

Rapport de Stage

Analyse cybersécurité



Réalisé par

Kévin GIORDANI

Sous la direction de

Trevor Martin

Pour l'obtention du DUT Informatique

Année universitaire 2015 - 2016

Remerciements

Je tiens à remercier toutes les personnes qui ont contribué au succès de mon stage et qui m'ont aidé lors de la rédaction de ce rapport.

Tout d'abord, j'adresse mes remerciements à mon professeur, M GARCIA Francis qui m'a aidé dans ma recherche de stage et m'a permis de postuler dans cette université. Ses propositions m'ont permis de trouver ce stage à l'étranger qui était en totale adéquation avec mes attentes.

Je tiens à remercier vivement mon maitre de stage, M TREVOR Martin, professeur en intelligence artificielle a l'université de bristol, pour son accueil, et son expertise. Grâce à sa confiance j'ai pu m'accomplir dans mes missions. Il fut d'une aide précieuse dans les moments les plus délicats.

Je remercie également toute l'équipe des relations internationales de l'IUT pour leur aide, et leurs attentions durant la préparation et la durée de mon stage.

Enfin, je tiens à remercier toutes les personnes qui m'ont conseillé et relu lors de la rédaction de ce rapport de stage.

Sommaire

Introduction.....	page 1
1 — Présentation de l'entreprise	page 2
1.1 Création et évolution de l'entreprise.....	page 2
1.2 Activités.....	page 3
1.3 Présentation de la filiale Group Security.....	page 4
1.4 Configuration du cadre de travail.....	page 4
2 — Développement du Framework.....	page 5
2.1 – Cahier des charges.....	page 5
2.1.1 Intitulé du projet.....	page 5
2.1.2 – Intérêts et contraintes du projet.....	page 6
2.1.3 Expressions des besoins.....	page 6
2.1.3.1 Besoins fonctionnels.....	page 6
2.1.3.2 Besoins non fonctionnels.....	page 7
2.2 – Rapport technique.....	page 8
2.2.1 Conception.....	page 8
2.2.1.1 Technologies utilisées.....	page 8
2.2.1.2 Arborescence des fichiers.....	page 8
2.2.1.3 Diagrammes UML.....	page 9
2.2.1.4 Description des algorithmes.....	page 13
2.2.2 Résultats.....	page 15
2.2.3 Perspectives.....	page 15
2.3 – Manuel d'utilisation.....	page 16
3 — Analyse de données.....	page 18
3.1 – Cahier des charges.....	page 18
3.1.1 Intitulé du projet.....	page 18
3.1.2 – Intérêts et contraintes du projet.....	page 18
3.1.3 Expressions des besoins.....	page 19
3.1.3.1 Besoins fonctionnels.....	page 19
3.1.3.2 Besoins non fonctionnels.....	page 19
3.2 – Rapport technique.....	page 19
3.2.1 Conception.....	page 19
3.2.1.1 Technologies utilisées.....	page 19
3.2.1.2 Arborescence des fichiers.....	page 20
3.2.1.3 Diagrammes UML.....	page 20
3.2.1.4 Description des algorithmes.....	page 22
3.2.2 Résultats.....	page 23
3.3 – Manuel d'utilisation.....	page 24
4 — Rapport d'activité.....	page 29
4.1 Méthodes de développement.....	page 29
4.2 Méthodes de travail.....	page 29
4.3 Planification.....	page 30
Conclusion.....	page 32

Table des figures

Figure 1 — Bête à cornes.....	page 5
Figure 2 - Diagramme pieuvre.....	page 7
Figure 3 - Arborescence framework.....	page 8
Figure 4 - Diagramme Use Case Framework.....	page 9
Figure 5 — Diagramme classe framework.....	page 10
Figure 6 — Diagramme classe Tools.....	page 10
Figure 7 — Diagramme classe Execution.....	page 11
Figure 8 — Diagramme classe MainView.....	page 11
Figure 9 — Diagramme classe Menu.....	page 12
Figure 10 - Diagramme séquence framework.....	page 13
Figure 11 — Capture d'écran du framework.....	page 16
Figure 12 — Capture d'écran du framework 2.....	page 17
Figure 13 - Arborescences executable JAR.....	page 20
Figure 14 — Diagramme Classe ReaderCSV.....	page 20
Figure 15 — Diagramme Classe WriterCSV.....	page 21
Figure 16 - Résultat scénario.....	page 24
Figure 17 - Grouping.....	page 25
Figure 18 - Sorting.....	page 25
Figure 19 - Journey.....	page 26
Figure 20 - Replacing.....	page 26
Figure 21 — Deduction.....	page 27
Figure 22 — Étape 6.....	page 27
Figure 23 — Inversion.....	page 28
Figure 24 — Planning prévisionnel.....	page 30
Figure 25 — Planning réel.....	page 31

Glossaire

framework : Ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel.

javax.swing : Swing fait partie de la bibliothèque Java Foundation Classes (JFC). C'est une API dont le but est similaire à celui de l'API AWT, mais dont les modes de fonctionnement et d'utilisation sont complètement différents.

java.io : Fournis des outils de gestion de fichiers via des flux de données, de la sérialisation et des fichiers système.

java.awt : AWT propose un ensemble de composants et de fonctionnalités pour créer des interfaces graphiques.

CSV : Fichier tableur, contenant des données sur chaque ligne séparée par un caractère de séparation (généralement une virgule, un point-virgule ou une tabulation).

Introduction

Pour l'obtention de mon DUT informatique à l'IUT de Montpellier-Sète je dois réaliser un stage en entreprise d'une durée de 12 semaines. Ce stage a trois objectifs : valider les compétences acquises lors de mes deux années de DUT, approfondir mes connaissances dans une mise en situation réelle sur un projet lié à ma formation, se confronter, dans notre spécialisation, au milieu du travail.

Je choisis de réaliser ce stage au sein de l'Université de Bristol pour étendre mes compétences linguistique et culturelle. C'est ainsi que du 8 Février 2016 au 29 avril 2016, je travaille auprès de mon tuteur de stage M.TREVOR Martin.

Je travaille sur un projet auquel M.TREVOR Martin fait partie, en collaboration avec British Telecom security. Les problèmes de sécurité dans l'enceinte d'une entreprise ou même au sein d'un réseau sont quelque chose de très important et ne doivent pas être pris à la légère. Dans le cas où les codes d'accès d'un employé auraient été découverts, il est essentiel de pouvoir le détecter en remarquant que le compte de la personne fait des actions inhabituelles. Par exemple envoyer beaucoup d'email ou même accéder à des bâtiments où elle n'était pas allée avant.

Le but principal de ce projet sera donc est de pouvoir prédire et détecter des comportements ou séquences d'événements anormaux. Mon objectif principal sera de développer un **framework*** permettant à l'utilisateur d'utiliser des outils Java pour traiter des jeu de données de différents formats, et de les enregistrer. Ce projet est réalisé dans le langage Java. Il viendra être complété par un scénario de test permettant de démontrer l'utilité du framework.

Je commencerai par présenter British Telecom, son organisation, ses activités. J'aborderai ensuite les éléments nécessaires à la compréhension du projet en détaillant le cahier des charges du framework, la mise en œuvre concrète de ce travail, ainsi que le manuel d'utilisation étape par étape. Ensuite le cahier des charges, le rapport technique et le manuel d'utilisation du scénario de test mis en place seront présentés. Enfin, dans la dernière partie je prendrais du recul sur le travail effectué durant le stage.

1 Présentation de l'entreprise

1.1 Création et évolution de l'entreprise

Le 1er octobre 1981, Post Office Telecommunications a été renommé British Telecom, et est devenu une entreprise publique indépendante de la Poste. En 1982, le monopole de BT dans les télécommunications a été supprimé, l'entreprise Mercury Communications ayant aussi obtenu une licence. Elle est donc l'opérateur historique britannique de télécommunications.

L'ouverture à la concurrence du marché date de 1982, mais BT est toujours leader dans le secteur de la téléphonie fixe. BT est présent dans plus de 170 pays, et près d'un tiers de ses revenus viennent de sa filiale Global Services. L'opérateur est célèbre en Grande-Bretagne pour ses cabines téléphoniques rouges (cabines publiques qu'il recycle actuellement en hotspots Wi-Fi), et pour avoir intenté un procès (abandonné) au fournisseur d'accès américain Prodigy.

Le groupe BT est organisé autour des filiales suivantes :

- BT Retail : responsable des lignes fixes aux particuliers
- BT Wholesale : gestion du réseau de télécommunications global et de la vente de minutes
- Openreach : partie externalisée de BT Wholesale chargée d'assurer l'accès équitable du réseau BT à ses concurrents
- BT Global Services : services aux entreprises et activités de conseil
- BT Exact / One IT : informatique interne et consultance, il y a parfois des recouvrements avec BT Global Services

- Group operations : sécurité des données et employés de l'entreprise ; recherche et développement, et toutes les fonctions support du groupe comme les ressources humaines, services juridiques, services généraux.
- BT Vision : le service de télévision ADSL.

1.2 Activités

BT gère les centraux téléphoniques, le réseau principal et la boucle locale de la plus grande partie des téléphones fixes du Royaume-Uni. Aujourd'hui, BT gère environ 25 millions de lignes fixes dans le royaume. À l'exception de la ville de Kingston-upon-Hull qui a son propre opérateur de télécommunications, Kingston Communications, BT a une obligation de service public sur tout le reste du Royaume. Ceci signifie donc que la société a l'obligation de fournir une ligne téléphonique fixe à n'importe quelle adresse en Angleterre. L'entreprise est aussi obligée de mettre à la disposition de tous des cabines téléphoniques.

Suite à sa privatisation en 1991, BT s'est retrouvé en position dominante sur certains marchés. Afin d'équilibrer cette position, l'entreprise doit satisfaire des obligations supplémentaires par rapport à ses concurrents dans ses secteurs, comme pratiquer des tarifs raisonnables et traiter sa clientèle de façon équitable.

Tout en continuant à fournir ses services dans ses secteurs traditionnels d'activité, secteurs que BT est donc obligé de fournir ou qui sont très réglementés, l'entreprise s'est diversifiée dans des produits et services plus rentables, et d'autres moins réglementés. On trouve dans cette catégorie la fourniture d'Internet à haut débit, ainsi que des solutions sur-mesure dans les télécommunications et les technologies de l'information pour les entreprises.

1.3 Présentation de la filiale Group Security

BT Group Security a pour but d'aider BT à protéger ses clients, réseau, propriété intellectuelle et autres atouts à travers le monde. Leurs priorités sont de maintenir un environnement de travail sécurisé pour tous les clients et employés de l'entreprise en réduisant les attaques envers les réseaux, gérer la sécurité des informations confidentielles et la détection et prévention des tentatives de fraudes.

BT Group Security est composée de deux petites équipes : Politique et communication et de deux plus grandes équipes : Protection des atouts et investigation. L'équipe d'investigation a pour but d'identifier et contrer toute action criminelle envers BT. L'équipe de protection des atouts fournit aux employés et clients de BT l'opportunité de signaler tout les incidents, une ligne de conseil et mise en place pour tous les besoins en sécurité.

1.4 Configuration du cadre de travail

Je travaille pour mon stage de fin de DUT en collaboration avec M.TREVOR Martin professeur en intelligence artificielle à l'université de Bristol. Il est mon tuteur et m'encadre tout au long de mon stage, en m'expliquant les attentes et en m'orientant dans ma conception.

Mon bureau se situe dans le service d'ingénierie mathématique de l'université de Bristol. Le bureau dans lequel je travaille comporte 4 postes informatiques, dont deux occupés. Les personnes occupant ces bureaux sont peu présentes, car elle travaille sur plusieurs projets. Un ordinateur de l'université m'a été attribué pour ce stage. Celui-ci étant connecté au réseau de la l'université, j'ai accès à tout le matériel dont je peux avoir besoin, comme les imprimantes ou l'accès à Internet.

2 — Développement du Framework

2.1 Cahier des charges

2.1.1 Intitulé

L'entreprise BT a besoin d'un framework capable d'exécuter des outils Java sous le format exécutable JAR. Dans le but de gérer et d'analyser des jeu de données. Ce framework va donc devoir être capable d'exécuter des fichiers JAR et d'enregistrer l'exécution. Il va également devoir fournir a l'utilisateur la possibilité d'ajouter pour chaque exécution des informations complémentaires. Par exemple le nombre de lignes qui vont être affecté dans les différents fichiers par l'exécutable, mais aussi il va devoir être capable de stocker un exécutable JAR d'inversion fourni lui aussi par l'utilisateur. Pour permettre de faire une action inverse dans le cas ou des résultats ne serait pas normal.

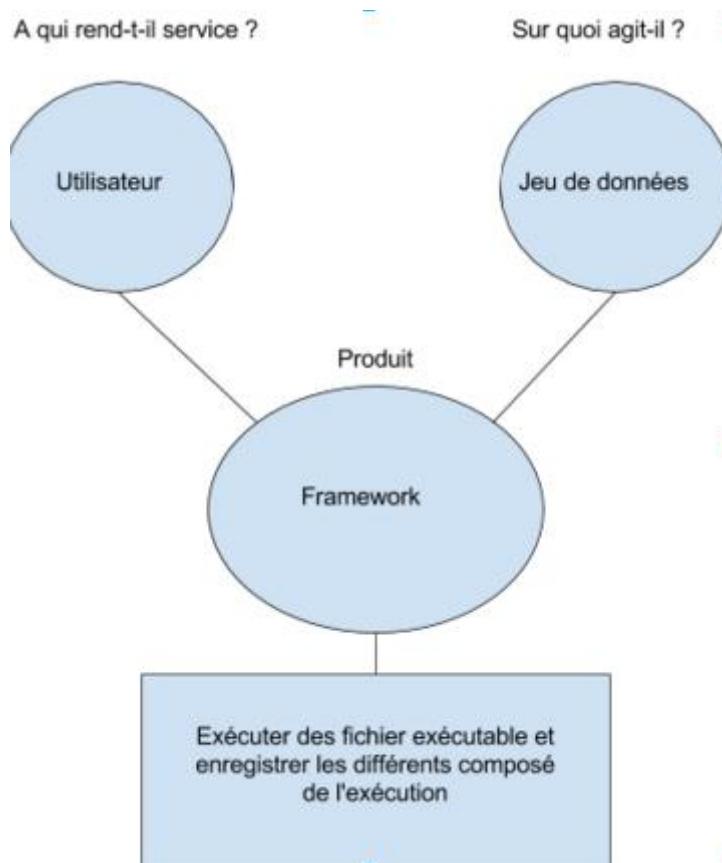


Figure 1 — Bête à cornes

2.1.2 – Intérêts et contraintes du projet

Ce projet apporte une aide au traitement des données. Ce qui va rentrer dans la continuité du projet principal qui a pour but de détecter des comportements anormaux.

Pour ce projet, il faudra donc :

- Utiliser des connaissances personnelles en programmation déjà acquises lors du cursus scolaire. Mais aussi prendre connaissance de notions de programmation plus avancées afin de pouvoir configurer judicieusement un programme pour une utilisation ultérieure.
- Apprendre à maîtriser la conception d'une interface graphique afin de pouvoir respecter les différentes fonctionnalités que l'on va mettre en place.

2.1.3 – Expression des besoins

2.1.3.1 – Besoins fonctionnels

Pour assurer une expérience optimale à l'utilisateur, il est nécessaire de s'assurer une programmation optimisée de chaque paramètre impliqué, dans leur définition comme leur exécution.

- Choix des fichiers d'exécution.
- Exécuter et rendre compte de l'état et du résultat de l'exécution.
- Interface de saisie de tout paramètre nécessaire pour l'exécution.
- Interface de saisie pour des informations facultatives supplémentaires pour une analyse ou bien une action inverse.
- Permettre à l'utilisateur de sauvegarder une session pour la reprendre a un autre moment.
- Possibilité de refaire une série d'exécution à l'aide d'un bouton.

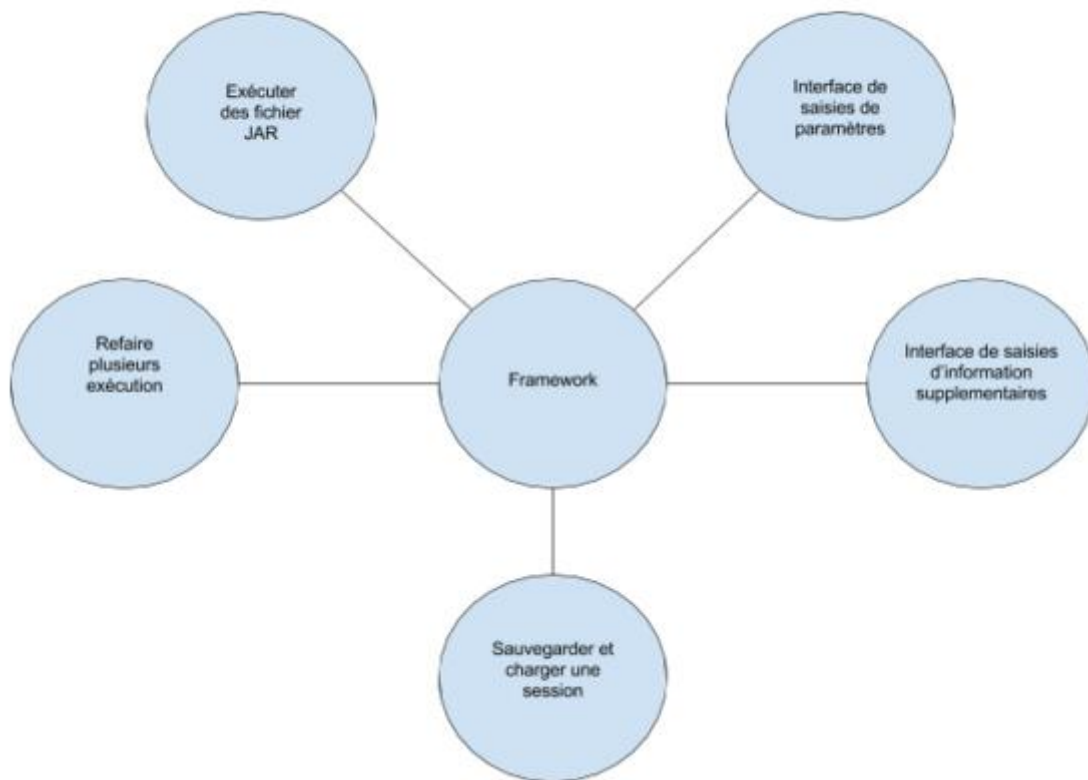


Figure 2 - Diagramme pieuvre

FC1 : Exécuter des exécutables JAR

FC2 : Interface de saisie de paramètre pour l'exécutable

FC3 : Interface de saisie d'information facultative

FC4 : Sauvegarder une session

FC5 : Refaire plusieurs exécutions

2.1.3.2 Besoin non fonctionnel

Le framework devra être réalisé en java. L'utilisation d'eclipse sera fortement appréciée. De plus, on aura recours à l'utilisation d'une librairie graphique.

2.2 Rapport technique

2.2.1 Conception

2.2.1.1 Technologies utilisées

Le Framework est entièrement programmé en utilisant le langage Java, car c'est une demande du cahier des charges. Pour l'aspect graphique le package **javax.swing*** est utilisé, pour gérer les entrées/sortie de fichier le package **java.io*** est utilisé, la gestion des évènements est faite grâce au package **java.awt**.*. Ces bibliothèques ont été choisies pour leurs simplicités d'utilisation et d'implémentation dans un programme.

L'entièreté du projet a été développée en utilisant Eclipse, car cet IDE possède une large panoplie d'outils utiles pour le développement en java comme la génération de javadoc, et l'exportation en exécutable JAR.

2.2.1.2 Arborescences des fichiers

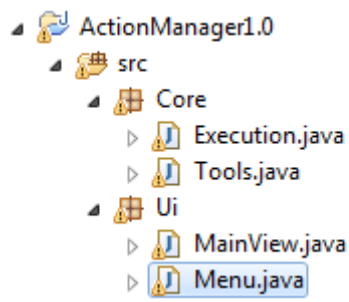


Figure 3 - Arborescence framework

Le découpage des classes est fait en deux package, le package UI qui contient tout l'aspect graphique du framework et le package Core qui contient les outils et les objets. Donnant l'arborescence ci-dessus.

2.2.1.3 Diagrammes UML

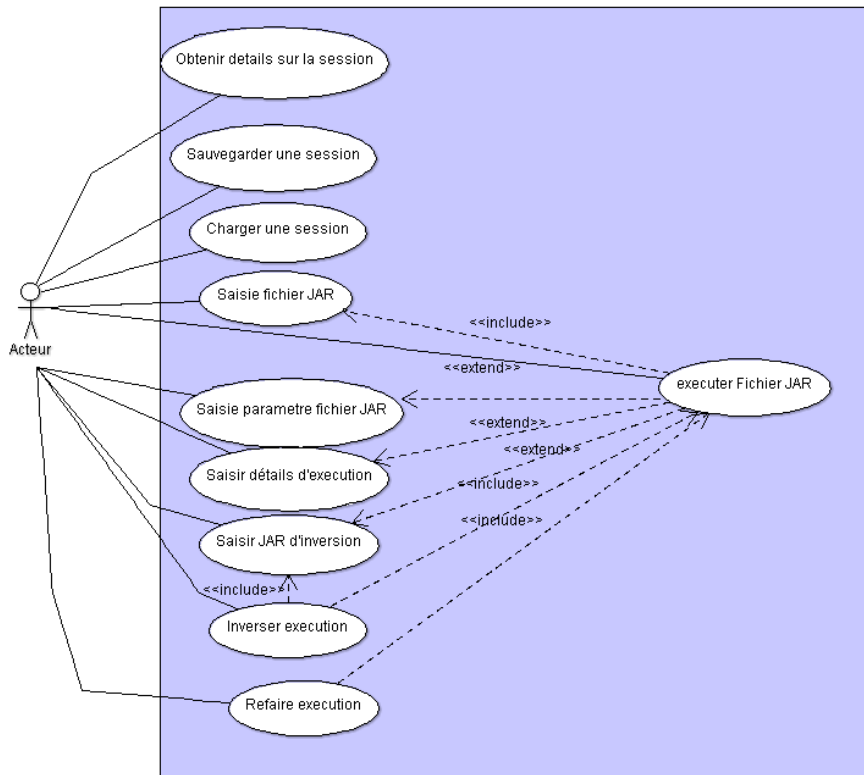


Figure 4 - Diagramme Use Case Framework

Le diagramme des cas d'utilisation nous montre l'ensemble des fonctionnalités que le framework fournit à l'utilisateur. Des fonctionnalités liées à la gestion des sessions, sauvegarder, charger et obtenir des détails sur la session courante. Il a aussi plusieurs options de saisies, saisir le fichier JAR qu'il souhaite exécuter, saisir les paramètres nécessaires à l'exécution, saisir des détails sur l'exécution et enfin saisir un fichier d'inversion. Tant qu'il a au moins saisi le fichier JAR à exécuter, il peut lancer l'exécution. S'il a déjà lancé des exécutions au préalable, il peut les refaire ou bien les inverser s'il a rentré un fichier JAR d'inversion.

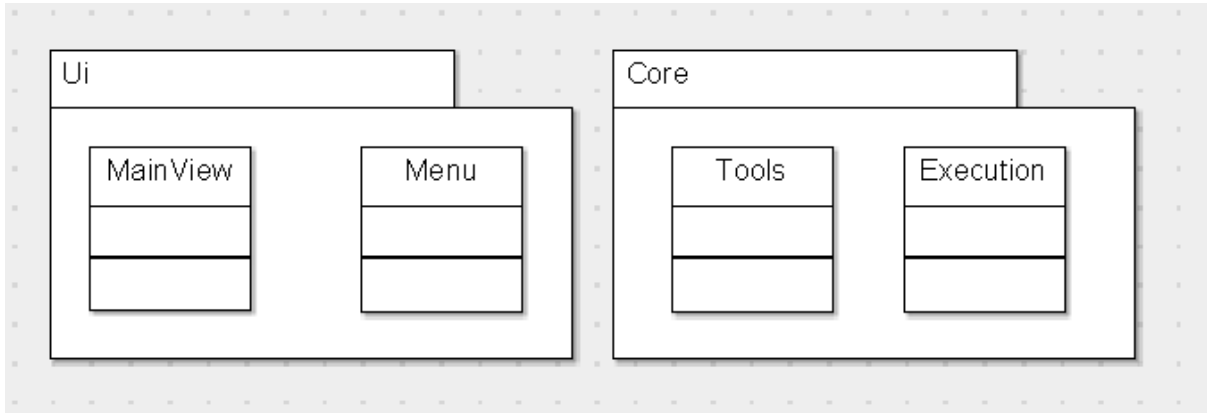


Figure 5 — Diagramme classe framework

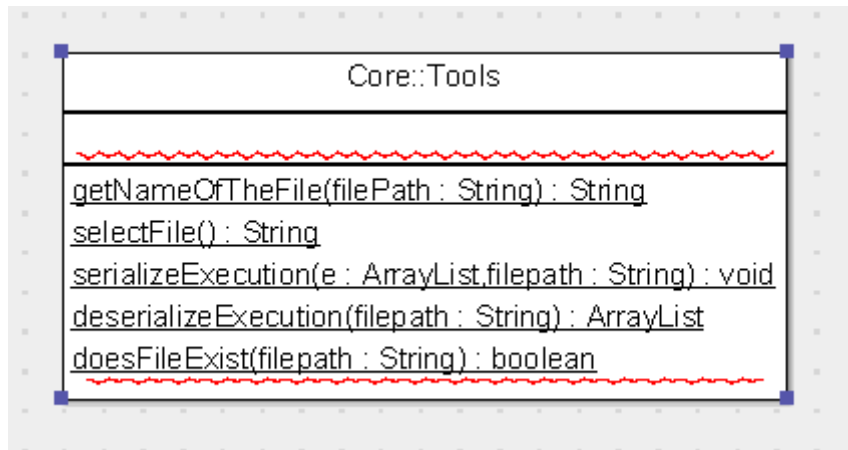


Figure 6 — Diagramme classe Tools

La classe « Tools » contient des méthodes statiques utiles accessibles dans tout le code. La méthode « `serializeExecution()` » pour la sauvegarde de session, « `deserializeExecution()` » le chargement de session, gérer les fichier via « `selectFile()` » pour permettre à l'utilisateur de sélectionner un fichier. Puis des méthodes utilitaires comme « `getNameOfTheFile()` » permet d'avoir le nom du fichier sélectionner via son chemin et « `doesFileExist()` » pour savoir si un fichier existe ou est accessible.

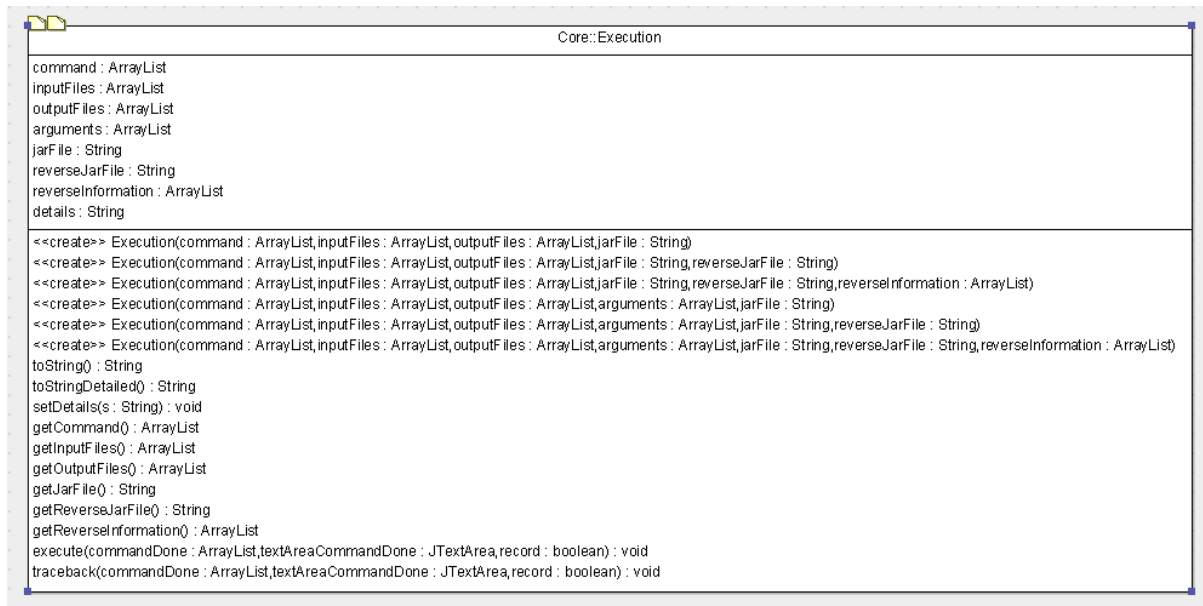


Figure 7 — Diagramme classe Execution

La classe `Execution` permet de définir une exécution. Elle est composée de la commande qui va servir à son exécution, des chemins de ses fichiers d'entrées, fichier de sortie, les différents paramètres, le chemin du fichier JAR qui va être exécuté, le chemin du fichier JAR d'inversion, et des informations sur l'exécution. Plusieurs constructeurs sont disponibles selon les entrées de l'utilisateur. Des méthodes classiques d'accesseur sont aussi disponibles telles que « `toString()` ». La méthode principale étant « `execute()` » qui va exécuter la commande comprise dans l'objet et va selon les paramètres l'afficher et la sauvegarder.

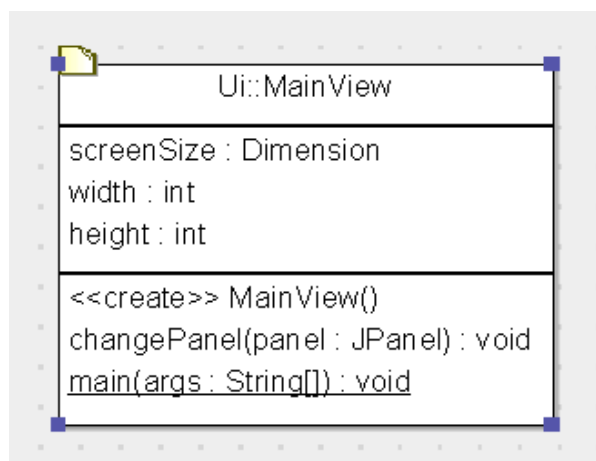


Figure 8 — Diagramme classe MainView

La classe `MainView` extends `JFrame`, c'est la fenêtre du programme, elle est définie par une taille en largeur et longueur, un constructeur qui se charge de la créer et de l'initialiser, une méthode « `changePanel()` » permettant de changer le contenu de la fenêtre. C'est aussi cette classe qui contient la méthode « `main` » du programme qui va être lancé lors du démarrage, elle va se charger de créer une nouvelle instance de `MainView`.

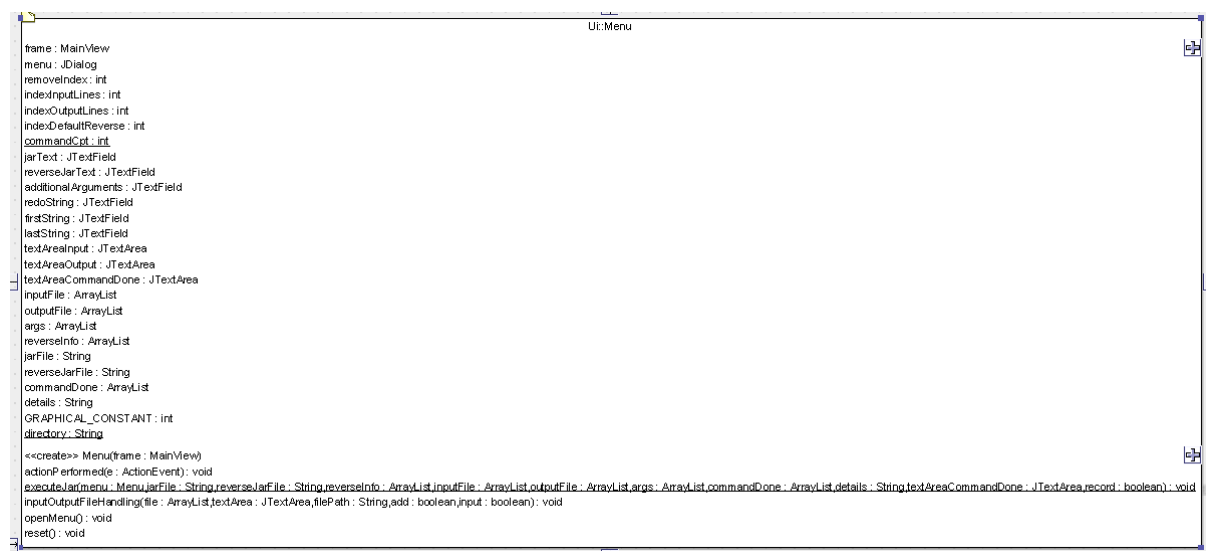


Figure 9 — Diagramme classe Menu

La classe `Menu` extends `JPanel` et implements `ActionListener`, elle se charge d'afficher le menu du framework, tous les boutons et labels. À sa création elle va initialiser tous les éléments nécessaires, c'est à dire leurs donner leurs positions, taille et `ActionListener`. La méthode « `actionPerformed()` » va être appelé à chaque fois qu'un bouton sera pressé, elle va se charger d'effectuer l'action définie par le bouton. La méthode « `executeJar()` » va être appelé lorsque l'utilisateur va appuyer que le bouton « exécuter jar », elle va se charger de créer et d'exécuter une exécution en fonction des paramètres saisis par l'utilisateur.

La méthode « `inputOutputFileHandling()` » permet de gérer la saisie de nouveaux fichiers d'entrée ou de sorties ou bien la suppression de fichier saisi précédemment. La méthode « `openMenu()` » va ouvrir une boîte de dialogue. La méthode « `reset()` » va se charger de réinitialiser les paramètres entrés par l'utilisateur.

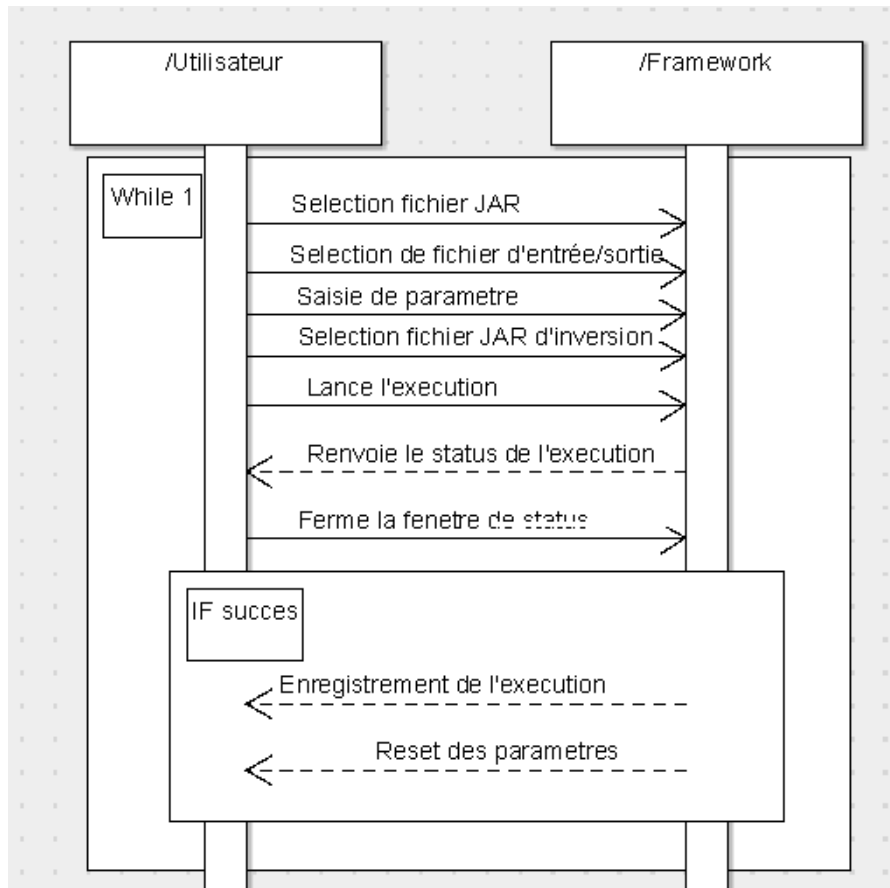


Figure 10 - Diagramme sequence framework

Dans ce cas, l'utilisateur sélectionne un fichier JAR, des fichiers d'entrée/sorties, des paramètres supplémentaires et un fichier JAR d'inversion. Ensuite, il lance l'exécution, le framework lui renvoie le statut de l'exécution. Une fois que l'utilisateur ferme la fenêtre le framework vérifie si l'exécution est un succès, si c'est un succès il va enregistrer l'exécution et réinitialiser tous les paramètres de la session.

2.2.1.4 Description des algorithmes

Étant donné que l'ensemble des algorithmes utilisé reste, des algorithmes peu complexes et que le fonctionnement du framework a été expliqué plus haut,

seulement la méthode « executeJar() » va être expliquée, car elle gère la fonctionnalité principale du framework. Le pseudo-code suivant résume le fonctionnement de la fonction dans sa globalité sans rentrer dans des détails inutiles.

```
Si ( fichierJAR != null)
    String command = "java -jar" + fichierJAR;
    Si ( fichierEntrée != null)
        command += fichierEntrée
    Fin si
    Si ( fichierSortie != null)
        command += fichierSortie
    Fin si
    Si ( argument != null)
        command += argument
    Fin si
    ProcessBuilder pb = new ProcessBuilder(command)
    pb.directory(fichierJAR.getDirectory())
    Process p = pb.start()
    Boolean process = true
    ouvrirFenetreDeStatut()
    TantQue ( process )
        Si (p.exitValue )
            fenetre.dispose()
            process = false
        Fin Si
    Fin TantQue
    Si ( record )
        Execution exec = new Execution()
        listeExecutionFaite.add(exec)
        afficherExecution(exec)
    Fin Si
    reset()
Fin Si
```

Le code va créer une commande à exécuter en fonction de toutes les entrées de l'utilisateur. Il va ensuite créer une instance de ProcessBuilder, chaque instance de ProcessBuilder va gérer un ensemble d'attributs de processus. La méthode « start() »

créer une nouvelle instance de Process avec ces attributs, cette instance permet d'obtenir le statut du processus en cours via la fonction « `exitValue()` ». Le booléen « `record` » est un paramètre qui va permettre au framework de savoir s'il faut sauvegarder cette exécution ou non dans l'historique et s'il faut par conséquent l'afficher.

2.2.2 Résultats

Ce framework a été conçu dans le but de permettre à l'utilisateur d'exécuter des fichiers exécutables JAR. Les fichiers JAR devront avoir un effet sur un jeu de données, ainsi ils devront avoir comme arguments un/des fichiers en entrées et un/des fichiers en sortie. L'exécution du fichier JAR se fera en utilisant la commande suivante : `Java — jar {Fichier d'entrée} {Fichier de sortie} {Arguments additionnels}`. Le fichier JAR devra donc suivre une certaine rigueur dans les arguments nécessaire a son exécution.

Toutes les exécutions ont la possibilité d'être détaillés par l'utilisateur en sélectionnant le nombre de lignes affecté par exemple : `1-1`, `1—` plusieurs... et/ou donné une description de l'exécution elle-même sous format de texte. L'utilisateur peut aussi donner un fichier JAR d'inversion qui sera enregistré et que l'utilisateur pourra exécuter s'il le souhaite.

Une fois une session d'exécution terminée l'utilisateur peut l'enregistrer et la recharger dans le cas il voudrait la reprendre plus tard. Il peut aussi obtenir un fichier texte avec le détail de toutes les exécutions effectuées durant la session pour pouvoir l'analyser.

2.2.3 Perspective

Le framework est fonctionnel, il reste néanmoins des imperfections et des points à améliorer.

Graphiquement, l'interface pourrait être plus attrayante et ergonomique. En effet, l'espace n'est pas très bien utilisé, l'utilisation de couleurs peut aussi être envisagée. Les textes utilisés pour décrire les exécutions pourraient être améliorés et plus versatiles pour permettre à l'utilisateur d'avoir de meilleur retour.

D'un point de vue technique, les fichiers de session utilise des chemins de fichier absolu, ce qui empêche de charger un fichier de session venant d'un autre

ordinateur, l'utilisation de chemin relatif serait plus adéquate. La possibilité de changer les arguments d'un fichier d'inversion n'est aussi pas implémentée et pourrait être appréciée pour certaines utilisations précises. Avoir une barre de progression pour lors de l'exécution d'un fichier JAR pourrait aussi permettre une meilleure expérience, pour cela il est possible d'imaginer une interface communiquant entre le processus et le framework pour lui donner des informations sur son état.

Ces axes sont des exemples de modifications du framework, qui pourrait permettre une meilleure expérience pour l'utilisateur.

2.3 Manuel d'utilisation

Le Framework propose plusieurs fonctionnalités qui vont être détaillées ci-dessous, en utilisant le code couleur de l'image pour séparer les différentes parties du programme.

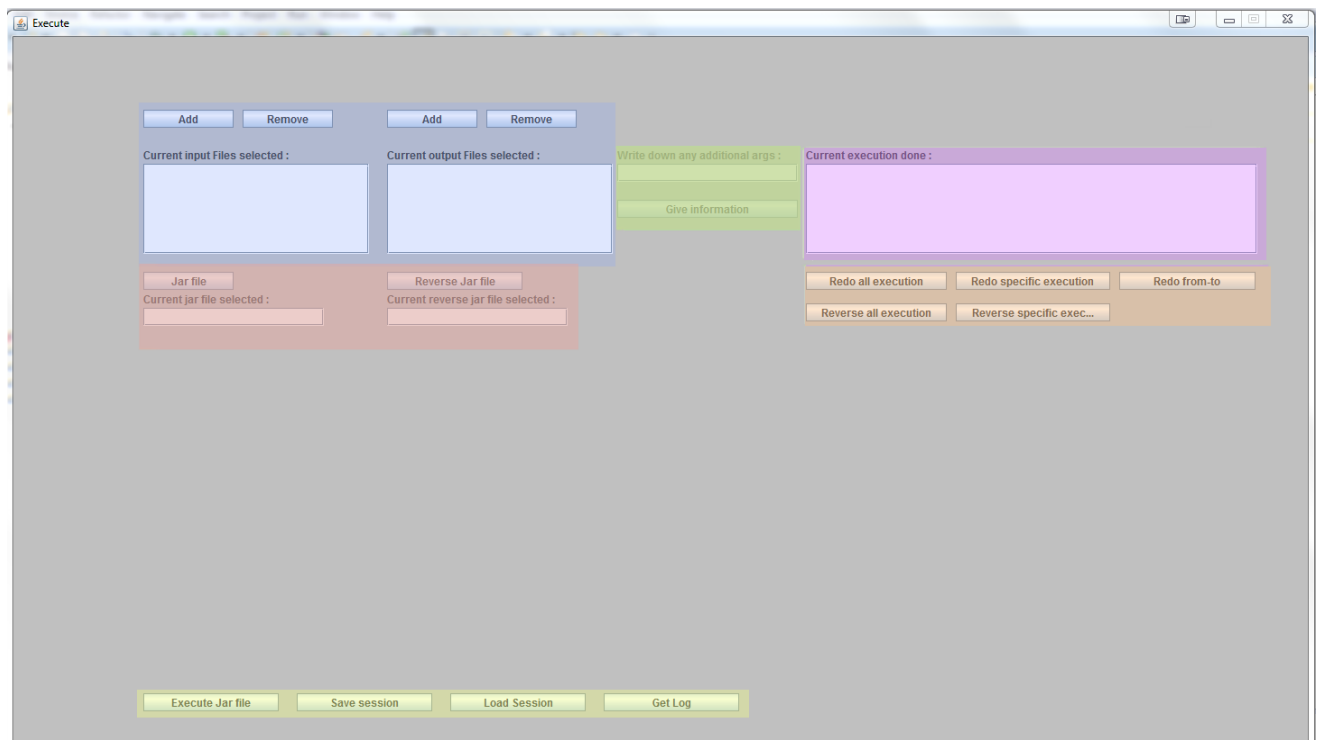


Figure 11 — Capture d'écran du framework

La zone en bleu : Permet à l'utilisateur d'ajouter des fichiers d'entrées et de sortis à son exécution via le bouton « ADD ». Il peut aussi enlever un fichier choisi grâce au bouton « REMOVE » tous les fichiers sélectionnés seront écrits dans la zone de texte correspondante.

La zone rouge : Permet à l'utilisateur de choisir le fichier JAR à exécuter, et facultativement choisir un fichier d'inversion.

La zone violette : Affiche le détail de toutes les exécutions qui ont été effectuées dans la zone de texte.

La zone verte : Permet à l'utilisateur d'écrire tout argument supplémentaire nécessaire à l'exécution de son fichier JAR, les arguments doivent être séparés par des espaces. Grâce au bouton « GIVE INFORMATION », une boîte de dialogue va s'ouvrir (ci-dessous), l'utilisateur pourra alors ajouter une description à son exécution, donner des détails sur le type d'opération effectuée ou bien spécifier un fichier d'inversion.

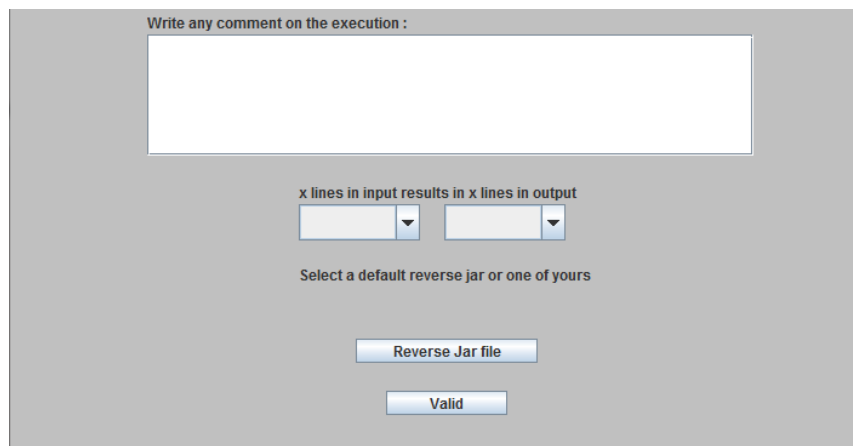
The screenshot shows a configuration window with a grey background. At the top, there is a text label "Write any comment on the execution :" followed by a large white text area. Below this, the text "x lines in input results in x lines in output" is displayed above two dropdown menus. Further down, the text "Select a default reverse jar or one of yours" is shown above a button labeled "Reverse Jar file". At the bottom of the window is a button labeled "Valid".

Figure 12 — Capture d'écran du framework 2

La zone orange : Permet à l'utilisateur de soit refaire une exécution précise via le bouton « REDO ». Refaire toutes les exécutions via le bouton « REDO ALL ». Refaire plusieurs exécutions grâce au bouton « REDO SPECIFIC EXECUTION » les

exécutions a refaire devront être séparées par des espaces. Ou bien refaire les exécutions depuis un certain point jusqu'à un autre avec le bouton « REDO FROM-TO ». Il peut aussi inverser des exécutions s'il a fourni un fichier d'inversion au préalable grâce au bouton « REVERSE ALL EXECUTION » et « REVERSE SPECIFIC EXECUTION ».

La zone jaune : Permet à l'utilisateur d'exécuter son fichier JAR avec les paramètres courants grâce au bouton « EXECUTE JAR FILE ». Sauvegarder la session courante grâce au bouton « SAVE SESSION », charger une session grâce au bouton « LOAD SESSION » et enfin obtenir un fichier texte détaillant les exécutions de la session courante avec « GET LOG ».

3 — Analyse de données

3.1 Cahier des charges

3.1.1 Intitulé

Afin de prouver que le framework est utilisable pour la gestion de données et la sécurité, un scénario de test est nécessaire, ce scénario devra utiliser le framework et un jeu de données.

3.1.2 – Intérêts et contraintes du projet

Ce projet permet de montrer l'utilité et le bon fonctionnement du framework sur des jeux de données. Ce projet va donc venir compléter le précédent et par conséquent le projet principal ayant pour but de détecter des comportements anormaux.

Pour ce projet, il faudra donc :

- Utiliser le framework développé en tenant donc compte de ses contraintes d'utilisation.
- Analyser un jeu de données et programmer des outils Java pour permettre d'obtenir des conclusions.

3.1.3 – Expression des besoins

3.1.3.1 – Besoins fonctionnels

Le scénario va devoir permettre de montrer que le framework fonctionne pour l'utilisation qui lui est destinée, pour ce faire il faudra donc :

- Analyser les données afin d'avoir une bonne vue d'ensemble et d'ainsi pouvoir faire la conception des outils qui vont être nécessaires.
- Prendre en compte les contraintes d'utilisation du framework comme l'ordre des paramètres pour pouvoir programmer des outils fonctionnels.
- Programmer les outils Java nécessaires capables de traiter les données et d'être utilisés avec le framework
- Montrer que les outils Java peuvent être utilisés avec le framework pour obtenir des conclusions.

3.1.3.2 Besoin non fonctionnel

Le framework devra être utilisé pour exécuter les outils développés, ils devront être développés en java et exporter en fichier JAR exécutable.

3.2 Rapport technique

3.2.1 Conception

3.2.1.1 Technologies utilisées

Le scénario de test est entièrement programmé en utilisant le langage Java, car c'est un prérequis pour l'utilisation du framework. Aucun aspect graphique n'a été intégré aux outils. Pour gérer les entrées/sortie de fichier le package `java.io.*` est utilisé pour sa simplicité d'utilisation et d'implémentation dans un programme.

Tous les outils ont été développés en utilisant Eclipse, car cet IDE possède une large panoplie d'outils utiles pour le développement en java comme la génération de javadoc, et l'exportation en exécutable JAR.

3.2.1.2 Arborescences des fichiers

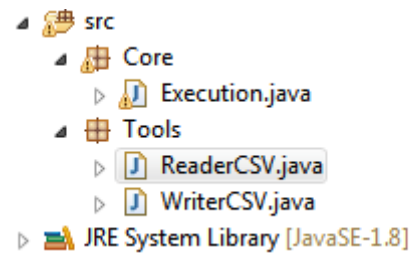


Figure 13 - Arborescences executable JAR

Le découpage des classes est similaire pour chaque outil et suit une logique, ils sont donc découpés en deux packages, le package Tools qui contient les outils permettant les actions de lecture et écriture de fichier **CSV***. Et le package Core qui contient une classe exécution effectuant les actions relatives a l'outil.

3.2.1.3 Diagrammes UML

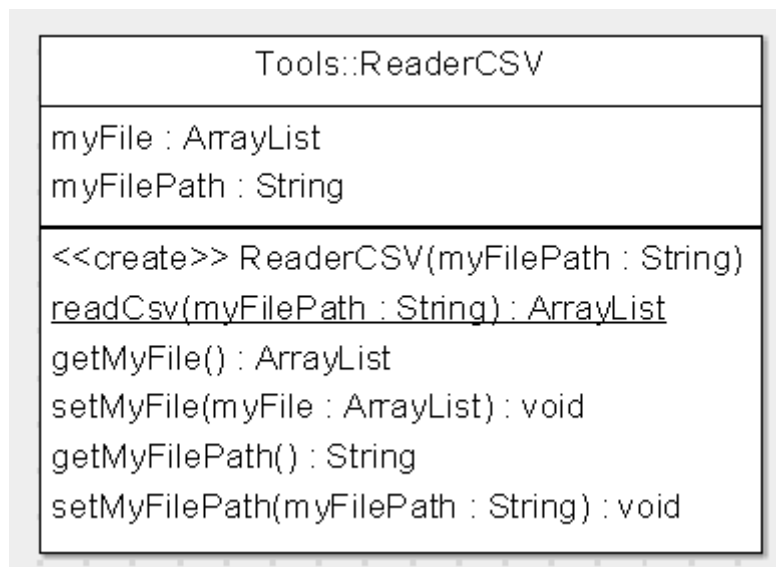


Figure 14 — Diagramme Classe ReaderCSV

La première classe commune a tous les outils java est `readerCSV` permettant de lire un fichier CSV. Elle est composée du chemin du fichier à lire et d'une liste d'éléments représentant les valeurs. Elle possède un constructeur, ainsi que des accesseurs classiques. La méthode la plus importante étant « `readCsv()` ». C'est une méthode statique prenant en argument le chemin du fichier et renvoyant une `ArrayList<String[]>` où chaque élément de l'`ArrayList` représente une ligne du fichier et chaque élément des `String[]` représente un élément de la ligne courante.

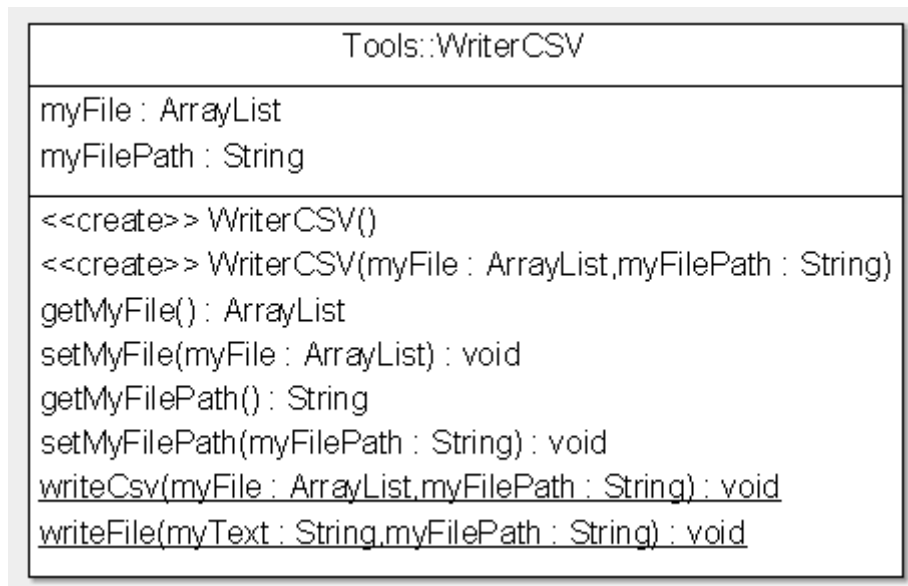


Figure 15 — Diagramme Classe `WriterCSV`

La seconde classe commune a tous les outils java est `writerCsv` permettant d'écrire des fichiers CSV. Elle est composée du chemin où le fichier sera écrit et d'une liste d'éléments à écrire. Elle possède des constructeurs, ainsi que des accesseurs classiques. Les méthodes les plus importantes sont « `writeFile()` », et « `writeCsv()` », elles sont statiques. La première prend en argument une `ArrayList` d'éléments à écrire et un chemin de fichier où l'écrire. La seconde prend en argument un texte à écrire et un chemin de fichier où l'écrire.

3.2.1.4 Description des algorithmes

Chaque outil possédant un fonctionnement différent, seules les méthodes d'écriture et de lecture de fichier vont être détaillées, les algorithmes principaux des outils restant plutôt simples.

```
function readCsv(String Myfilepath)
    FileInputStream ips = new FileInputStream(myFilePath);
    InputStreamReader ipsr = new InputStreamReader(ips);
    BufferedReader br = new BufferedReader(ipsr);
    ArrayList<String[]> myTab = new ArrayList<String[]>();
    String ligne ;
    Tant que ( (ligne = br.readLine()) != null)
        myTab.add(ligne.split(«, »))
    Fin tant que
    br.close()
    return myTab
end Function
```

La fonction « readCsv() » utilise les outils fournis par la bibliothèque java.io.*, en utilisant la méthode « readLine() », on obtient une chaîne de caractères si une nouvelle ligne est disponible ou un indicateur null en cas de fin de fichier. La fonction lit donc chaque ligne grâce à la boucle « tant que » à chaque ligne la fonction sépare la chaîne de caractère avec la méthode « split() » pour obtenir chaque élément distinct de la ligne et l'ajoute au tableau qui va être renvoyé.

```
function writeFile(String mytext,, String myFilePath)
    File file = new File(myFilePath+".java")
    Si (!file.exists())
        file.createNewFile() ;
    Fin si
    FileWriter fw = new FileWriter(file)
    fw.write(myText)
    fw.flush()
    fw.close()
end Function
```

La fonction « writeFile() » utilise les outils fournis par la bibliothèque java.io.*, en utilisant la méthode « exist() » la fonction peut vérifier si le fichier a bien été créé, si ce n'est pas le cas, elle va créer le fichier avec la méthode « createNewFile() ». La méthode write écrit tout simplement une chaîne de caractères dans un fichier, on l'utilise donc pour écrire la chaîne de caractère en paramètre.

```
function writeCsv(ArrayList<String[]> myFile, String myFilePath)
    BufferedWriter fichier =
        new BufferedWriter( new FileWriter(myFilePath))
    For ( int i = 0 ; i < myFile.size() ; i++ )
        For( int x = 0 ; x < myFile.get(i).length ; x++)
            fichier.write(myFile.get(i)[x])
            Si ( x != myFile.get(i).length - 1)
                fichier.write(«, »)
            Fin Si
        Fin For
        fichier.newLine()
    Fin For
    fichier.close()
end Function
```

La fonction « writeCsv() » utilise les mêmes outils que la fonction « writeFile() ». Elle va effectuer une double boucle For, la première représente la ligne dans le fichier et la seconde représente l'élément de la ligne, chaque élément dans le fichier est séparé par une virgule.

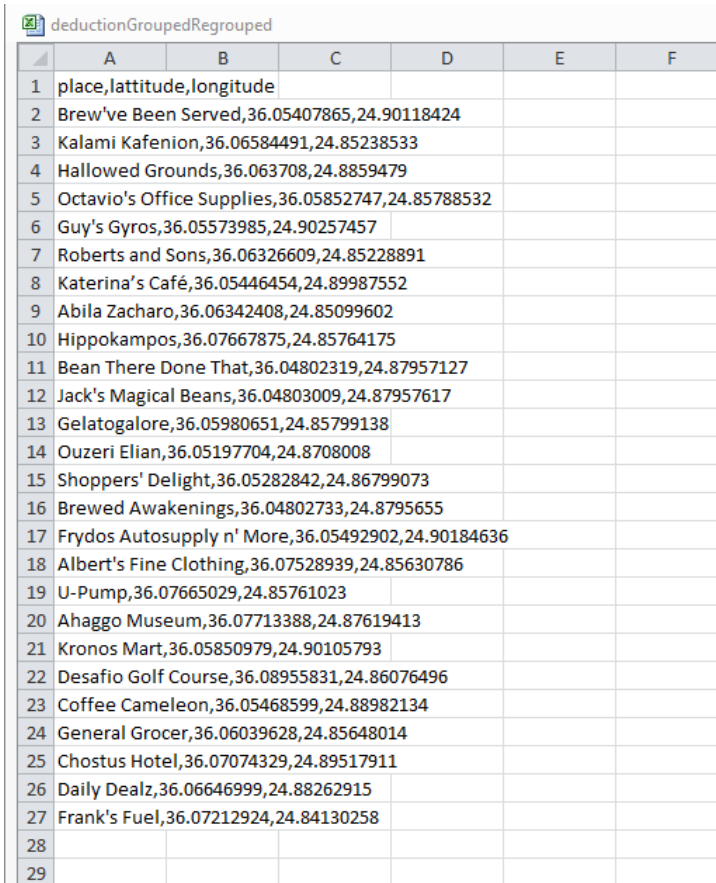
3.2.2 Résultats

Le scénario de test mis au point permet avec les données fournies d'obtenir après une suite d'opération une déduction sur l'emplacement de chaque commerce. Il permet aussi de retourner en arrière et tracer les données pour savoir quelles informations on peut arriver à cette déduction.

Chaque outil java peut et doit être utilisé avec le framework.

3.3 Manuel d'utilisation

Le détail de tous les fichiers jar, leurs utilités et conditions d'utilisation, est fait dans les annexes (voir pages : II - III). En utilisant ces fichiers Jar dans le bon ordre, on peut obtenir un fichier contenant les locations approximatives des commerces, le fichier se présente sous cette forme :



	A	B	C	D	E	F
1	place,lattitude,longitude					
2	Brew've Been Served,36.05407865,24.90118424					
3	Kalami Kafenion,36.06584491,24.85238533					
4	Hallowed Grounds,36.063708,24.8859479					
5	Octavio's Office Supplies,36.05852747,24.85788532					
6	Guy's Gyros,36.05573985,24.90257457					
7	Roberts and Sons,36.06326609,24.85228891					
8	Katerina's Café,36.05446454,24.89987552					
9	Abila Zacharo,36.06342408,24.85099602					
10	Hippokamos,36.07667875,24.85764175					
11	Bean There Done That,36.04802319,24.87957127					
12	Jack's Magical Beans,36.04803009,24.87957617					
13	Gelatogalore,36.05980651,24.85799138					
14	Ouzeri Elia,36.05197704,24.8708008					
15	Shoppers' Delight,36.05282842,24.86799073					
16	Brewed Awakenings,36.04802733,24.8795655					
17	Frydos Autosupply n' More,36.05492902,24.90184636					
18	Albert's Fine Clothing,36.07528939,24.85630786					
19	U-Pump,36.07665029,24.85761023					
20	Ahaggo Museum,36.07713388,24.87619413					
21	Kronos Mart,36.05850979,24.90105793					
22	Desafio Golf Course,36.08955831,24.86076496					
23	Coffee Cameleon,36.05468599,24.88982134					
24	General Grocer,36.06039628,24.85648014					
25	Chostus Hotel,36.07074329,24.89517911					
26	Daily Dealz,36.06646999,24.88262915					
27	Frank's Fuel,36.07212924,24.84130258					
28						
29						

Figure 16 - Résultat scénario

Détaillons chaque étape :

- Regrouper les données GPS par id de voiture :

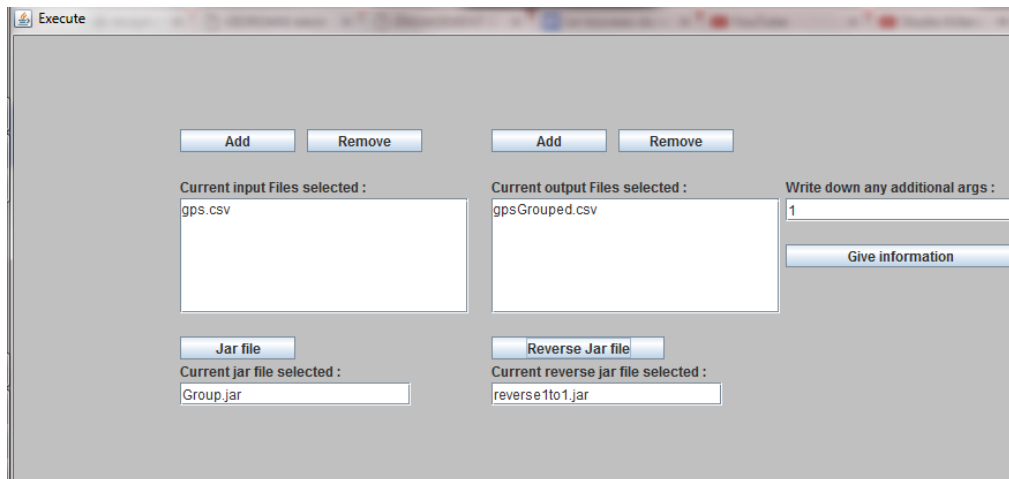


Figure 17 — Grouping

- Trier chronologiquement le fichier gpsGrouped.csv :

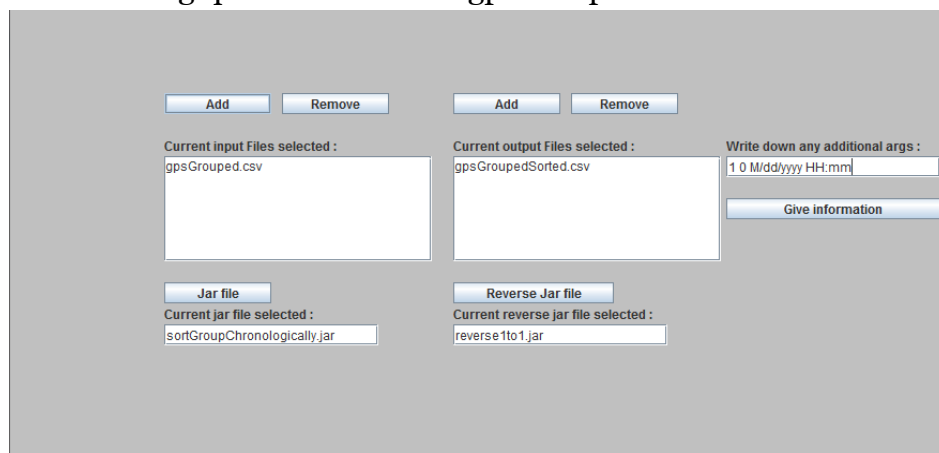


Figure 18 — Sorting

- Obtenir les voyages espacer de 10 minutes du fichier gpsGroupedSorted.csv :

The screenshot shows the 'Journey' application interface. It features two columns for file selection. The left column, titled 'Current input Files selected:', contains a text box with 'gpsGroupedSorted.csv' and a 'Jar file' section with 'getJourney.jar'. The right column, titled 'Current output Files selected:', contains a text box with 'journey10minutes.csv' and a 'Reverse Jar file' section with 'reverseWithIndication.jar'. Above each column are 'Add' and 'Remove' buttons. To the right of the output column is a 'Write down any additional args:' section with a text box containing '600000' and a 'Give information' button.

Figure 19 — Journey

- Il faut maintenant traiter le fichier cc_data.csv, il faut commencer par changer le nom des gens par l'id de la voiture qui leurs est attribué :

The screenshot shows the 'Replacing' application interface. It features two columns for file selection. The left column, titled 'Current input Files selected:', contains a text box with 'cc_data.csv' and 'car-assignments.csv', and a 'Jar file' section with 'replaceNameById.jar'. The right column, titled 'Current output Files selected:', contains a text box with 'ccDataWithId.csv' and a 'Reverse Jar file' section with 'reverseWithIndication.jar'. Above each column are 'Add' and 'Remove' buttons. To the right of the output column is a 'Write down any additional args:' section with an empty text box and a 'Give information' button.

Figure 20 — Replacing

- Il nous faut maintenant regrouper et trier le fichier résultant de la même manière que pour les données GPS.
- On peut maintenant déduire les locations des commerces en utilisant le fichier de trajet et de carte de crédit :

Current input Files selected :
ccDataWithIdGroupedSorted.csv
journey10minutes.csv

Current output Files selected :
deduction.csv

Write down any additional args :
3 0 0 1 2 5 6 M/d/yyyy HH:mm MM/dd

Give information

Jar file
Current jar file selected :
deducePlacesLocation.jar

Reverse Jar file
Current reverse jar file selected :
reverseWithIndication.jar

Figure 21 — Deduction

arguments : 3 0 0 1 2 5 6 M/d/yyyy HH:mm MM/dd/yyyy HH:mm:ss
C:\\Users\\Rifhice\\Documents\\Rapport de stage\\VAST 2014\\CSV
files\\Output\\Output2\\deduction.csv

- Il faut regrouper les déductions par commerces.
- Enfin il nous faut regrouper chaque déduction pour avoir une location définitive :

Current input Files selected :
deductionGrouped.csv

Current output Files selected :
deductionGroupedRegrouped.csv

Write down any additional args :

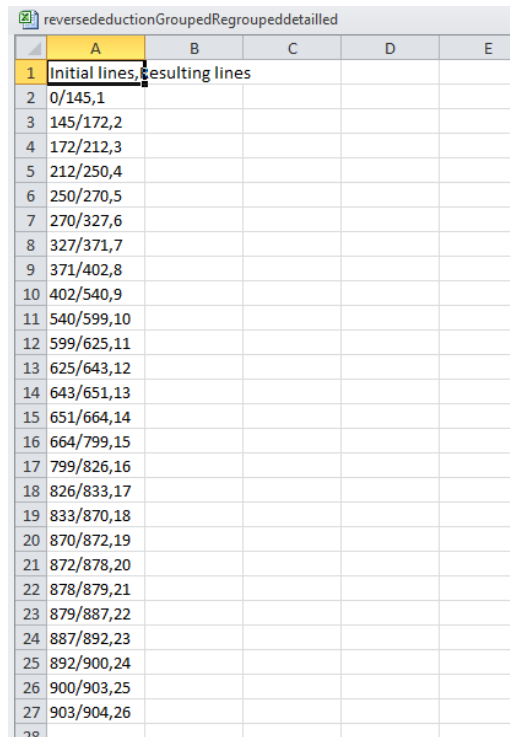
Give information

Jar file
Current jar file selected :
regroupDeduction.jar

Reverse Jar file
Current reverse jar file selected :
reverseWithIndication.jar

Figure 22 — Étape 6

D'un autre côté, on peut utiliser les fichiers d'inversion pour retracer les informations obtenues. Il faut simplement cliquer sur le bouton « reverse specific execution » et choisir l'étape que l'on veut retracer, les fichiers obtenus seront de cette forme :



	A	B	C	D	E
1	Initial lines,	Resulting lines			
2	0/145,1				
3	145/172,2				
4	172/212,3				
5	212/250,4				
6	250/270,5				
7	270/327,6				
8	327/371,7				
9	371/402,8				
10	402/540,9				
11	540/599,10				
12	599/625,11				
13	625/643,12				
14	643/651,13				
15	651/664,14				
16	664/799,15				
17	799/826,16				
18	826/833,17				
19	833/870,18				
20	870/872,19				
21	872/878,20				
22	878/879,21				
23	879/887,22				
24	887/892,23				
25	892/900,24				
26	900/903,25				
27	903/904,26				
28					

Figure 23 — Inversion

4 Rapport d'activité

4.1 Méthodes de développement

Le tuteur n'avait pas un projet bien défini sur les objectifs à faire, nous avons donc décidé de prendre une méthode de développement itérative, car cela semblait plus approprié dans le cadre de ce projet. Nous avons donc mis en place des réunions hebdomadaires pour prendre conscience des avancées et donner des directives et des objectifs pour la semaine d'après.

4.2 Méthodes de travail

Le travail de la semaine était découpé en plusieurs parties :

- Analyser les notes prises lors de la réunion avec le tuteur le vendredi d'avant, donner de la forme et du sens à ces notes pour ensuite avoir une meilleure vue sur les attentes et comment les atteindre. Cela était écrit et enregistré dans un google drive pour y avoir un accès à n'importe quel moment.
- Trouver des solutions aux objectifs et les programmer de manière optimale.
- Je tenais un rapport de la semaine et écrivais dedans tout ce qui avait été accompli au cours de la semaine (voir annexes : page IV – VI). Ce rapport était exporté et ensuite envoyé à mon tuteur pour la veille de la réunion pour qu'il ait une vue sur mon travail.
- La communication entre mon tuteur et moi se faisait principalement via E-mail.
- Pour le transfert des fichiers JAR et du framework, j'ai utilisé un service fourni par l'université de bristol permettant de transférer des fichiers volumineux.

4.3 Planification

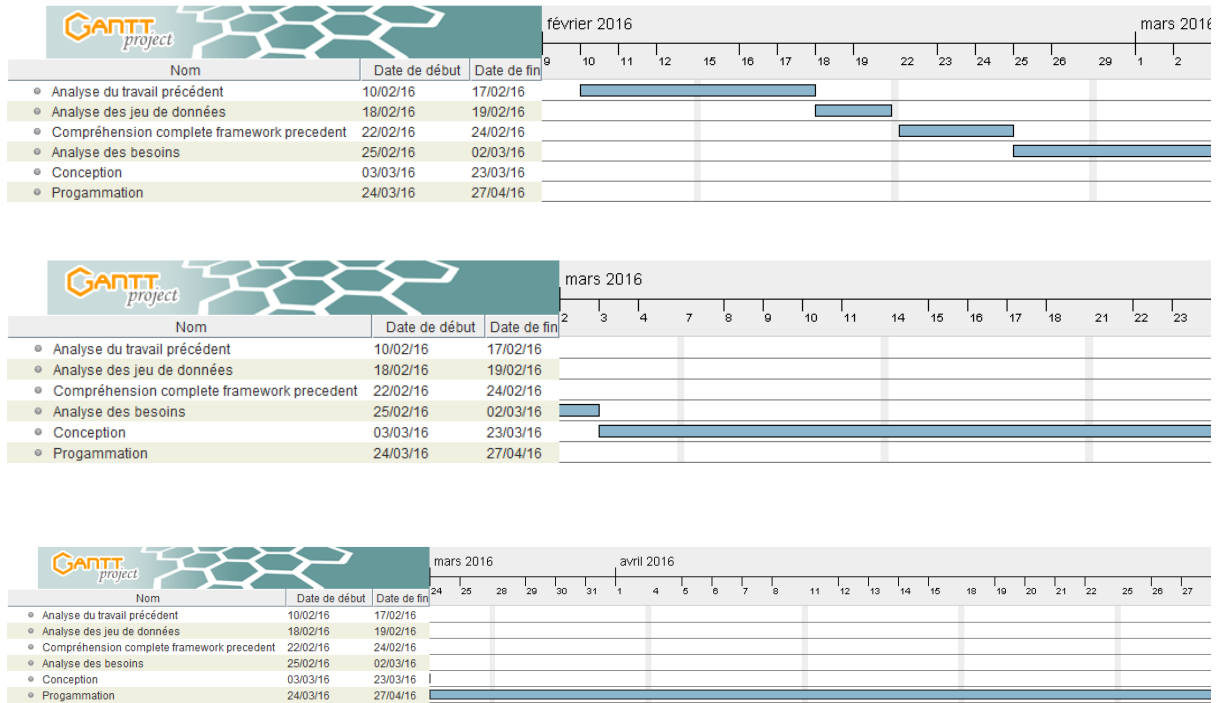


Figure 24 — Planning prévisionnel

À la première impression, je pensais que mon tuteur possédait un cahier des charges et des besoins précis, le diagramme était donc relativement simple avec des phases d'analyse, conception et programmation séparée pour avoir un résultat à la fin du stage. Or il c'est avéré qu'il n'avait pas d'idées précises, il a donc fallu passer sur un modelé itératif.

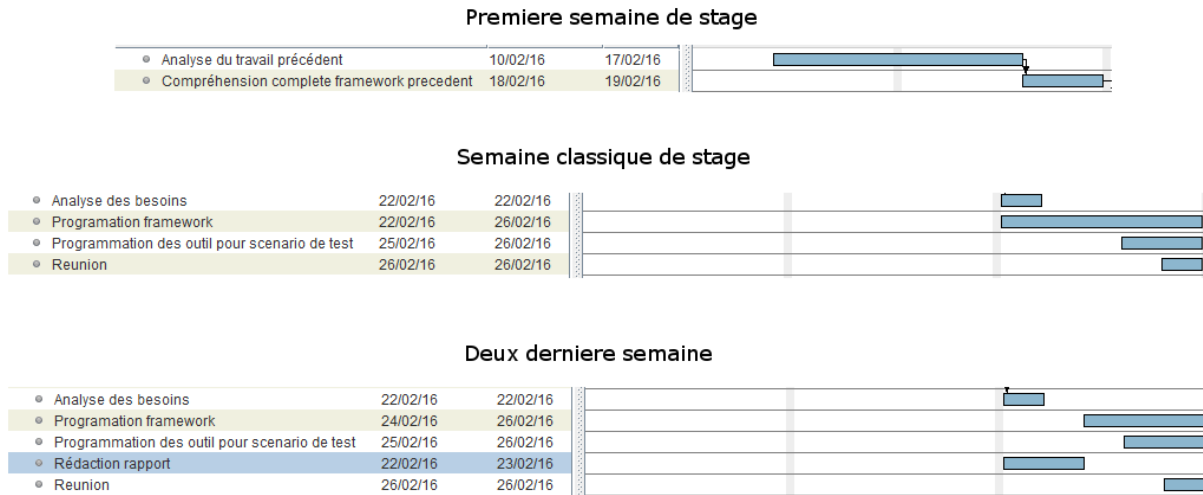


Figure 25 — Planning réel

On peut remarquer que dans le diagramme initial il me fallait analyser les jeu données, mais j'ai vite compris que le framework n'agirait pas sur les jeu de données directement. Je n'ai donc pas passé de temps à analyser les jeu de données cela ma donc laissé plus de temps pour prendre en compte les objectifs et le contexte du projet afin de pouvoir commencer l'analyse des besoins. Mais comme il me fallait commencer à programmer pour pouvoir comprendre quelles étaient les possibilités, j'ai alors commencé la programmation et l'analyse, je réalisais en parallèle une autre mission qu'il m'avait donnée. J'ai commencé la rédaction du rapport en même temps que la programmation durant les deux dernières semaines.

Conclusion

Les objectifs de ce stage étaient de développer un framework permettant de gérer des exécutions de fichier JAR ainsi que de fournir un scénario de test d'utilisation du framework sur un jeu de données.

Les résultats obtenus au regard des attentes initiales ne sont pas si différents. Le cahier des charges du framework a été respecté. La programmation du framework amène tout de même à penser qu'il reste des perspectives d'évolution comme l'amélioration de l'interface graphique ou encore un lien direct à une base de données plutôt que l'utilisation de fichiers. Le scénario de test a été mené à bien et montre bien l'utilité du framework.

Grâce à la méthode de développement itérative, ce projet m'a tout de même permis d'acquérir plus d'expérience en termes de méthodes de travail. Notamment dans la communication dans une entreprise ainsi qu'avec un client, pour amener un projet à bout. L'indépendance qu'il m'a tout de même été donné sur ce projet m'a permis de développer des méthodes de travail individuelles, notamment la répartition de mon temps et du travail au cours de la semaine.

J'ai pu acquérir des connaissances techniques surtout pour la réalisation de l'interface graphique avec javax.swing. J'ai aussi pu voir comment l'analyse de données peut être utilisée pour la sécurité.

Je conclurais sur le fait que le projet a été intéressant non seulement sur la manière dont j'ai pu prendre en main le sujet. Mais aussi sur l'aspect de la programmation d'un outil de gestion avec une interface graphique et de l'analyse de données.

Bibliographie / Sitographie

Adresse de la page	Date de consultation	Type de site	Information recherché
http://www.btplc.com/Innovation/Innovation/Cybercrime/	14/04/2016	Site d'entreprise	Information sur l'entreprise BT
https://www.newscientist.com/article/dn21989-ai-system-helps-spot-signs-of-copper-cable-theft/	14/04/2016	Journal	Information sur l'entreprise BT
http://www.btplc.com/Thegroup/	14/04/2016	Site d'entreprise	Information sur l'entreprise BT
https://fr.wikipedia.org/	14/04/2016	Wiki	Information sur l'entreprise BT
http://home.bt.com/	14/04/2016	Site d'entreprise	Information sur l'entreprise BT

Annexes

Our objective is to get an approximate location of the shops using this data. To do so, we will be programming JAR file that will process the data in order for us to get results, to execute the JAR file in an optimise way we will be using the framework.

The different JAR file that will be used :

- **group.jar** : This jar file will group the lines according to their common argument in a given column and write it in a new file.
 - @param 0 Input file location.
 - @param 1 Output file location.
 - @param 2 Index of the Id column in the input file.
- **sortGroupChronologically.jar** : This jar file will sort a each group from a grouped file chronologically.
 - @param 0 Input file location.
 - @param 1 Output file location.
 - @param 2 Index of the Id column in the input file.
 - @param 3 Index of the time column in the input file.
 - @param 4 Format of the time.
- **getJourney.jar** : This jar file will find every journey for each individual in the file, a journey is defined by a starting point and time, ending point and time and every journey in separated by a defined amount of time given by the user.
 - @param 0 Input file location.
 - @param 1 Output file location.
 - @param 2 Amount of time between 2 journey.
 - @all subsequent param the program will only get the journey from those id.
- **getDistinctValue.jar** : This jar file will find every distinct value of a given column and write them into a new file
 - @param 0 Input file location.
 - @param 1 Output file location.
 - @param 2 Index of the column.
- **deducePlacesLocation.jar** : This jar file will deduces the approximative location of places with the given data.
 - @param 0 Input file location. Credit card info grouped and sorted chronologically.
 - @param 1 Input file location. Journey information grouped and sorted chronologically.
 - @param 2 Index of the Id column in the credit card file

@param 3 Index of the Id column in the journey file
 @param 4 Index of the time column in the credit card file
 @param 5 Index of the time journey start column in the journey file
 @param 6 Index of the time journey end column in the journey file
 @param 7 Index of the latitude end of the journey column in the journey file
 @param 8 Index of the longitude end of the journey end column in the journey file
 @param 9 Format of the time in the credit card file
 @param 10 Format of the time in the journey file
 @param 11 Output file location.

- **regroupDeduction.jar** : This jar file will regroup the deduction find with the deducePlacesLocation.jar file and write them into a new file. To do so it'll read the input file, and get the average of all the location find for a place. Note : the deduction file need to be grouped by place before executing this file.
 @param 0 Input file location.
 @param 1 Output file location.
- **replaceNameById.jar** : This jar file will replace all the names of the people in the credit card transaction file by their id, it will write this in a new file.
 @param 0 Input file location. Credit card transaction.
 @param 1 Input file location. Assignement of the car to the people.
 @param 2 Output file location.

The reverseJAR file that will be used :

- **reverse1to1.jar** : This jar file tell the user how a line resulted in an other file after any 1 to 1 computation.
 @param 0 Input file location.
 @param 1 Output file location.
- **reverseWithIndication.jar** : This jar file tell the user how a line resulted in an other file after any computation if an indication on the line is given.
 @param 0 Input file location.
 @param 1 Output file location.

Weekly report #2

Early week report :

To achieve the objectives, I figured that I had to either take the examples from the given software or to make my own operations. I decided to make my own operation using in part the code from the software given. For now I've done 1 jar file, allowing to merge two CSV files into one output file. The input files and output file name are entered as so: input1 input2 outputName.

I now need to create maybe 2 or 3 other jar files doing other simple operation. I'll do so in parallel as the main program. The main program for now just has a basic UI, which isn't doing anything, the functionality will be develop in the future. However, we can already say that it we'll have to be able to execute jar files, allowing the user to enter all the arguments he could possibly need to enter. Then give a report of the execution. It'll record all operations done to allow the users to redo them. Eventually, it should be able to reverse the operations given by the user.

End of the week report :

The program is currently working, it allows to select as many input files as wanted, as many output files (Needs to be enhanced, the UI isn't user-friendly). Choose one jar file and finally execute it.

It records all the execution done, print them on a text area, several options allow the user to redo the execution following different pattern: Redo all executions at once, Redo only specific operations, Redo execution from a starting point to an ending point. Those functionalities haven't been tested yet since I don't have enough jar files to try it out. However the code is written and I believe it should work fine.

Next week objectives report :

Need to do some more Jar files to test the whole program entirely. Then I need to run some test on the functionality. And then hopefully I will probably be able to work onto the traceback functionality.

Weekly report #3

During the first days of the week I have designed and created 3 Jar file, a Sort.jar allowing to regroup every similar string from a specific column next to them, this is useful to regroup all the action from one id in the file for example. The second file is GetMovementInfo.jar, it allow me from a sorted file to get the information of their movement, their starting point and time, ending point and time and the duration of the movement. The final jar file is a reverse jar file Reverseito1.jar it allows me by giving him the input files and output files from an execution to get a new file that contains, for each line of the output file the corresponding line in the input file.

Then I improved the ui a little, there is still a lot to do to make it look good but it's a first step. The redo functionality has been tested, corrected and are now working. The user can now give an additional reverse jar file, he can also give information about how the jar file work (1-1, 1-many...) there is not a lot of choice: 1, Miami, all. But the nuances will be added later. I'm currently working on a way to able the user to save a session and reload it another time. The traceback buttons have been added, but are not implemented yet.

For the next week, an acceptable objectives might be to finish the save/reload session. and to partially implement the traceback function.

Weekly report #5

First I've worked on creating two jar files that will respectively allow me to group the data following an argument, i.e. an id and then to Chronologically sort the data for each group. Once this done I've used it onto the gps.csv file, this allowed me to get the GPs data organized for the next computation.

Then I worked on being able to get all the journey for either all the id or specific id given the pause time for a journey. The cc_data.csv was using the names of the individual, so I had to write a jar file that will from the car_assignement.csv file get the id attached to each name and change the names by the id in the cc_data.csv so all the data are using the same denomination.

Finally, I wrote a jar file that allows me to get from the journey file and the cc_data.csv some deduction on places location. I can then regroup all this information by places, and eventually use an other jar file that allows me to calculate approximately it's location by doing the average of all the position for a given place.

Incidentally, I also wrote a jar file that displays the location of the end of each journey on the map of the city, this allows me to visually see where the peoples are stoping their trip.

All the jar file is very specific, and can't be used in other data set, however, during the end of the week, I've been working on making some jar file reusable for instance the jar file that allows me to sort chronologically or the grouping jar file.

Le stage est divisé en deux projets principaux, le développement d'un framework entrant dans la continuité d'un projet d'analyse ayant pour but d'améliorer la sécurité et la détection des comportements anormaux. Le framework permet d'exécuter des fichiers exécutables JAR permettant ainsi d'améliorer le travail collaboratif en posant certaines conventions et donc faciliter la réutilisation d'outils intelligents, il est entièrement développé en Java avec l'IDE eclipse. La seconde partie du stage est le développement d'outils intelligent pour mettre le framework en action et démontrer son utilité.

Mots-clés : Java, eclipse, gestion de projet, analyse de besoin, implémentation et design de système, scénario de test.

The internship is divided into two projects, the design and implementation of a java-based workflow system made for use in a greater project named "Cybersecurity analysis Platform", the software will give a needed help to support collaborative analysis and reapplication of intelligent tools. This software was fully programed using eclipse. The second project is the programming of intelligent tools capable of being used by the software to prove the utility of such software in a professional environment.

Key words : Java, eclipse, project management, requirement analysis, system design and implementation, testing.