École Polytechnique de Montréal
Département de génie informatique et génie logiciel

LOG2810 : Structures discrètes

Tp2 : Automates – Hiver 2018

Soumis par :

Syphax Ait Yahia 1776868

Georges Louis 1880098

Bassam Ajam 1777689

Groupe 01

17 avril 2018

Introduction

Objectif : l'objectif de ce travail pratique est de se familiariser avec les notions des automates en appliquant des méthodes concrètes.

Mise en situation : Dans notre cas, on utilise les automates dans le concept où on doit lire des règles, qui nous sont déjà fourni par des fichiers « txt », et qui seront nôtres automates (le code qui va faire ouvrir les coffres-forts), puis après on lit des variantes, qui nous sont déjà fourni par des fichiers « txt », et qui seront tester pour avoir le mot de passe correcte des coffres-forts et de l'afficher.

Dans notre cas, on a utilisé la programmation orienté objet avec Java qui est la meilleure façon de résoudre un problème parce qu'on donne à chaque donnée une classe. Ces classes seront plus tard reliés ensemble pour résoudre le problème au complet.

Présentation des travaux

Dans notre cas, on a utilisé plusieurs classes qui représente des données différentes mais reliées ensemble. On peut voir ci-dessous les classes qu'on a décidé d'utiliser :

- Classe Automate :

C'est la classe où on définit un automate qui suit une suite de caractère spécifique. Elle contient les fonctions suivantes :

1. Automoate() : constructeur par défaut pour initialiser ces attributs (ArrayList<String> etats_ et index_).

2. Automate(ArrayList<String> etat, int index) : constructeur par paramètre pour initialiser ces attributs selon les paramètres.

3. CreerAutomate(String nomDeFichier) : cette fonction permet de créer une règle à partir d'un fichier dont le nom est passer en paramètre. Tous les états possibles dans le fichier, qui se trouve après le symbole « = », sont stocker dans notre attribut « etats_ ». Cela est fait pour qu'on puisse comparer plus tard (dans la classe Variante) une règle avec une variante (si les deux ont le même index_, ce qui est le premier chiffre dans chaque fichier et qui représente que la regles1 peut être seulement fait avec variantes1), alors on a besoin d'avoir tous les règles et tous les variantes pour effectuer ce travail.

4. ArrayList<String> getEtats_() : retourne notre attribut « etats_ ».

5. setEtats_(ArrayList<String> etats_) : on remplace notre attribut « etats_ » par le paramètre « etats_ »

6. getIndex_() : retourne notre attribut « index_ »

7. setIndex_(int index_) : on remplace la valeur de notre attribut « index_ » par la valeur en paramètre « index_ »

- Classe Variantes :
    1. Variante() : constructeur par défaut pour initialiser ces attributs (ArrayList<String> variantes_ et index_).

    2. Variante(ArrayList<String> variantes, int index) : constructeur par paramètre pour initialiser ces attributs selon les paramètres.

    3. traiterLesEntrees(String nomDeFichier) : cette fonction permet de stocker dans notre attribut « variantes_ » tous les variantes possible. Cela est fait pour qu'on puisse comparer tous ces variantes avec la règle correspondant (celle qui contient le même index_), alors on a besoin d'avoir toutes les variantes et tous les règles pour effectuer ce travail.

    4. getVariantes_() : retourne la valeur de l'attribut « variantes_ ».

    5. setVariantes_(ArrayList<String> variantes_) : on remplace la valeur de notre attribut « variantes_ » par la valeur passer en paramètre « variantes_ »

    6. getIndex_() : retourne la valeur de l'attribut « index_ »

    7. setIndex_(int index_) : on remplace la valeur de l'attribut « index_ » par le paramètre « index_ ».

- Classe MotDePasse :
    1. MotDePasse() : constructeur par défaut de la classe.

    2. MotDePasse(String s, int i) : constructeur par paramètre qui initialise les attributs par les valeurs passer en paramètres.

    3. getMot_() : fonction qui retourne l'attribut « mot_ »

    4. setMot_(String mot_) : on remplace la valeur de l'attribut « mot_ » par la valeur passer en paramètre.

    5. getIndex_() : fonction qui retourne l'attribut « index_ »

    6. setIndex_(int index_) : on remplace la valeur de l'attribut « index_ » par la valeur passer en paramètre

- Classe Traitement :
    1. Traitement() : constructeur par défaut

2. Traitement(Automate automate, Variante v) : constructeur par paramètre.

3. getAutomate_() : retourne la valeur de l'attribut automate_

4. getVariante_() : retourne la valeur de l'attribut variante_

5. setAutomate_(Automate automate) : on remplace la valeur de l'attribut « automate_ » par la valeur passer en paramètre « automate »

6. setVariante_(Variante variante_) : on remplace la valeur de l'attribut « variante_ » par la valeur passer en paramètre « variante »

7. trouverMotDePasse(Variante var) : c'est la fonction qui compare les variantes et les règles déjà stocker selon leur « index_ ». Le travail s'effectue si « index_ » de la règle et « index_ » de la variante sont égaux. Dans tous les cas, un message sera affiché pour savoir si les entrés sont bien faits ou non et si on a trouver le mot de passe.

8. afficherLesMotsDePasse() : fonction qui affiche tous les mot de passe déjà rencontrer.

- Classe Menu :

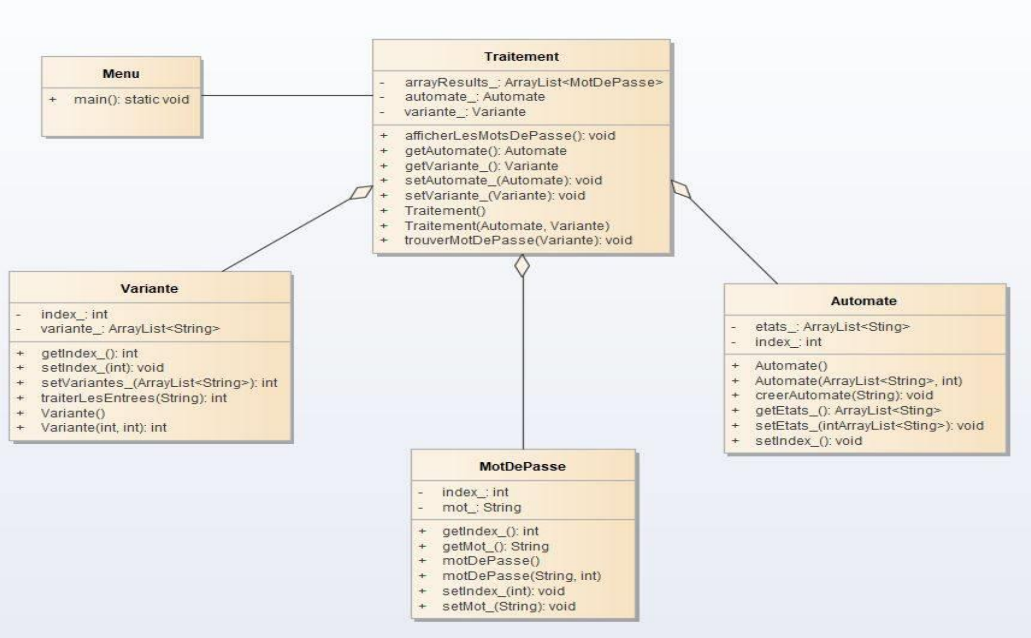
  C'est la classe qui contient le « main » et qui affiche les options possible à l'usager qui sont les suivantes :

  a) Creer l'automate
  b) Traiter les requetes
  c) Afficher les mots de passe valides obtenus
  d) Quitter

  Selon l'option choisit, les fonctions appropriés seront appelées pour effectuer le travail requis.

Diagramme de classes :

Fig1.



Difficultés rencontrées

Les difficultés rencontrées sont généralement sur le temps. Toutes les TPs d'autre cours étaient à remettre en même temps que ce TP. En plus, on avait le mini contrôle 3 de ce cours dans la même journée que la remise du TP, ce qui ne nous a pas laissé assez de temps pour nous préparer à l'examen de mini contrôle 3.
Mais les exigences sont moins complexes que celles du TP1.

Conclusion

Nous sommes satisfaits de ce TP puisqu'on a appris le concept d'automate qui est un sujet important dans notre domaine. Dans ce travail pratique nous avons eu la liberté de structurer notre code comme nous le voulons encore une fois comme le TP1, sans se baser sur un ancien code et prendre la responsabilité de notre code. Ce n'était pas trop long mais la date de remise était en même temps avec tous les autres cours et le mini contrôle 3. On espère pour les prochaines session vous réglez ce problème de temps.