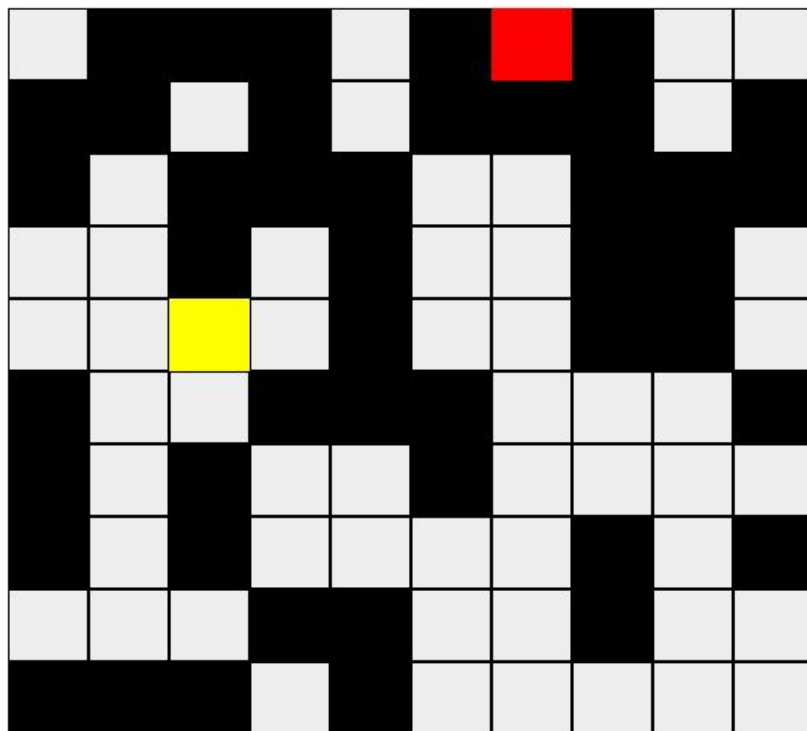


Projets Tuteurés 2.I

L'algorithme d'Ariane



INTRO

Bonjour à vous,

Dans ce rapport d'avancement, nous allons vous parler de notre projet en Java qu'on a réalisé lors du second semestre de notre première année en DUT Informatique. Ce projet tuteuré est le moyen de concrétiser et d'appliquer nos connaissances acquises lors de notre semestre en Java. Le but de ce travail est de s'organiser en équipe afin de produire un programme permettant de simuler les déplacements d'un personnage (ici, Thésée) jusqu'à la sortie d'un labyrinthe. Ce labyrinthe doit être personnalisable par l'utilisateur à travers une interface graphique efficace afin qu'il puisse tester les algorithmes de déplacements du personnage dans divers situations. Et notre simulation aura le rôle de renvoyer une valeur et une moyenne à l'utilisateur.

FONCTIONNALITÉS

ACCUEIL

Après avoir lancé le programme, une fenêtre d'accueil apparaît. Un menu barre est alors accessible et propose plusieurs choix.

L'utilisateur peut générer une nouvelle grille, en sauvegarder une (elle doit être chargée au préalable) ou trouver plus d'informations sur le fonctionnement.

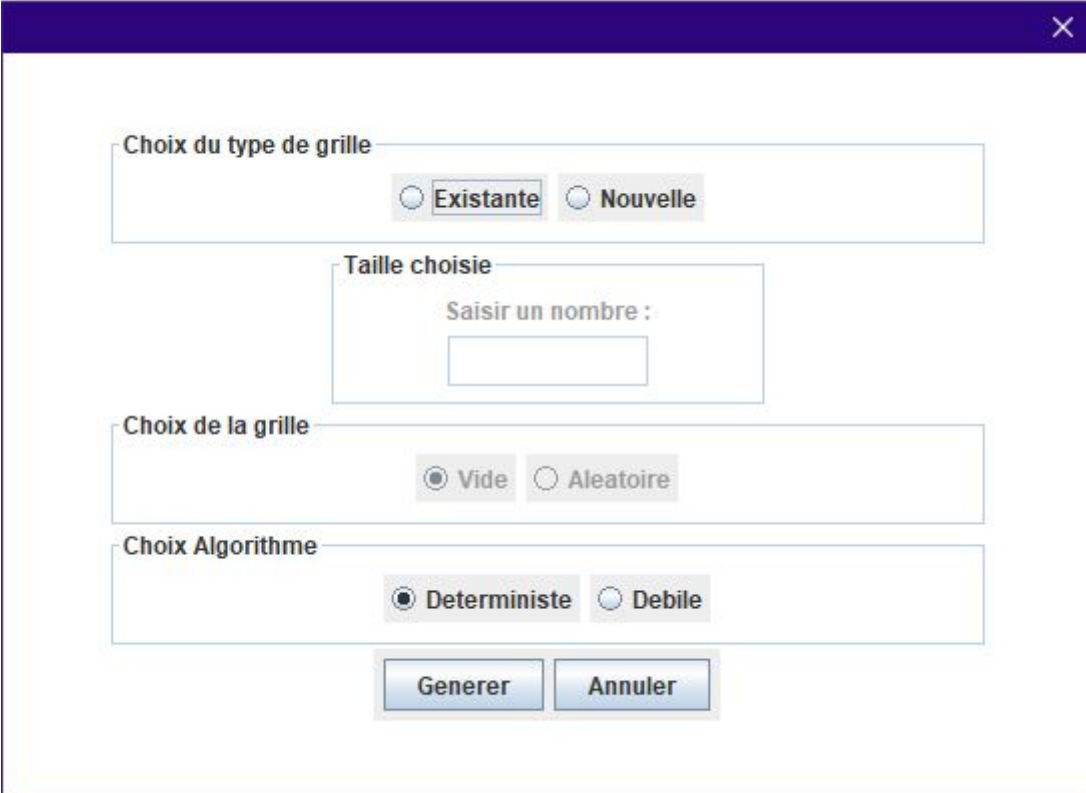


Regardons de plus près ces choix :

- **Informations** : En cliquant sur "?" un bouton "commandes" apparaît. Lorsqu'il est activé, un message apparaît indiquant à l'utilisateur le fonctionnement de chaque touche. Ces touches permettent de modifier la grille générée en attribuant une fonction (Thésée, sortie, mur, vide) aux cases choisies.
- **Générer** : Ouvre une fenêtre de dialogue proposant plusieurs choix à l'utilisateur pour la création d'une grille et du déroulement de la simulation.
- **Sauvegarder** : Ouvre une fenêtre de choix de fichier où l'utilisateur pourra sauvegarder la grille générée dans un dossier choisi. Le mode de fonctionnement de la sauvegarde (et du chargement sera détaillé plus tard).

FORMULAIRE

Le formulaire est une interface qui permet à l'utilisateur de choisir entre plusieurs options, ses préférences pour la simulation. En fonction des choix effectués, il sera renvoyé des données différentes :



The image shows a software window titled "FORMULAIRE" with a close button (X) in the top right corner. The window contains four main sections, each with a title and a set of options:

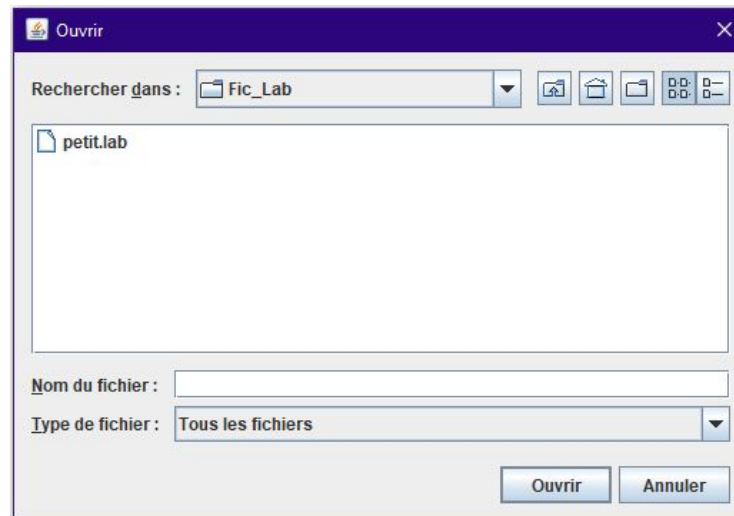
- Choix du type de grille**: Contains two radio buttons, "Existante" (selected) and "Nouvelle".
- Taille choisie**: Contains the text "Saisir un nombre :" followed by a text input field.
- Choix de la grille**: Contains two radio buttons, "Vide" (selected) and "Aleatoire".
- Choix Algorithme**: Contains two radio buttons, "Deterministe" (selected) and "Debile".

At the bottom of the window, there are two buttons: "Generer" and "Annuler".

- Choix de grille :

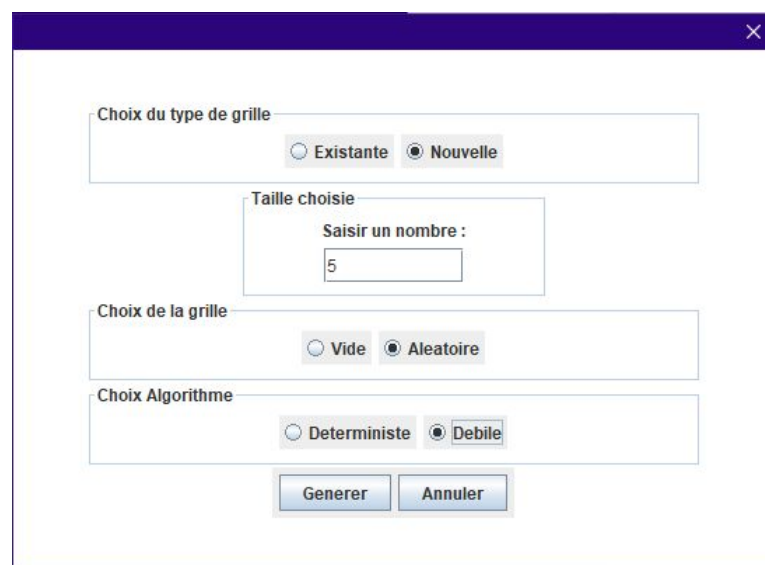
Ici l'utilisateur peut choisir entre charger une grille existante ou créer une nouvelle grille.

S'il choisit de charger une grille, alors il ne pourra que choisir son type d'algorithme, et lorsqu'il aura validé son choix en cliquant sur générer, une fenêtre de choix de fichier s'ouvrira. Il n'aura plus qu'à choisir le fichier qu'il souhaite et cliquer sur ouvrir pour lancer la simulation.



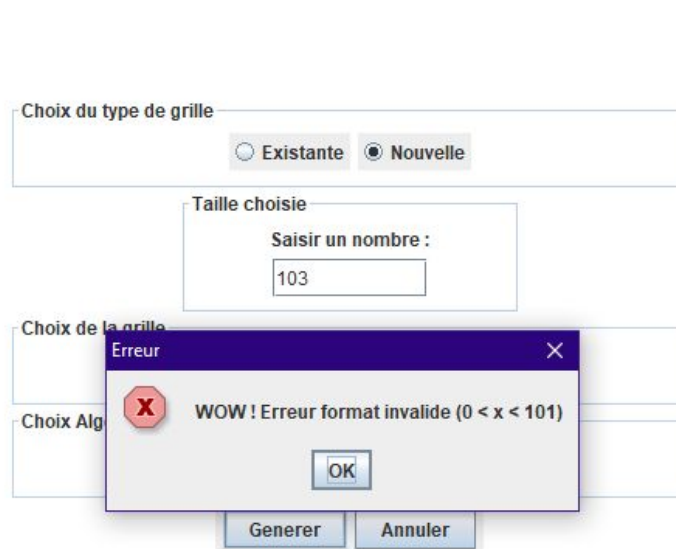
S'il choisit de créer une nouvelle grille, alors des nouveaux choix se débloquent.

Il pourra choisir la taille de la grille, si il souhaite que la grille soit vide ou remplie aléatoirement et l'algorithme de déroulement.

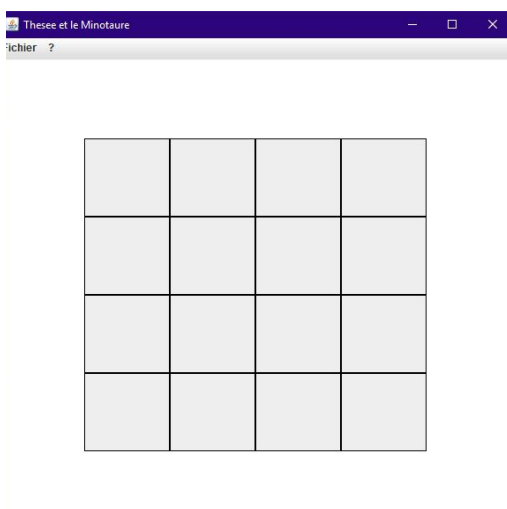


- Choix de création :

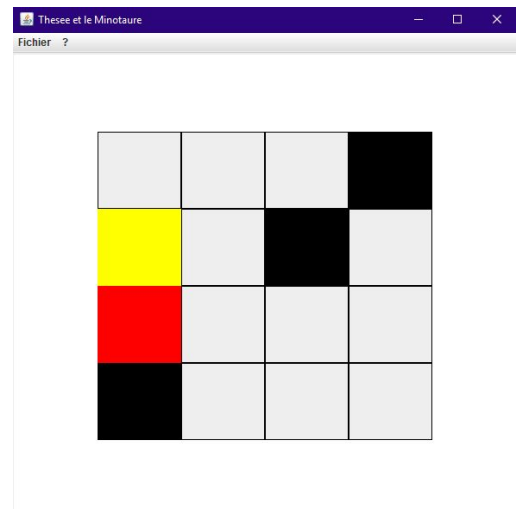
Ayant choisi de créer une nouvelle grille, l'utilisateur devra alors insérer un entier compris entre 0 et 100 (si ce n'est pas le cas un message d'erreur lui avertira du soucis) qui sera la taille de la grille.



Il pourra ensuite choisir si les positions des cases sont créées aléatoirement ou si la grille est vide (il lui sera alors demandé de la remplir manuellement lors du lancement de la simulation).



Grille générée vide



Grille générée aléatoirement

Et enfin il pourra choisir le type d'algorithme de fonctionnement.

- Choix d'algorithme :

L'utilisateur peut choisir entre deux types d'algorithme : le "Débile" et le "Déterministe".

L'algorithme "Débile" fera avancer Thésée de manière aléatoire en fonction des cases vides autour de lui.

L'algorithme "Déterministe" fera avancer Thésée en fonction de sa position actuelle et de ses positions précédentes. Cet algorithme étant plus complexe, nous verrons son fonctionnement plus tard.

SIMULATION

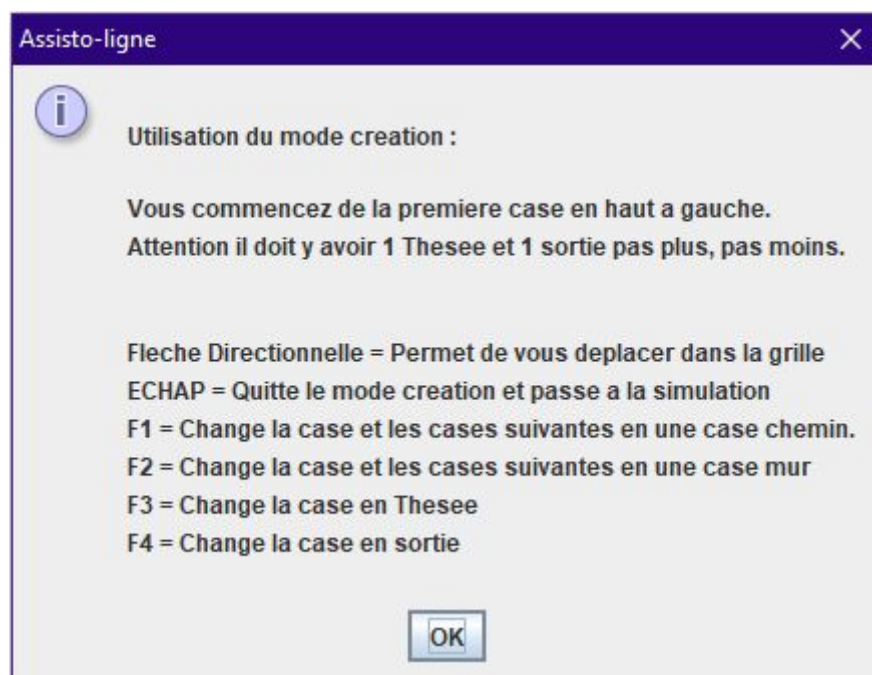
Une fois les paramètres choisis, l'utilisateur peut lancer la simulation.

Elle consistera dans un premier temps de modifier/compléter (dans le cas d'une grille vide) la grille en ajustant les positions des cases noires, de Thésée et de la sortie.

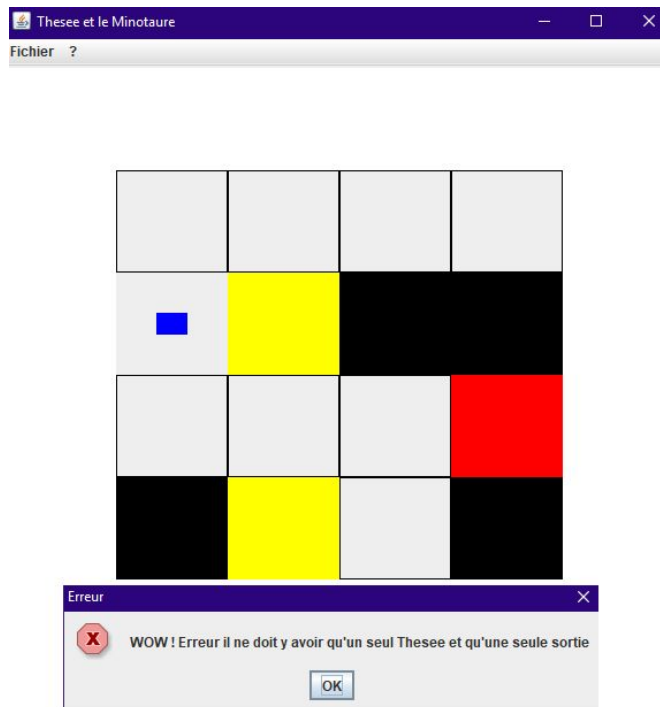
Et dans un second temps de choisir le déroulement de l'algorithme (entre le mode manuel et le mode automatique) qui mènera ou pas Thésée à la sortie.

- Modification de grille :

Quand la grille est chargée, une fenêtre d'information s'affiche, et indique les touches à utiliser afin de pouvoir modifier la grille.



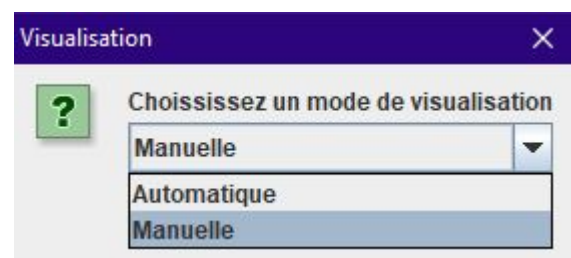
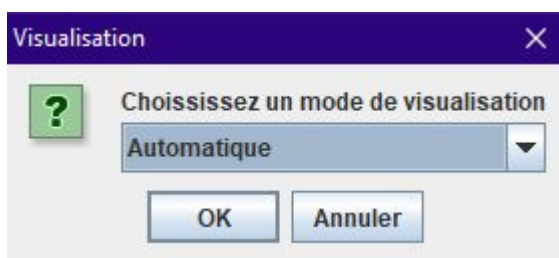
Attention : si l'utilisateur ne remplit pas les conditions indiquées pour la création d'une grille, un message d'erreur apparaît et oblige l'utilisateur à modifier la grille à nouveau.



Si tout est rempli comme il faut, alors une fenêtre de choix s'ouvre demandant à l'utilisateur de choisir le déroulement de l'algorithme.

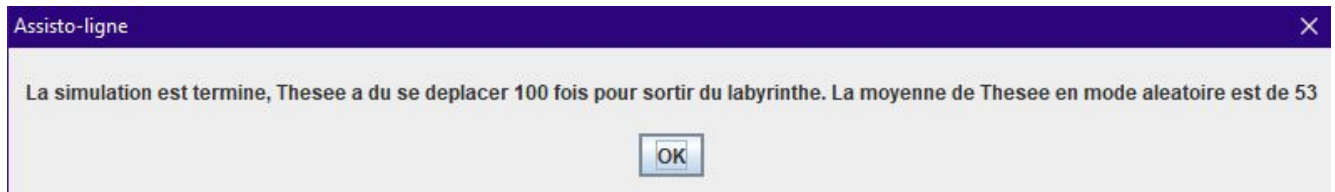
- Choix de déroulement de l'algorithme :

Deux choix sont possibles : un déroulement manuel et un déroulement automatique.

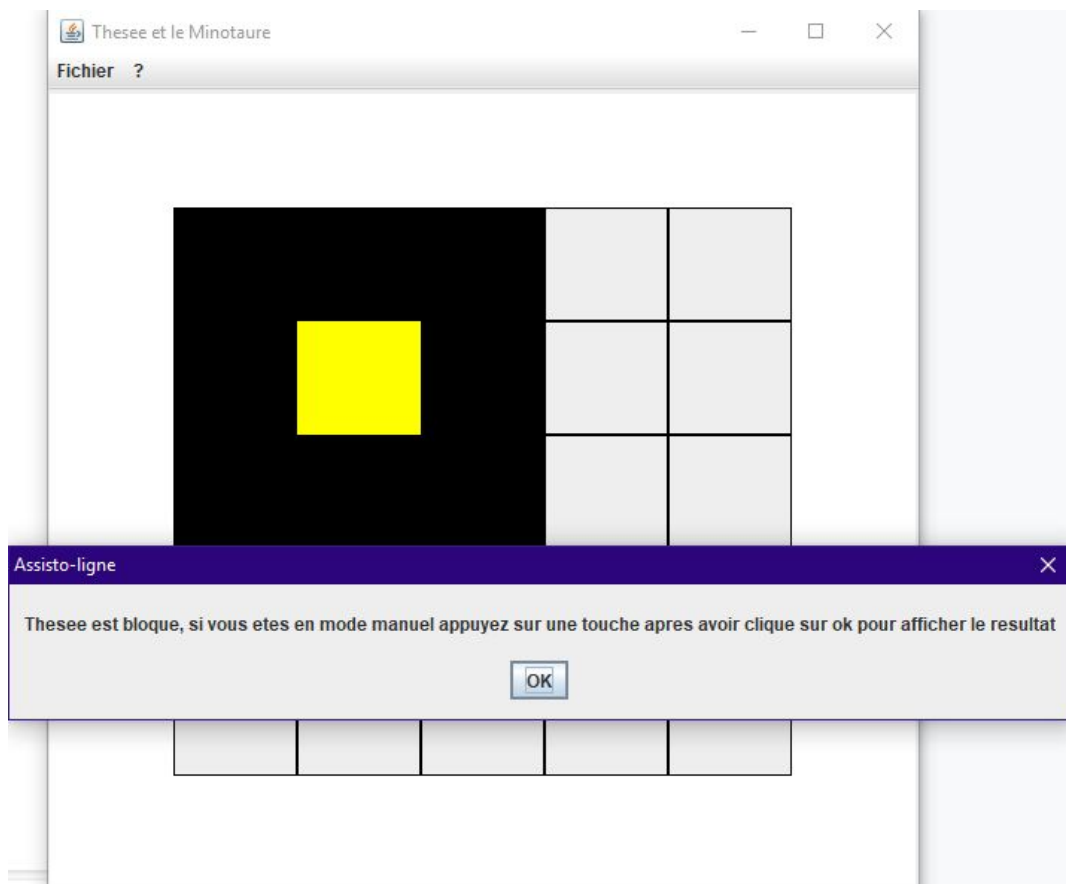


Si l'utilisateur choisit le mode manuel, alors il devra appuyer sur n'importe quelle touche pour faire avancer Thésée d'une case. Il pourra ainsi voir en détail le déplacement et les choix opérés par l'algorithme.

Si l'utilisateur choisit le mode automatique, alors l'algorithme se déroulera et l'avancement de Thésée ne sera pas visible sur la grille. Une fois la simulation terminée, un message apparaît indiquant : le nombre de fois que Thésée a dû se déplacer avant de trouver la sortie du labyrinthe ainsi que la moyenne des déplacements de simulations effectuée avec le même algorithme.

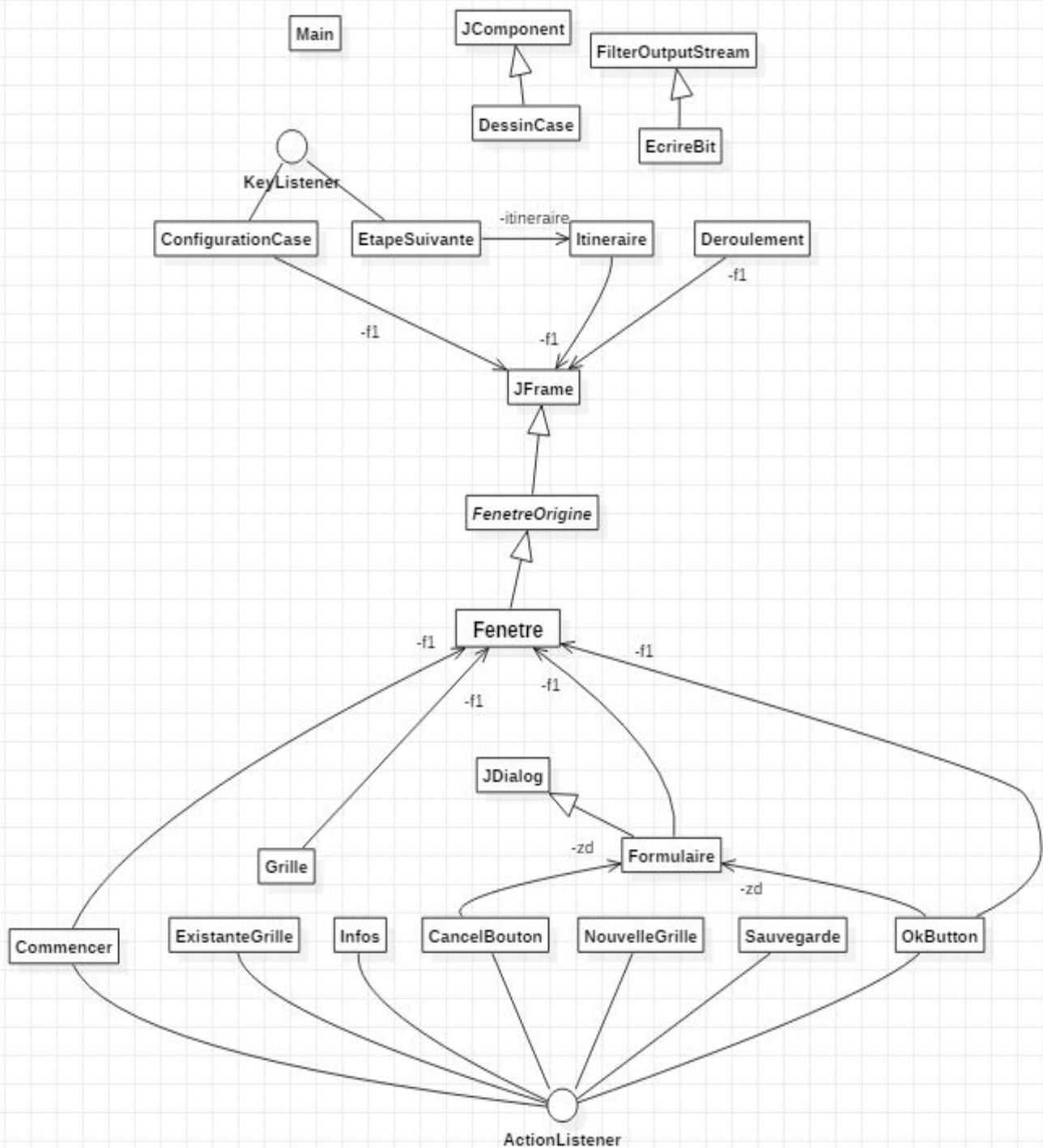


Si Thésée ne trouve pas la sortie, alors un message d'erreur apparaît. L'algorithme déplacera alors Thésée sur une case libre random et relancera le déplacement de Thésée jusqu'à la sortie.



STRUCTURE DU PROGRAMME

Pour la construction de ce programme, on a divisé chaque fonctionnalité en plusieurs classes contenant des méthodes. Le tout forme une structure que voici :



Certaines de ces classes sont des réécritures de classes préexistantes et sont utilisées par des classes d'interfaces.

Voyons donc plus en détail cette organisation.

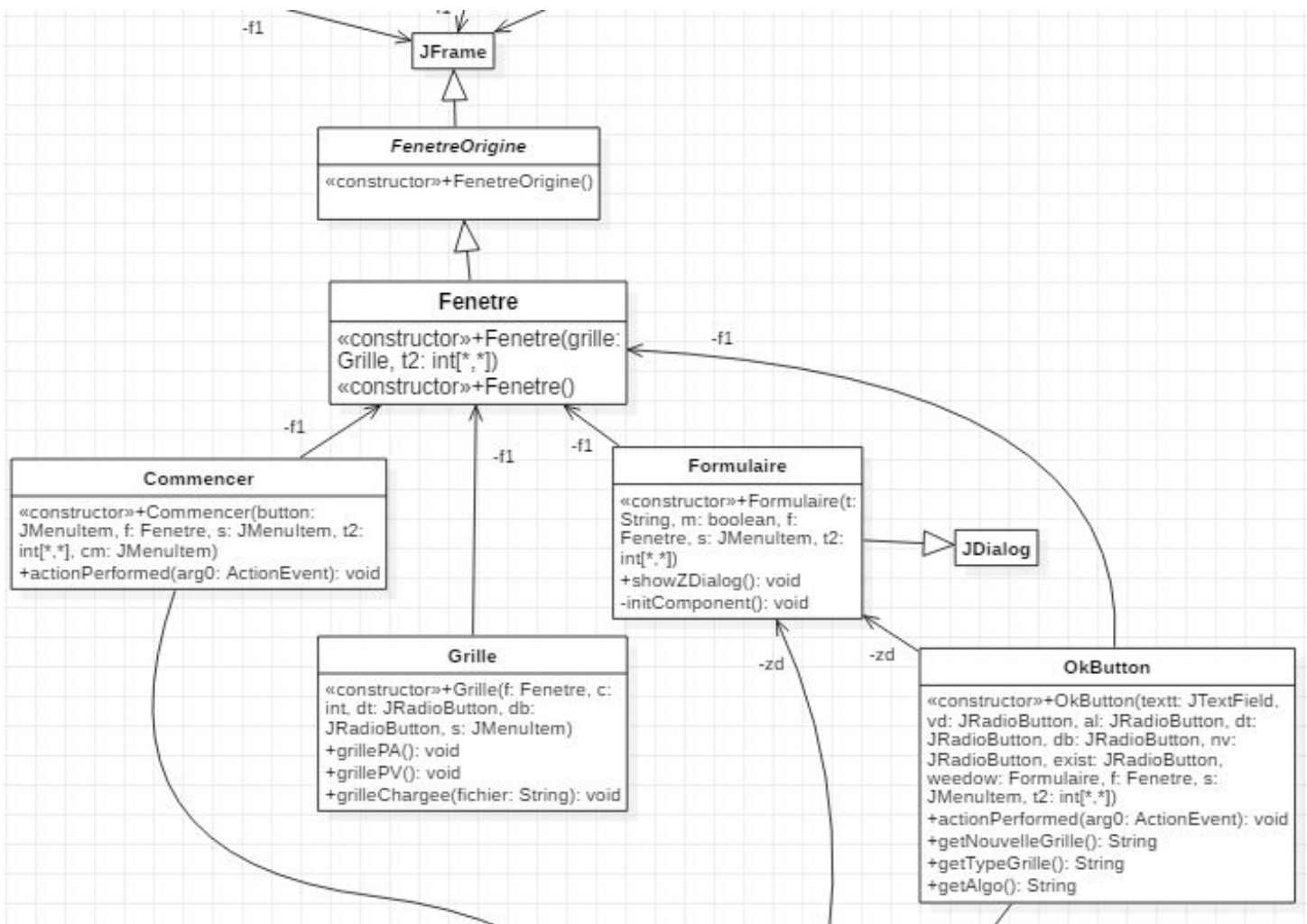
INTERFACE GRAPHIQUE :

- Fenêtre d'accueil :

La fenêtre d'accueil est créée grâce à la classe "FenetreOrigine" qui définit les caractéristiques de base de la fenêtre (taille...).

La classe "Fenêtre" dépend de la classe précédente et est la partie qui contient le menu barre puis par la suite la grille.

Le menu barre contenu dans cette fenêtre, possède un bouton servant à commencer la simulation en ouvrant une fenêtre de formulaire codée par la classe "Formulaire". Ce bouton est lui même désigné par la classe "Commencer"



- Boutons d'action :

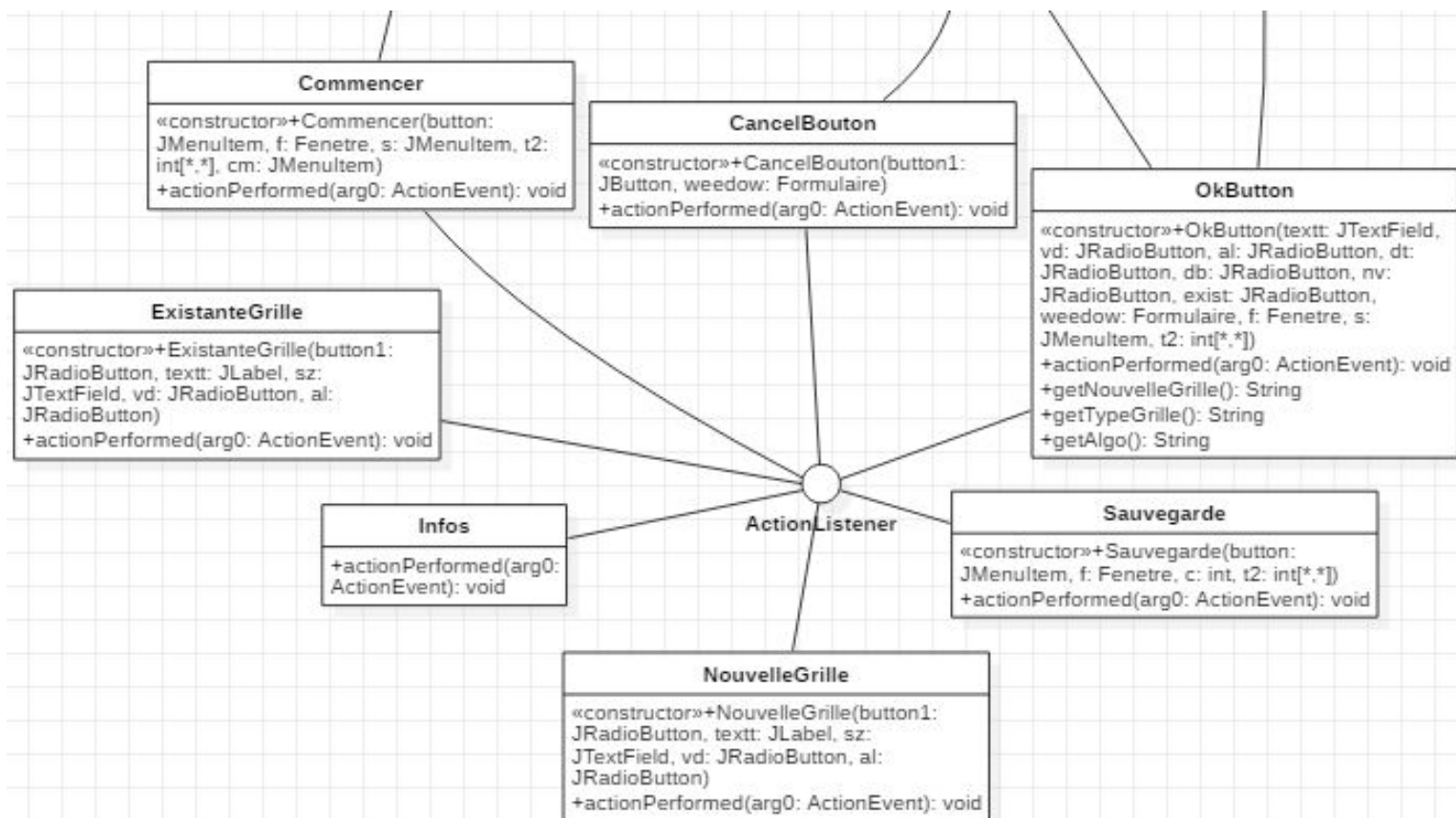
Plusieurs boutons sont disponibles tout au long du programme. Ils sont de types différents et renvoient à des actions spécifiques.

Les boutons du menu barre sont actifs tout au long du fonctionnement du programme (sauf le bouton sauvegarder, classe "Sauvegarde" n'est disponible qu'une fois une grille chargée).

L'utilisateur doit appuyer sur le bouton générer de la classe "Commencer" pour ouvrir le formulaire. Une fois cela fait, il pourra choisir les caractéristiques de la simulation et de la grille.

Après avoir fait ses choix il pourra cliquer sur le bouton générer qui renvoie à la classe "OkButton". Dans cette classe, il y a plusieurs méthodes qui récupèrent les choix effectués et qui lance en conséquence un affichage différent de la grille. Par exemple, si il a choisi de charger une grille existante alors le bouton générer va lancer la méthode de la classe "ExistanteGrille" qui va ouvrir une fenêtre de choix de fichier.

A la suite du chargement de la grille, la simulation et le lancement de l'algorithme peut commencer.



ALGORITHMIE :

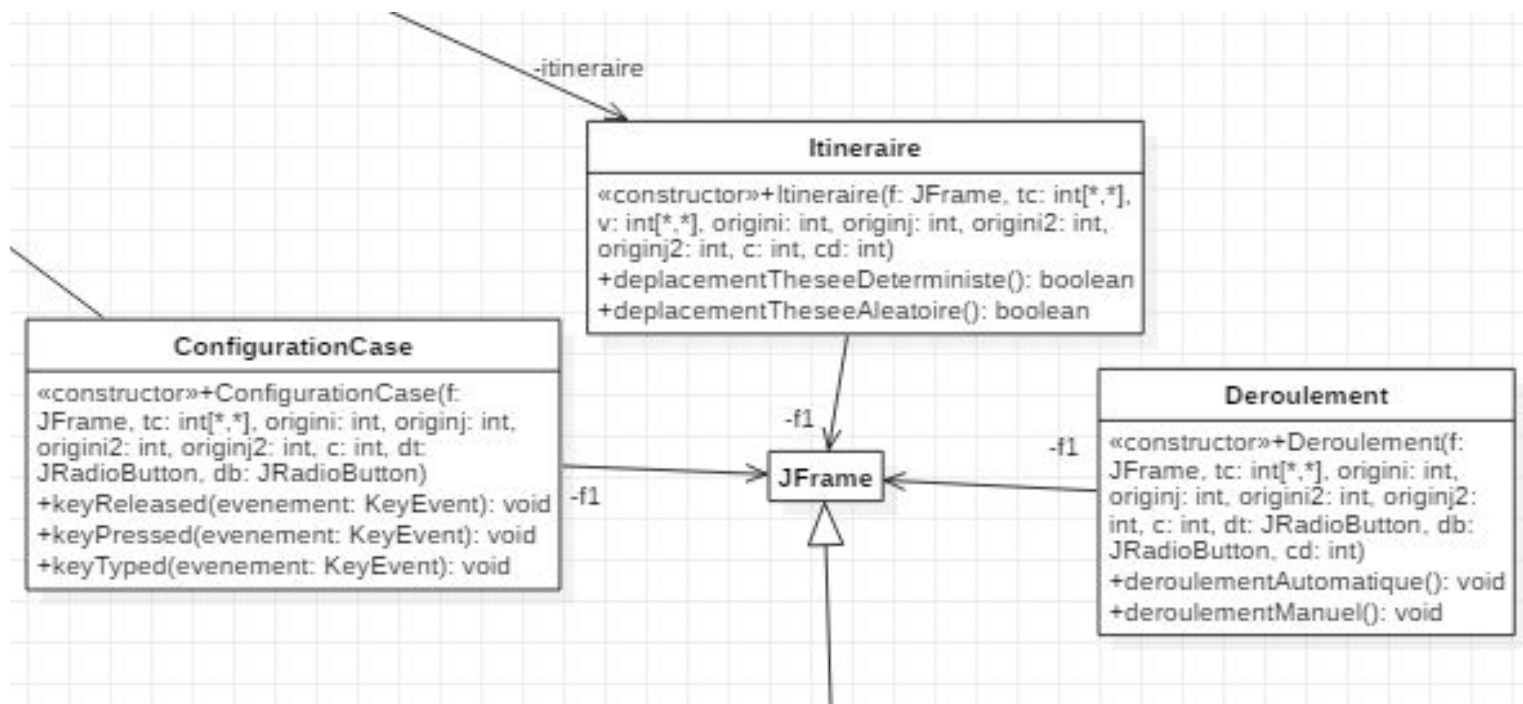
Afin de mener Thésée à la sortie, plusieurs algorithmes sont disponibles le "déterministe" et le "débile" (le fonctionnement de l'algorithme "déterministe" sera détaillé par la suite). Il est aussi possible à l'utilisateur de modifier la grille affichée afin de simplifier/complicuer le déplacement du personnage.

- Action sur la simulation :

Tout d'abord, l'utilisateur pourra modifier la grille. Ces modifications sont permises grâce à la classe "ConfigurationCase" et les méthodes qu'elle contient (des méthodes liées à des actions de touches de claviers qui seront détaillées plus tard).

Ensuite, il devra choisir un mode de fonctionnement qui est actionné grâce à la classe "Deroulement". Dans cette classe, deux méthodes sont définies une pour le fonctionnement automatique ("deroulementAutomatique") qui exécutera la simulation de déplacement sans l'aide de l'utilisateur et qui n'affichera pas ceux-ci. Et le fonctionnement manuel ("deroulementManuel") qui exécutera la simulation de déplacement avec l'action de l'utilisateur sur n'importe quelle touche du clavier.

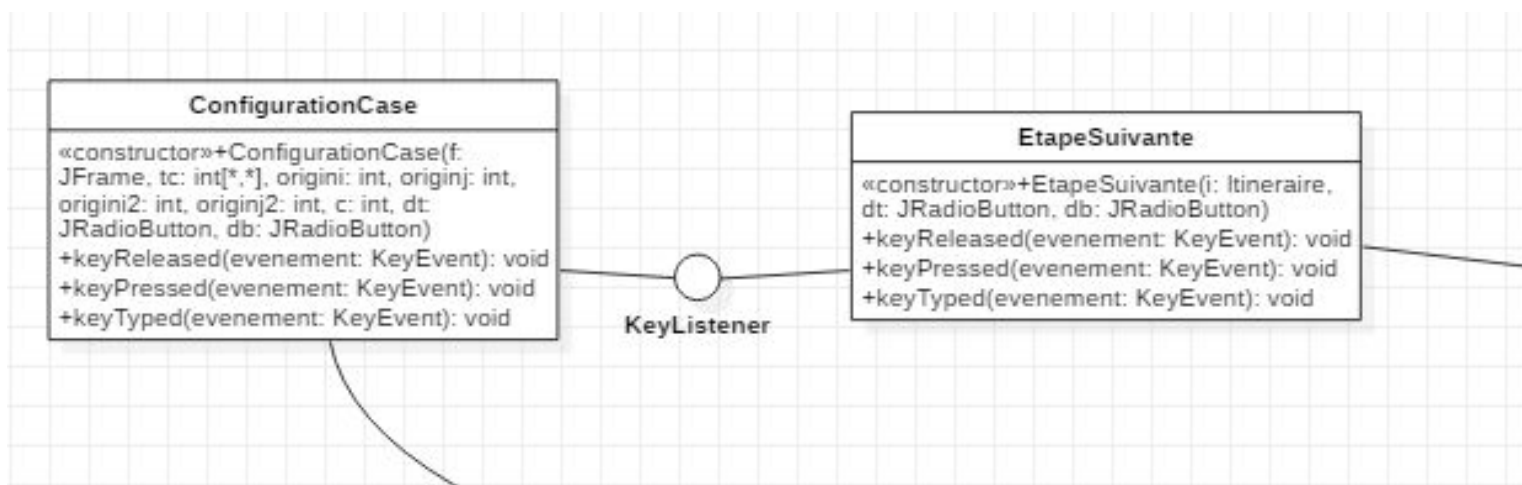
Enfin, l'algorithme choisi peut se lancer. La classe "Itineraire" contient les deux méthodes des deux algorithmes. Elle renvoie les coordonnées à chaque déplacements de Thésée lors du déplacement manuel et permet ainsi de visualiser ces déplacements.



- Modifications au clavier :

Tout comme les boutons, les touches du claviers permettent de modifier l'état du programme. Ici, il y a deux type de modifications : le premier, permet de parcourir la grille et d'attribuer une valeur à chaque case qui codera pour sa nature (Thésée, sortie, mur, chemin). Ceci s'effectue grâce à la classe "ConfigurationCase".

Le deuxième type de modification permet de faire progresser Thésée lors du mode manuel. Ainsi, n'importe quelle touche appuyée fera avancer Thésée dans la case suivante indiquée selon l'algorithme choisi. Ceci s'effectue grâce à la classe "EtapeSuivante".



L'ALGORITHME DÉTERMINISTE

Comment nous avons eu l'idée de cet algorithme ?

Cet algorithme est un des objectifs principaux de notre projet, il a fallu en premier étudier le problème afin d'y trouver une résolution.

On doit faire en sorte que le personnage Thésée trouve la sortie du labyrinthe intelligemment, afin de trouver avec le moins de déplacements possibles la sortie. S'il est impossible de résoudre ce labyrinthe, dans ce cas là, il devra répondre qu'il est bloqué.

Pour commencer à chercher le problème nous avons dessiné et modélisé une grille.

Suite à nos essais, nous tenions une méthode. Partons du principe qu'une grille est une sorte de graphe ou d'arbre. Dans ces cas là pour trouver la sortie à coup sûr en se basant sur la position du personnage et sa mémoire de ses déplacements, on doit parcourir toutes les branches de l'arbre, tous les sommets du graphe. Nous avons pu compléter notre idée en faisant l'analogie direct avec notre cours de Mathématiques sur les graphes et les algorithmes. En effet, cette idée correspond à l'algorithme de parcours en profondeur vu en cours que l'on applique concrètement sur un labyrinthe.

À partir de ce concept et de cet algorithme, nous dûmes le coder.

Comment se structure t-il ?

Pour que Thésée voit autour de lui et fasse son choix correctement, nous utilisons les coordonnées de Thésée afin de donner les types de case qui l'entoure préalablement disposés dans un tableau à deux dimensions correspondant à chaque case de la grille.

Par ailleurs, nous avons utilisé un tableau à deux dimensions correspondant aussi à la grille dans lequel on référence si on a visité 0, 1 ou 2 fois une case. Le but étant de ne pas passer plus de 2 fois sur la même case.

Nous avons dû choisir un moyen de structurer la mémoire de ses déplacements. Nous avons décidé de se servir de la classe Java "Stack" qui nous permet d'utiliser la structure de données correspondant à la pile traduit en français.

Qu'est ce qu'une pile ?

Une pile appelé "LIFO" en anglais pour "Last In, First Out" correspond à un moyen de ranger les données spécifiques. Dans une pile la dernière informations rentrée est la première sortie, ce qui l'oppose donc au concept de la file.

Comment nous avons composé l'algorithme ?

Cet algorithme est constitué d'un ordre de conditions correspondant à un parcours en profondeur.

Thésée doit d'abord regarder autour de lui afin de vérifier qu'il ne se trouve pas à côté de la sortie et si les cases autour de lui sont des murs. Sinon il vérifie ensuite dans le sens des aiguilles d'une montre sur quelle case il doit se diriger. Thésée doit prioriser le fait d'aller sur une case libre. C'est à dire que ce ne soit pas un mur ou qu'elle ne soit pas déjà visitée au moins une fois. Auquel cas il devra alors retourner à la position précédente en effectuant le même processus. Cette méthode représente en quelque sorte le fil d'Ariane dans la légende. Il doit donc rembobiner le fil lorsqu'il se retrouve à la fin d'une branche afin de retourner à l'intersection précédente et de visiter les autres branches. Enfin si il a déjà visité 2 fois toutes les cases autour de lui, c'est à dire avoir fait un aller/retour dessus cela veut forcément dire qu'il est déjà passé par ce chemin de la grille. Ce qui veut dire que depuis l'origine de l'arbre (la position initiale de Thésée dans la grille / le graphe) Thésée aura vérifié tous les cheminements possibles : se retrouvant donc bloqué à la position initiale. Il est donc capable d'en conclure qu'il est bloqué et que cette grille est impossible à résoudre.

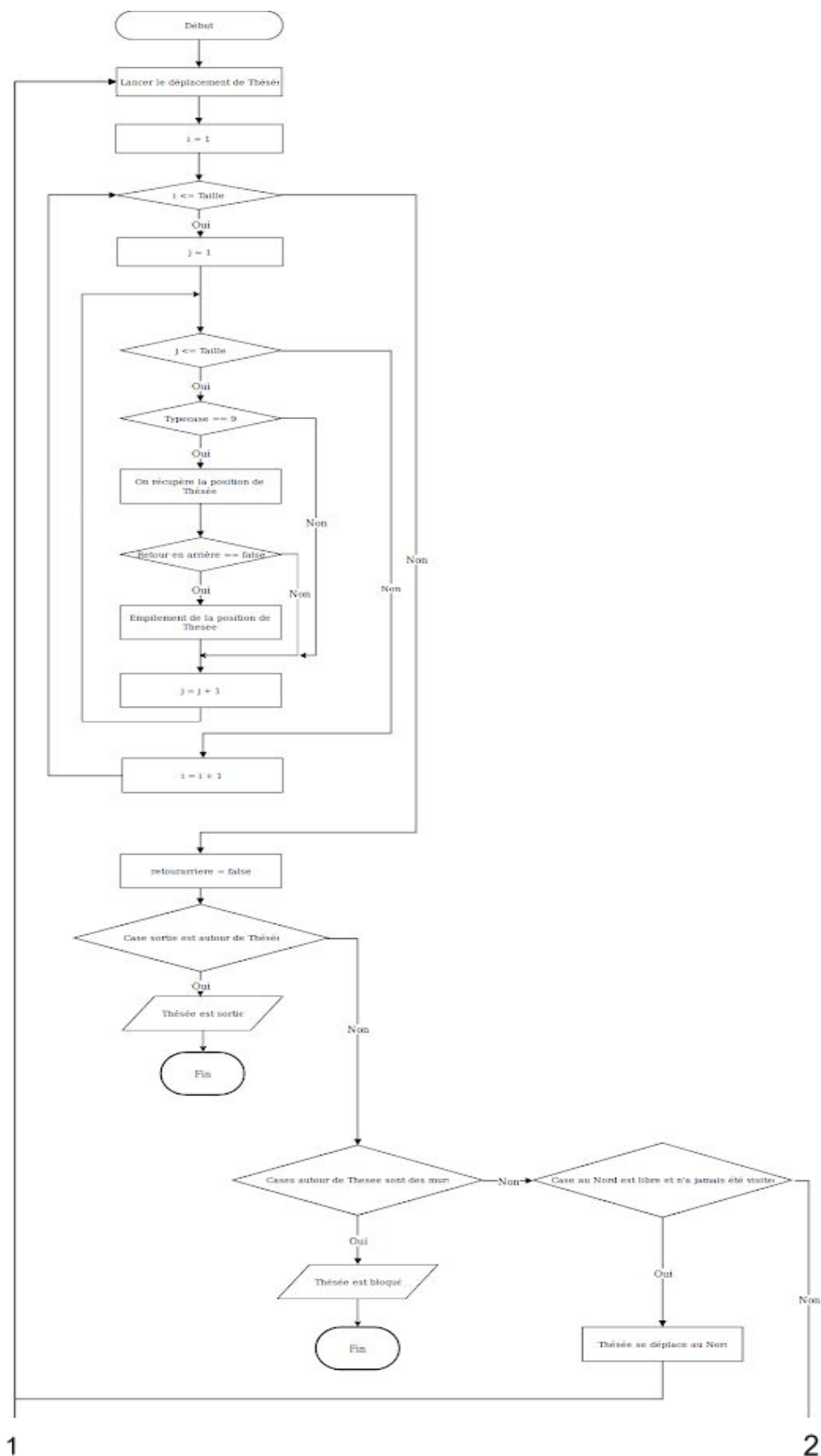
En résumé, il effectue ces tâches dans un ordre rigoureux :

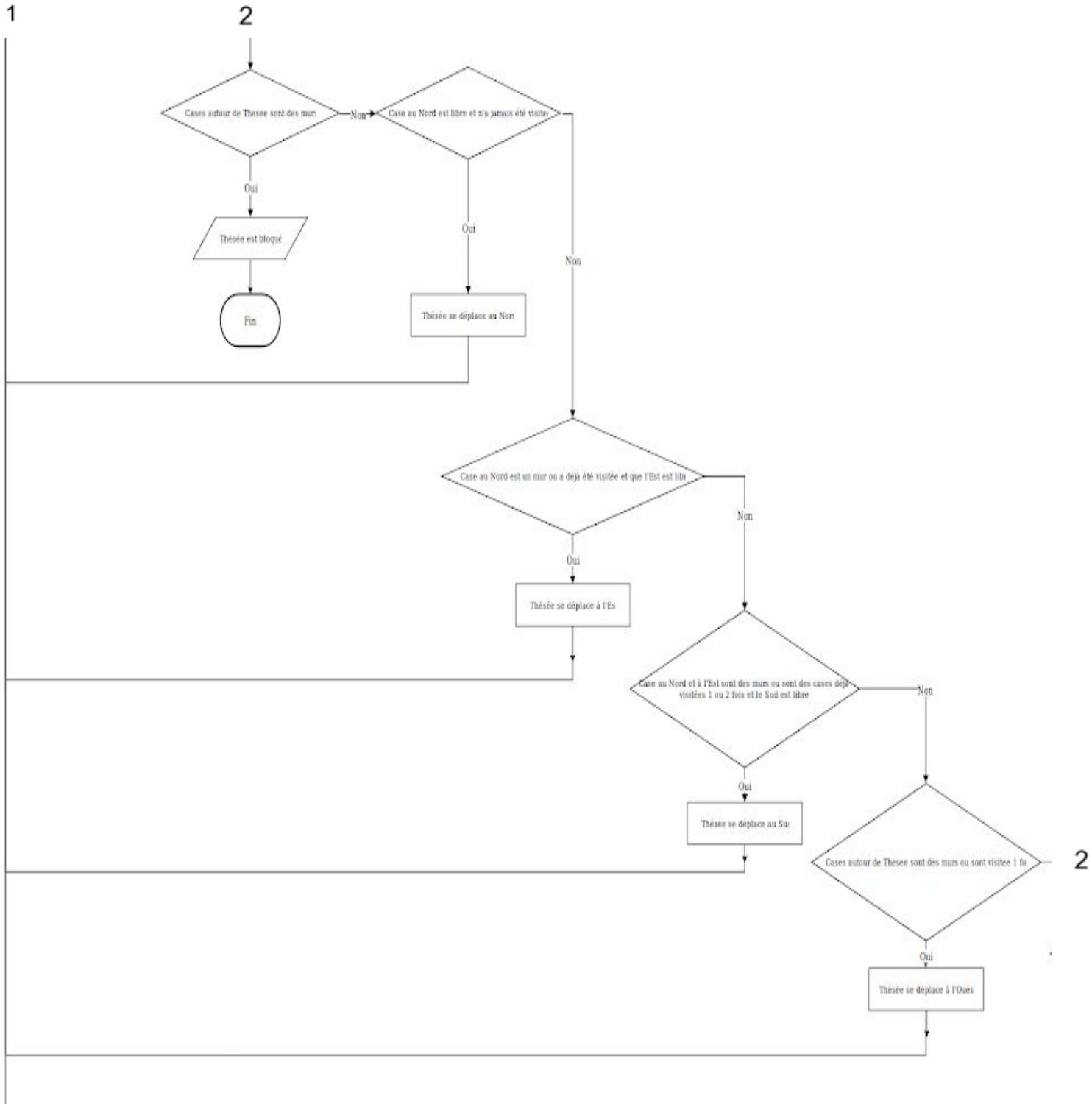
- On récupère la position de Thésée et on l'empile seulement si l'instruction de déplacement précédente était un retour en arrière (dépilement).
- Si la sortie est dans son champ de vision alors il sort.
- Sinon si les murs l'entourent, Thésée est donc bloqué
- Sinon si la case du Nord est libre*, il y va.
- Sinon si la case de l'Est est libre*, il y va.
- Sinon si la case du Sud est libre*, il y va.
- Sinon si la case de l'Ouest est libre*, il y va.
- Sinon si les cases tout autour de lui sont des murs ou visitées 2 fois, il détecte qu'il est bloqué.
- Sinon si les cases tout autour de lui sont des murs ou visitées 1 fois, retourner à la case précédente.

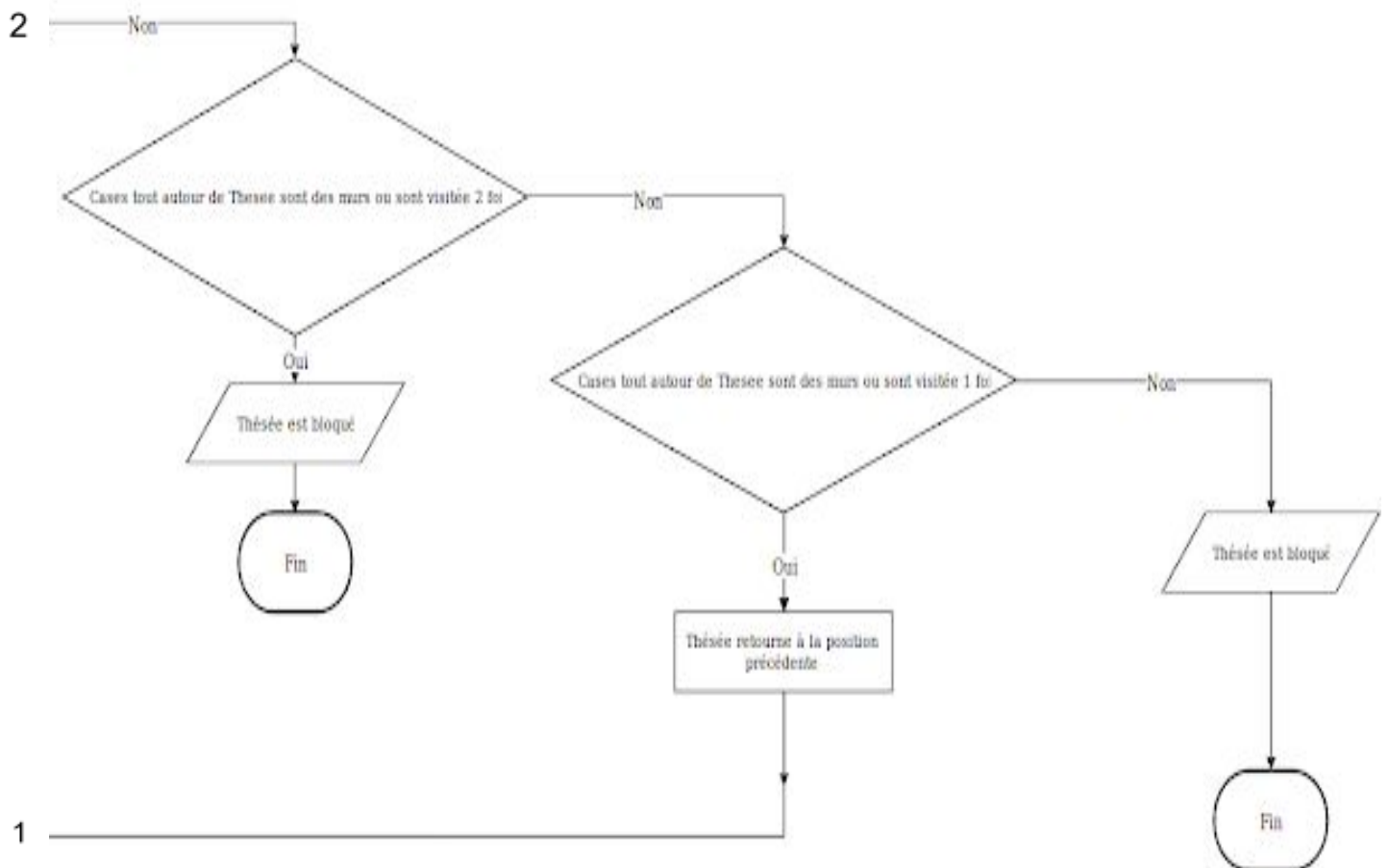
(*libre = une case non visitée qui n'est pas un mur)

Algorithme :

Ci-dessous vous pouvez retrouver la représentation de l'algorithme déterministe sous forme d'algorithme. Si vous voulez l'observer de plus près, notre algorithme vous sera fourni dans le répertoire "Rendu_Final" sous forme d'un fichier .PNG s'intitulant "Algorithme_AlgoDeter.png".







CONCLUSION

Jean-Baptiste SANCHEZ

Pour ma part ce projet à été très enrichissant dans le sens où il m'a permis de réutiliser et approfondir mes connaissances initiales dans le langage Java.

De plus, à mon avis, c'est un bon challenge qui nous a permis de travailler en équipe. La coopération et la rigueur dans ce type de travail collaboratif sont majeures. C'est un bon entraînement pour travailler plus efficacement en groupe et/ou même seul. En effet, à travers nos échanges, nous avons l'occasion de pouvoir se partager des idées, des manières de conceptualiser les choses ainsi que d'améliorer nos méthodes de travail, ce qui n'est vraiment pas négligeable. Ce fut d'autant plus pratique car mon binôme était présent et travaillait avec moi. Ce projet m'a fait naître des idées de petits projets à faire en langage objet à l'avenir.

Merci d'avoir lu notre rapport d'avancement. Ce fut un travail où nous avons mis beaucoup d'efforts et je suis fier du résultat.

Jean-Baptiste SANCHEZ

Marie PELLIER

J'ai trouvé ce projet intéressant et vraiment utile dans mon apprentissage en développement. En effet, il m'a permis de me rendre compte de possibilités qu'offre le Java et de développer mes connaissances sur le sujet. J'ai même préféré ce projet à celui réalisé précédemment en C, bien que les attentes étaient plus nombreuses.

De plus, le travail en groupe fut agréable; Pendant les vacances j'avais une perte de motivation (cadre oblige), mais mon binôme m'a aidé et m'a encouragé, chose très appréciable lors d'un travail de groupe.

On a su s'organiser et se répartir les tâches convenablement en fonction des préférences de chacun, ce qui fait qu'il y a eu une bonne cohésion de groupe.

Ainsi, on a réussi à obtenir un résultat dont je suis plutôt contente, même si deux trois petits détails auraient pu être ajoutés (un design plus poussé...).

Marie PELLIER