le cnam
Grand Est

Rapport de projet : Projet Java

LAMBOTTE Arthur, HENAUX Quentin, COURTOT Thomas
et SOL Guillaume

25 février 2021

Enseignant référent : SADDEM Ramla

Table des matières

1 Présentation du projet

1.1 Quels sont les attentes ?

Notre projet consiste à créer un programme pour extraire des Signatures Temporelle Causale (STC) depuis un fichier excel de manière automatisée. Il faudra ensuite afficher le graphe correspondant et indiquer les parties conditions des STC. Le projet se fera entièrement en Java et sera développé via l'IDE éclipse.

1.2 Contexte du Projet

Le projet est né d'une volonté d'automatiser la création des Signatures Temporelle Causale afin de faire gagner du temps aux experts qui auront pour tâche d'analyser ses dernières pour diagnostiquer l'état du système.

1.3 Quel est l'existant ?

Le projet ne possède pas d'existant, néanmoins il est mis à notre disposition un fichier excel nous permettant de développer notre programme.

1.4 Comité de pilotage

MOE

Nom	rôle
SADDEM Ramla	Enseignante référente du projet

Équipe de projet

Nom	rôle
LAMBOTTE Arthur	Développeur
SOL Guillaume	Développeur
HENAUX Quentin	Développeur
COURTOT Thomas	Développeur

1.5 Objectifs du Programme

Objectif Global

Suppression des éléments superflus et des doublons

L'excel qui nous est fourni comporte des éléments superflus pour nous, le premier objectif sera donc de les retirer pour nous permettre d'avoir un fichier prêt à l'emploi. Il nous faudra également convertir les dates si l'on veut pouvoir les utiliser pour la partie temporelle du STC.

Qui plus est, il nous faudra également retirer les doublons qui ne nous seront pas utiles. Un doublon correspond à 2 lignes consécutives ayant le même état, c'est à dire qu'aucun événement est survenu entre les 2 lignes.

Calcul de la liste d'événement et affichage du graphe temporel

Une fois les données de l'excel prêtes à l'emploi, il va falloir comparer les lignes 2 à 2 afin d'extraire les événements qui s'y sont produits ainsi que la contrainte de temps. Cela revient à vérifier si un élément est passé de 0 à 1 ou au contraire de 1 à 0, et de relever l'écart de temps entre les 2 dates d'enregistrement de l'état.

Il nous faudra ensuite afficher le graphe temporel correspondant afin de pouvoir constater les différents événements dans le temps.

Extraction de la partie conditionnel

Ceci est l'étape finale du programme, il va falloir extraire les parties conditions du graphe temporel généré à l'étape précédente.

2 Volet technique

2.1 Développement

Gestion du projet

Après avoir analysé le sujet et compris les attentes de ce dernier. Nous avons décidé de diviser notre développement en différentes classes et de nous répartir le travail en fonction des questions tout en sachant que chaque question dépend de la précédente mais que nous pouvons générer des tests avec 4 lignes pour tester les bases de nos algorithmes. Cependant, dès que l'un d'entre nous rencontrait une difficulté tout le monde était partant pour l'aider, ce qui a permis un développement rapide et efficace.

Le versioning ainsi que le développement du projet ont été réalisé à l'aide de gitHub ce qui a permis un échange de code efficace entre nous et un développement structuré.

Dans toutes nos classes nous avons décidé de mettre nos attributs en "private" dans le but de favoriser une meilleure sécurité et stabilité de notre code.

Afin de modéliser le graphe temporel de façon plus explicite nous avons décidé d'utiliser la librairie Guava. Guava est un ensemble de bibliothèques open source en Java qui est principalement développé par les ingénieurs de Google. La librairie permet entre autre la gestion des graphes, la gestion du cache, mets à disposition des fonctions de hashage,...

Installation de la librairie Guava avec Eclipse

Pour utiliser notre projet, il est nécessaire d'utiliser la librairie guava et d'installer le package sur la machine qui va exécuter notre code. Si vous effectuez un clone de notre projet, disponible sur notre gitHub, et que vous ouvrez le projet avec eclipse le package est normalement déjà enregistré mais si vous décidez de juste prendre une partie de notre code ou d'utiliser un autre IDE voici la marche à suivre. Le fichier Jar de la librairie est disponible sur notre gitHub dont le lien est spécifié en annexe avec le chemin suivant : "/code/src/assets/guava-30.1-jre".
Voici les étapes pour l'ajouter via Eclipse :

- Faire un clique droit sur "Projet_Java" dans le Package Explorer
- Choisir "Build Path"
- Appuyer sur "Configure Build Path..."
- Aller dans "Librairies"
- Appuyer sur "Add External JARs..."
- Appuyer sur Apply and Close
- Enfin redémarrer Eclipse pour que le changement soit pris en compte

Répartition du travail de façon globale

Nom	question traitée
SOL Guillaume	1
LAMBOTTE Arthur	2
COURTOT Thomas	3
HENAUX Quentin	4
Tout le monde	5

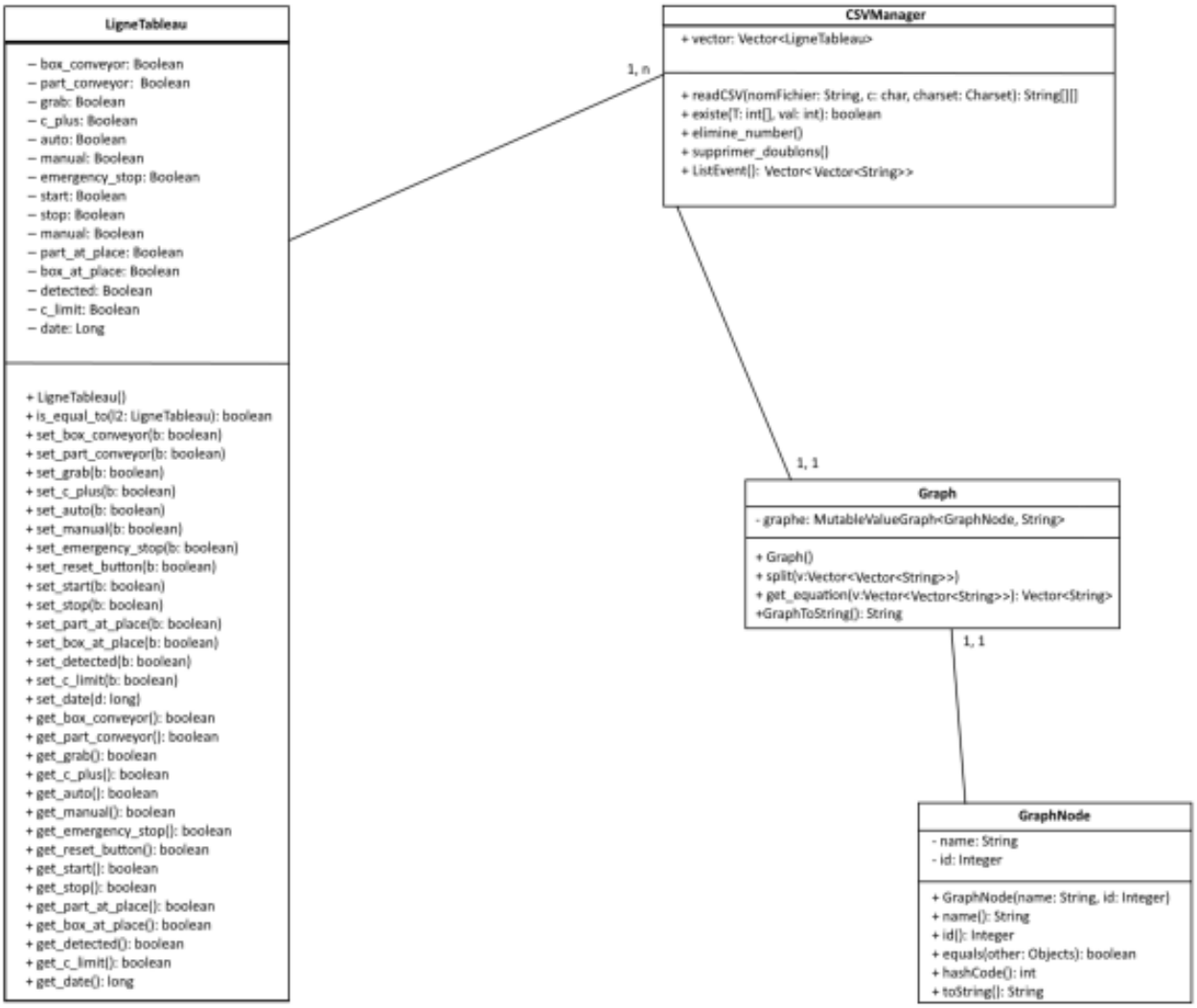

Schéma UML du projet

2.1.1 Classe Main

Cette classe permet d'exécuter le programme. Elle va créer les différents objets nécessaires à partir des autres classes du projet et appeler les méthodes afin d'effectuer le traitement attendu. C'est également elle qui va afficher le rendu sur la console.

2.1.2 Classe LigneTableau

La classe LigneTableau permet de reproduire l'organisation du fichier excel. Elle possède 15 attributs privés qui sont :

- box_conveyor / Boolean
- part_conveyor / Boolean
- grab / Boolean
- c_plus / Boolean
- auto / Boolean
- manual / Boolean
- emergency_stop / Boolean
- start / Boolean
- stop / Boolean
- manual / Boolean
- part_at_place / Boolean
- box_a_tplace / Boolean
- detected / Boolean
- c_limit / Boolean
- date / Long

Chaque attribut correspond à une des colonnes de l'excel, nous avons fait le choix de stocker la date sous forme de long en la mettant directement en nanoseconde afin de nous faciliter les calculs par la suite. La méthode possède tous les accesseurs et modificateurs nécessaires pour ses attributs et ne possède pas d'autres méthodes.

2.1.3 Classe CSVManager

La classe CSVManager est une des classes les plus importantes de notre projet car elle va permettre le traitement du fichier excel. Elle possède 1 attribut :

— Vector / Vector <LigneTableau>

Cette classe possède 5 méthodes dont 3 importantes :

— La méthode elimine_number qui permet de supprimer les éléments qui ne nous seront pas utiles lors de l'exploitation du fichier excel et de convertir les dates en nanoseconde. Cette méthode va peupler l'attribut vector de LigneTableau. Cette méthode ne retourne rien et ne prend pas de paramètre.

— La méthode supprimer_doublons qui permet de supprimer les lignes qui sont en double. C'est à dire les lignes consécutives qui possèdent le même état et ou aucun événement n'est survenu entre. Si il y a juste 2 lignes identiques on les conserve mais si plusieurs lignes consécutives sont égales on conserve uniquement la première et la dernière afin d'avoir la contrainte de temps. Cette méthode ne retourne rien et ne prend pas de paramètre.

— La méthode ListEvent qui permet de retourner la liste des événements avec la différence de temps survenue entre 2 lignes consécutives. Pour cela on vérifie pour chaque élément qu'il n'y a pas eu un changement d'état, c'est à dire soit un passage de 0 à 1 soit un passage de 1 à 0. Cette méthode retourne un Vector<Vector<String».

— La méthode readCSV qui permet d'extraire les donées du fichier excel, et qqui les retourne à la méthode elimine_numbe. Cette méthode retourne un tableau de tableau de String noté String[][] et prend en paramètre un string qui correspond au chemin du fichier, un char qui permet de délimiter les lignes et un charset qui correspond à la méthode d'extraction.

— La méthode existe qui permet permet de vérifier qu'une valeur se trouve déjà dans un tableau. Elle retourne true si la valeur est dans le tableau et sinon elle retourne false, et prend en paramètre un tableau de int et un int.

2.1.4 Classe GraphNode

La classe GraphNode sert à modéliser un sommet du graphe, elle possède 2 attributs :

— id / Integer
— nom / String

La classe possède 5 méthodes, dont 2 sont les accesseurs des attributs et les 3 autres sont :

— La méthode equals qui prend en paramètre un objets de type other et retourne un boolean. Cette méthode vérifie si 2 graphe sont identique. Elle retourne true si l'objet est le même et false sinon.
— La méthode hashcode qui ne prend pas de paramètre et qui retourne un entier. Cette méthode créée et retourne un hashage afin de rendre le sommet unique en utilisant l'id et le nom.
— La méthode toString qui prend pas de paramètre et qui retourne une chaîne de caractère. Cette méthode permet de renvoyer le nom du graphe entouré de parenthèse afin de simplifier son affichage à l'utilisateur.

2.1.5 Classe Graph

La classe condition sert à modéliser le graphe, elle possède 1 attribut :

— graphe / MutableValueGraph <GraphNode, String>

Cette attribut indique que les sommets sont de type GraphNode et que les valeurs des arcs sont de type String. La classe possède 3 méthodes qui sont :

— La méthode split qui prend en paramètre un Vector< Vector<String> > et qui ne retourne rien. Cette méthode permet de remplir le graphe à partir de la liste des événements, traités dans la classe CSVManager par la méthode ListEvent.
— La méthode GraphToString qui ne prends pas de paramètre et qui retourne un String. Cette méthode permet de modéliser le graphe sous la forme :
< (sommet1) -> (sommet2) >= valeur de l'arc.
— La méthode get_equation qui prend en paramètre un Vector< Vector<String> > et qui retourne un Vector<String>. Cette méthode permet d'obtenir les équations à partir de la liste des événements, traités dans la classe CSVManager par la méthode ListEvent ainsi que du graphe traité par la méthode split.

3 Annexe

3.1 gitHub

Le projet est disponible sur ce dépo github : `https://github.com/guiwil17/Projet_java`

3.2 Présentation

Notre présentation a été ralisée grâce à prezzi et est disponible à cette adresse : `https://prezi.com/view/N50j6zOc879mThemBILN/`