l'arbre, à partir du nœud qui correspond à la dernière entrée de l'utilisateur, on affiche toutes les possibilités possibles qui sont données par ce nœud avec tous ses nœuds-enfants.

De plus, lors d'une recherche, l'utilisateur doit pouvoir utiliser les trois types de recherches à la fois. Nous avons donc dû trouver un moyen pour afficher des suggestions qui sont en adéquation avec les trois recherches (pour reprendre l'exemple de l'énoncé, si nous saisissons A pour le nom, H pour le code et A pour le type, votre programme devra afficher tous les objets dont le nom commence par A, le code commence par H et dont le type est A).

Conclusion :

Ce premier TP nous a permis encore une fois de mettre en pratique les notions que nous avons étudiées en cours ainsi que la programmation en java, langage avec lequel nous sommes plus à l'aise par rapport au premier TP. Nous avons surtout pu concrétiser nos connaissances à propos des automates à états finis ainsi que leur fonctionnement que nous avons pu nous même implémenter avec la difficulté imposée du TP qui était la recherche entre plusieurs critères à la fois. Nous aimerions au futur éventuellement approfondir nos connaissances à propos de l'utilisation des automates en programmation, utilisation qui pourrait nous être utile lors de l'implémentation d'algorithmes