Programmation avancée et application en Java

Rapport de projet de débat phase 1

L3 Informatique appliquée 2022-2023

Groupe de TD 1

MABROUK Fayez & AHMED-ZAID Macyl

1 Introduction

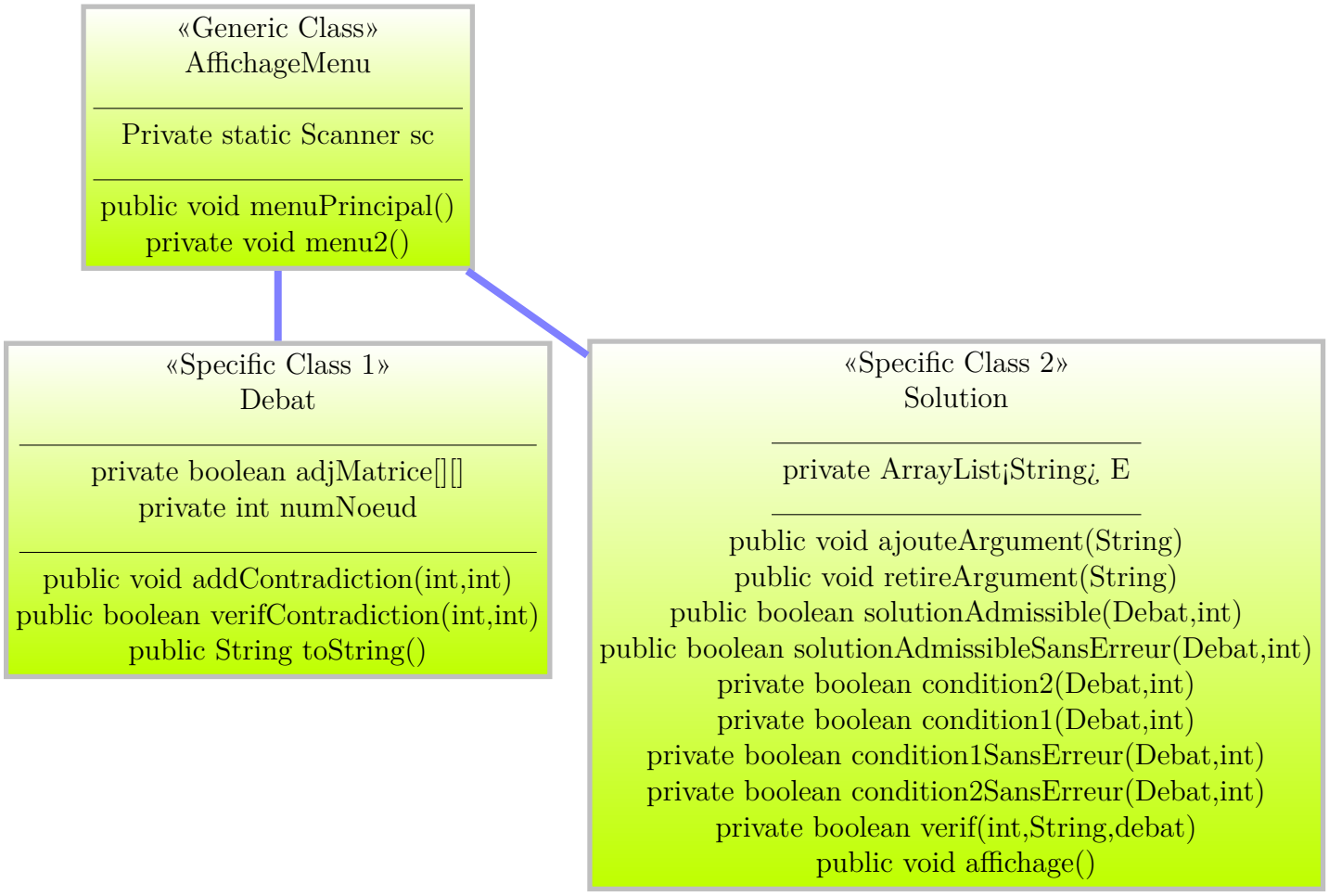
Notre projet consiste à proposer des solutions à un certain débat en se basant sur les arguments de personnes qui se contredisent.
Ce document contiendra 2 parties explicatives :

— La première qui définira les démarches de programmation suivies en représentant un diagramme de classe.

— La deuxième listera les différents packages utilisés, les classes, et expliquera la fonction de chaque méthode.

2 Démarche de programmation

2.1 Diagramme de classe



3 Documentation technique

3.1 La classe Main

La classe **main** fait appel seulement à l'affichage du menu.

3.2 La classe affichage

La classe **affichageMenu** est la classe principale qui permet d'afficher le menu et exécuter les différentes requêtes de l'utilisateur.Elle comporte deux méthodes :

* **menuPrincipal** : Affiche le premier menu pour ajouter les contradictions entre différents arguments.
* **menu2** : Affiche le menu secondaire qui permet d'ajouter un argument dans la solution ou bien le retirer et aussi vérifier la solution.

3.3 La classe Debat

Cette classe représente le graphe que nous utilisons pour définir les différentes contradictions entre arguments, elle comporte deux méthodes :

* **addContradiction** : Elle prend en paramètres le numéro de l'argument et met un vrai dans la matrice d'adjacence du graphe à l'emplacement souhaité.
* **verifContradiction** : Permet de vérifier si deux arguments se contredisent en renvoyant la valeur du booléen à l'emplacement passé en paramètres.

3.4 La classe solution

Cette classe permet de gérer le second menu, elle comporte 4 fonctions principales :

* **ajouteArgument** : ajouteArgument : Ajoute un argument dans la liste E (Ensemble des solutions admissibles) en vérifiant si l'argument est déjà dans la liste .
* **retireArgument** :Permet de retirer un argument de l'ensemble.
* **solutionAdmissible** : Permet de vérifier si un argument est une solution admissible cela en exécutant deux méthodes qui permettent de tester si un argument est une solution admissible :
  - **condition1** : Vérifie si dans le graphe nous avons deux arguments qui se contredisent cela en parcourant le graphe et vérifie si les emplacements (i,j) et (j,i) sont tous les deux a vrai.
  - **condition2** : Vérifie si pour un argument A qui contredit un élément de la liste de E il existe un élément de E qui puisse contredire l'argument A cela en parcourant la matrice d'adjacence on fixe un élément de la matrice, on recherche un élément qui est contredit par cet élément qui est fixé si on en trouve un on cherche un autreé élément qui contredit ce premier sinon on retourne faux.
* **solutionAdmissibleSansErreur** : Si l'utilisateur choisie l'option 4 (quitter le programme) on exécute le deux méthodes condtion1SansErreur qui est semblable à condtion1 sans affichage d'erreur et condtion2SansErreur.

4 Conclusion

Les classes principales du projet ont été définies correctement durant cette première phase ainsi que les méthodes. Nous avons un programme qui est fonctionnel pour débuter la seconde partie qui consiste à la recherche de solution admissible et préférée.