EL SAYED RAMY
NUNES Gabriel
ZAMFIROIU Georges
ZHUO Mickaël

PROJET : Le Jeu de la vie

Livrable 1: Conception

CAMPUS
D'ENSEIGNEMENT SUPÉRIEUR
ET DE FORMATION PROFESSIONNELLE

2ème Année de cycle Préparatoire Intégré
Mineure Informatique 2024 - 2025

Sommaire

I. Introduction

A. Contexte du sujet

Le jeu de la vie désigne un automate cellulaire proposé par le mathématicien John Conway. Il décrit l'évolution d'une population de cellules sur un intervalle de temps discret. Les cellules placées dans une grille rectangulaire deux dimensionnelle sont caractérisées par deux états ; elles sont soit vivantes, soit mortes. A l'exclusion des bordures, le voisinage d'une cellule est formé par 8 autres cellules directement adjacentes. Pour passer de l'itération *t* à l'itération *t+1*, l'état des cellules dans la grille est actualisé selon les règles suivantes :

- Une cellule morte possédant exactement trois voisines vivantes devient vivante.
- Une cellule vivante possédant deux ou trois voisines vivantes reste vivante, sinon elle meurt.

Nous devons proposer une implémentation en C++ du jeu de la vie en nous reposant sur les concepts de la programmation orientée objet. En entrée, notre programme consommera un fichier dont la première ligne permettra de spécifier la taille de la grille. Suivra une matrice de booléens, décrivant l'état initial, vivant (1) ou mort (0) des cellules comme dans l'exemple ci-dessous :

```
5 10
0 0 1 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0
0 1 1 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
```

Pour aboutir nous formerons des groupes de 4 étudiants. L'utilisation de GIT par tous les membres du projet sera indispensable sera le fruit d'un travail limité aux membres d'un groupe. L'évaluation portera sur les éléments suivants :

- La réponse au besoin
- La qualité du code
- La robustesse du programme
- La mise en œuvre des concepts de POO
- Le niveau d'aboutissement
- La maitrise du programme par le groupe

B. Objectifs pour le livrable

Pour ce premier livrable qui s'intitule « Livrable 1 - **Conception »**, nous avons pour objectif de présenter sous formes de diagrammes/schémas/algorithmes les différentes fonctionnalités de l'algorithme que nous allons réaliser par la suite et plus particulièrement :

- Un Diagramme de cas d'utilisation.
- Un Diagramme de classe.
- Un Diagramme de séquence.
- Un Diagramme d'activité.

II. Démarche scientifique

A. Diagrammes

Ces diagrammes auront pour objectifs de modéliser visuellement les différents aspects du projet que nous expliquerons au fur et à mesure du livrable dans les parties ci-dessous.

1. Diagramme de Cas d'Utilisation

Commençons par le diagramme de cas d'utilisation Il permet de représenter les services offerts par le système et chaque cas d'utilisation regroupe un ensemble de fonctionnalités :

- Les acteurs : utilisateurs qui interagissent avec un système. Un acteur peut être une personne, une organisation ou un système externe qui interagit avec votre application ou votre système. Il s'agit nécessairement d'objets externes qui produisent ou consomment des données.

- Le système : séquence spécifique d'actions et d'interactions entre les acteurs et le système. Un système peut également être appelé scénario.

- Les objectifs : résultat final de la plupart des cas d'utilisation. Un diagramme réussi doit décrire les activités et les variantes utilisées pour atteindre l'objectif.

- La relation « include » : indique que le cas d'utilisation cible est inclus dans le cas d'utilisation source, il sera donc exécuté lors de l'exécution du cas source.

- La relation « extend » : indique que le cas d'utilisation source étend (au sens « je rajoute un comportement ») les possibilités du cas d'utilisation cible, il sera exécuté par le cas d'utilisation source de manière optionnelle en fonction des conditions prédéfinies.

- La relation de « généralisation/spécialisation » est une relation d'héritage. Le cas qui hérite réalise toutes les capacités du cas hérité, et y rajoute ses spécificités.

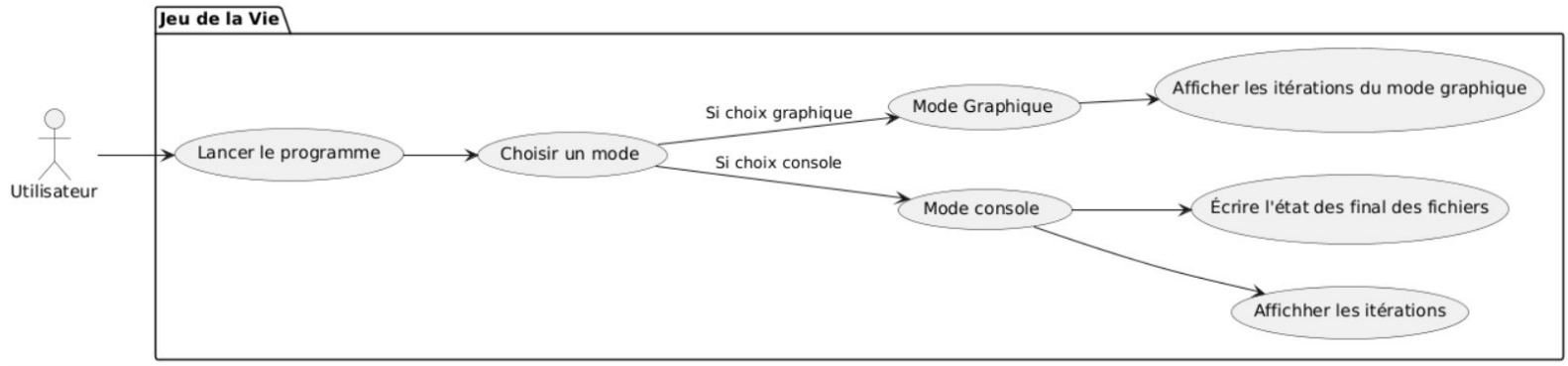


Diagramme du cas d'utilisation

2. Diagramme de Classe

Le diagramme de classe illustre de façon abstrait la structure du code, on y retrouve toutes nos classes ainsi que les relations qu'elles possèdent entre elles. Au sein même des classes nous retrouvons leurs principales propriétés à savoir :

- Le type de classe (en violet si classe virtuel)
- Les attributs et leurs types
- Les prototypes des méthodes
- L'encapsulation (Un point vert pour publique et un carré rouge pour privé)

On a aussi des flèches qui illustre les relations entre classes comme :

- La flèche pleine pour les dépendances
- La flèche creuse pour l'héritage

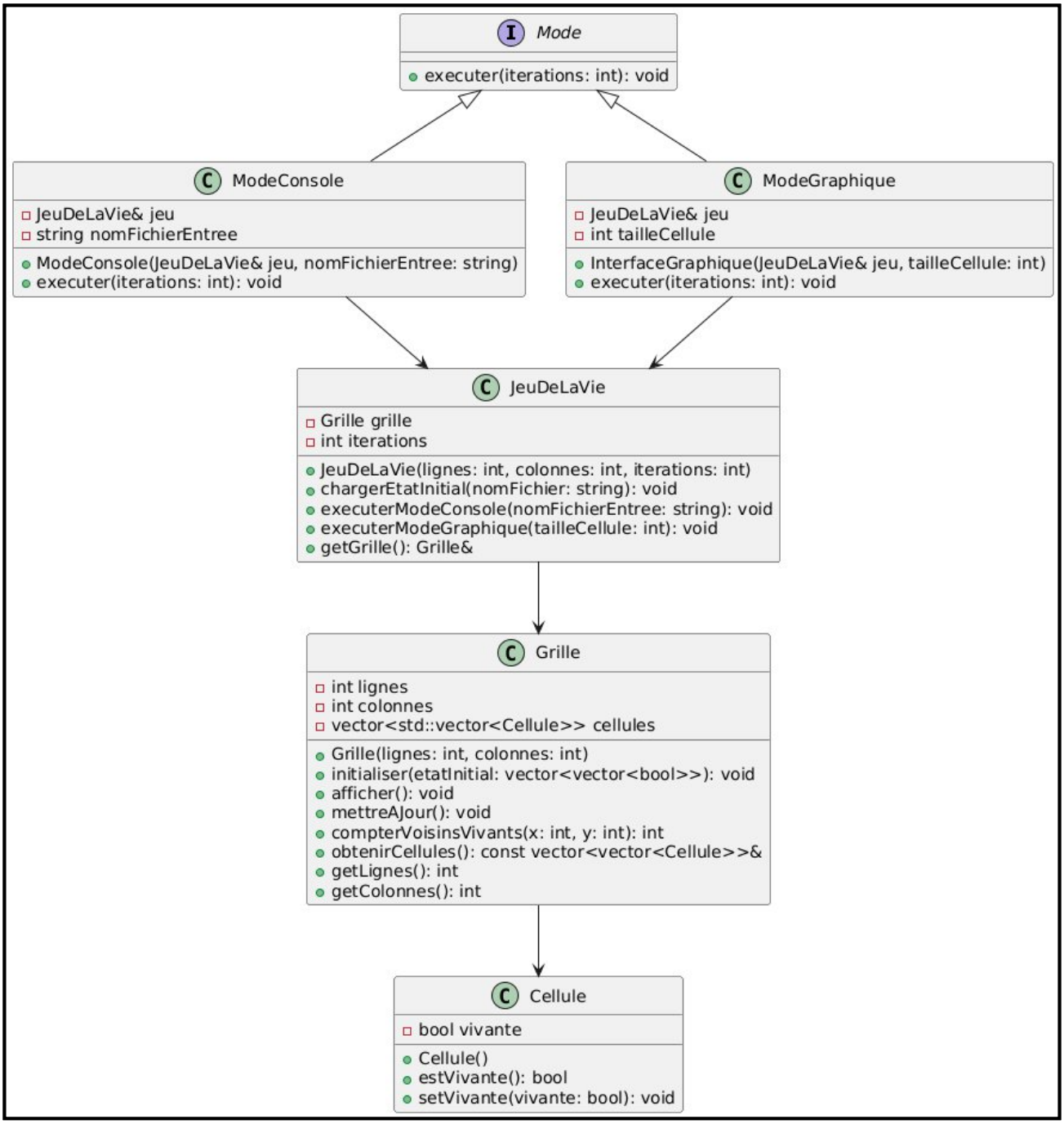


Diagramme de classe

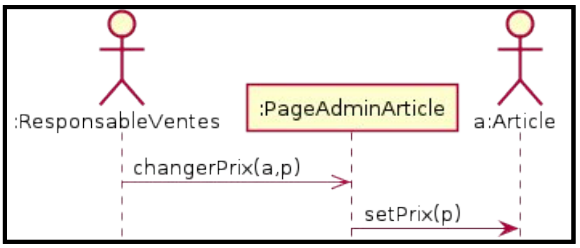
Nous avons la classe virtuelle mode qui prend en héritage les deux classes ModeConsole et ModeGraphique qui permettent de lancer l'un des deux modes d'affichage du Jeu de la vie. Ces dernières sont associées à la classe JeuDeLaVie et qui en sont dépendants, et qui elle-même est associée à la classe Grille qui permet de créer la grille où le jeu va évoluer et pour finir la classe Cellule qui est aussi fortement liée à la classe Grille car c'est cette classe qui va permettre à la grille d'afficher si la cellule est en vie ou non (1 pour est vivant et 0 lorsque la cellule est morte).

3. Diagramme de Séquence

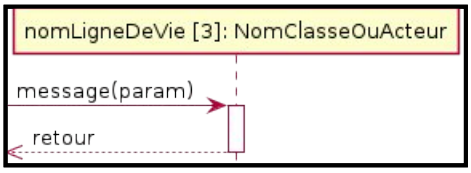
Ensuite, nous avons réalisé le diagramme de séquence. Il permet de montrer comment les différentes classes interagissent les uns avec les autres dans un système et permet de nous aider à comprendre comment, pourquoi et dans quel ordre les interactions se produisent. Les objets au coeur d'un système interagissent en s'échangent des messages tandis que les acteurs interagissent avec le système au moyen d'IHM (Interfaces Homme-Machine).

Tout comme les autres diagrammes, le diagramme de séquence regroupe des fonctionnalités :

- Une interaction : spécifier les opérations nécessaires.



- Une ligne de vie : instance qui représente un participant à une interaction (objet ou acteur)



- Un message : Les principales informations contenues dans un diagramme de séquence sont les messages échangés entre les lignes de vie, ils sont représentés par les flèches du haut vers le bas le long des lignes de vie, dans un ordre chronologique.

  Un message définit une communication particulière entre des lignes de vie.

  Il existe plusieurs types de messages dont les plus courants sont :

  - L'envoi d'un signal ;
  - L'invocation d'une opération (appel de méthode) ;
  - La création ou la destruction d'un objet.

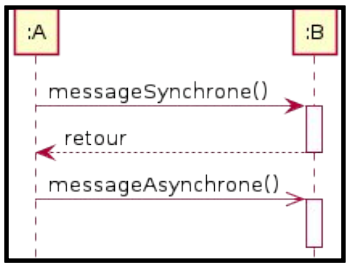
  La réception des messages provoque une période d'activité (rectangle vertical sur la ligne de vie) marquant le traitement du message (spécification d'exécution dans le cas d'un appel de méthode).

- Message synchrones et asynchrone :

  - Un message synchrone bloque l'expéditeur jusqu'à la réponse du destinataire. Le flot de contrôle passe de l'émetteur au récepteur. :

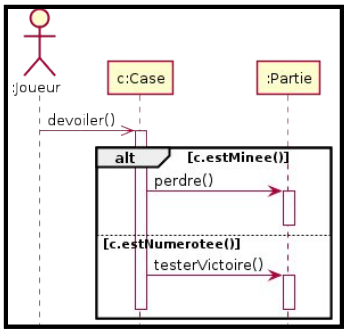
    - Si un objet A envoie un message synchrone à un objet B, A reste bloqué tant que B n'a pas terminé.
    - On peut associer aux messages d'appel de méthode un message de retour (en pointillés) marquant la reprise du contrôle par l'objet émetteur du message synchrone.

  - Un message asynchrone n'est pas bloquant pour l'expéditeur. Le message envoyé peut être pris en compte par le récepteur à tout moment ou ignoré.

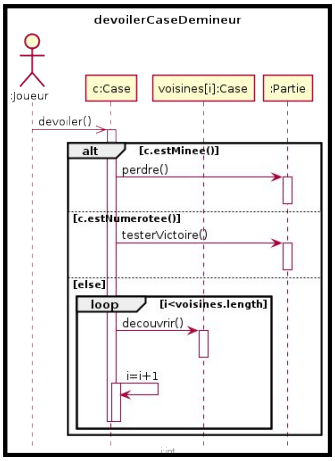


- Fragment alt : C'est un opérateur conditionnel avec les différentes alternatives qui sont spécifiées dans des zones délimitées par des pointillés.

  - Les conditions sont spécifiées entre crochets dans chaque zones.
  - On peut utiliser une clause [else]



- Fragment Loop : C'est un opérateur d'itération qui permet de répéter une commande en fonction d'une certaine condition indiqué entre crochets



Nous passons à la réalisation de nos diagrammes de séquence et nous avons décidé de commencer par le mode console.

Vous trouverez ci-dessous le diagramme de séquence du mode console :

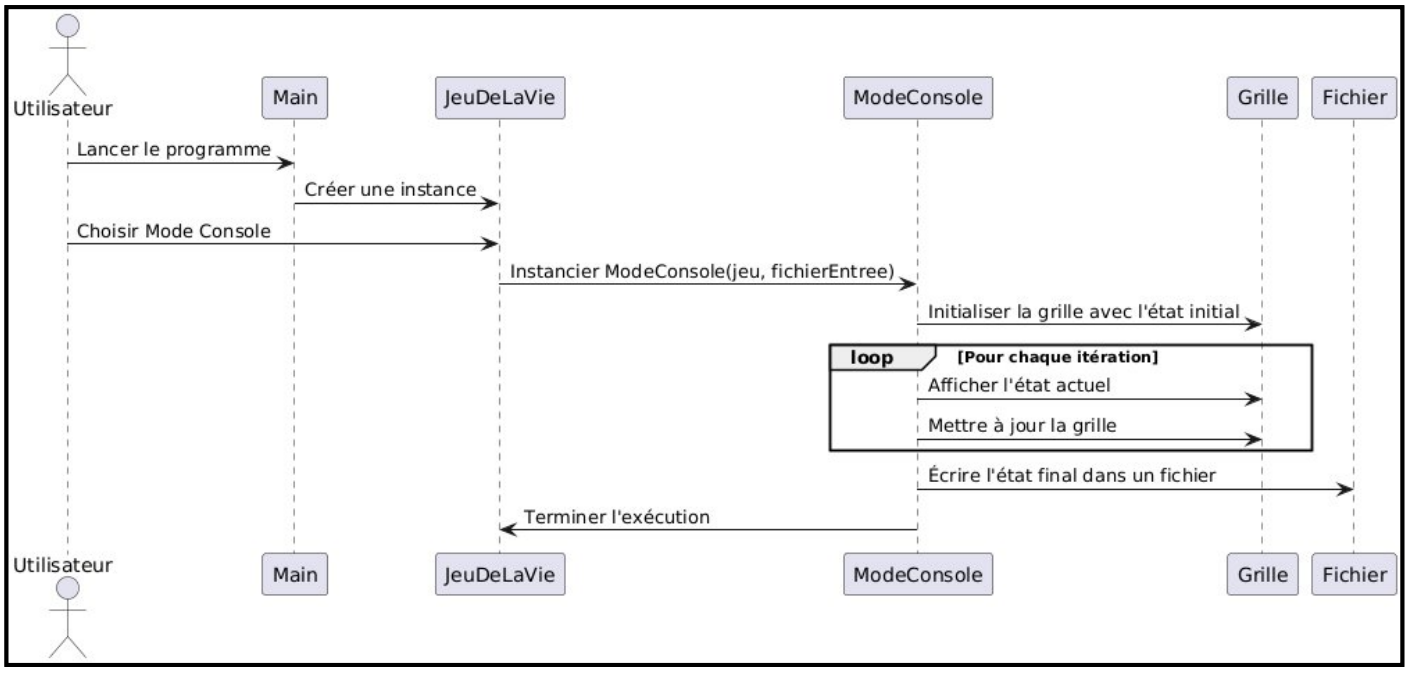


Diagramme de séquence (Mode console)

Et ci-dessous celui du mode graphique :

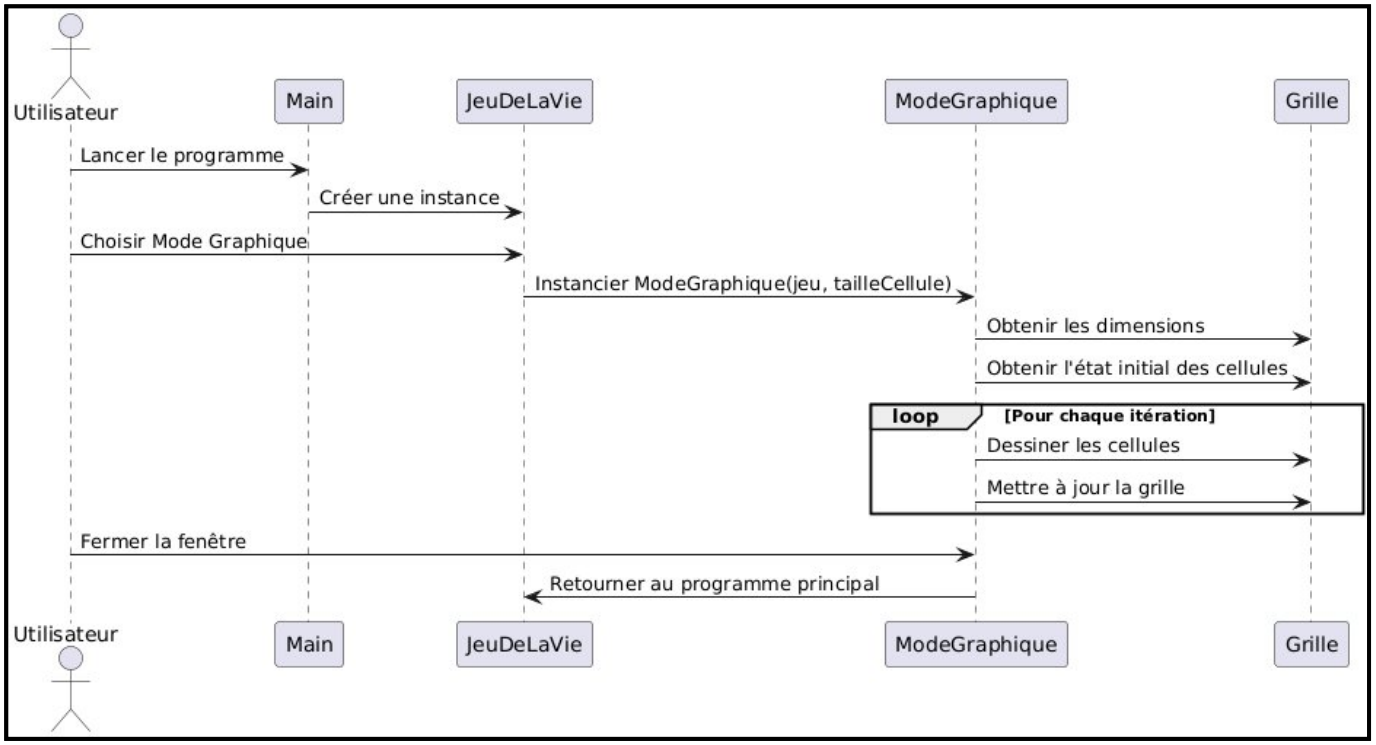


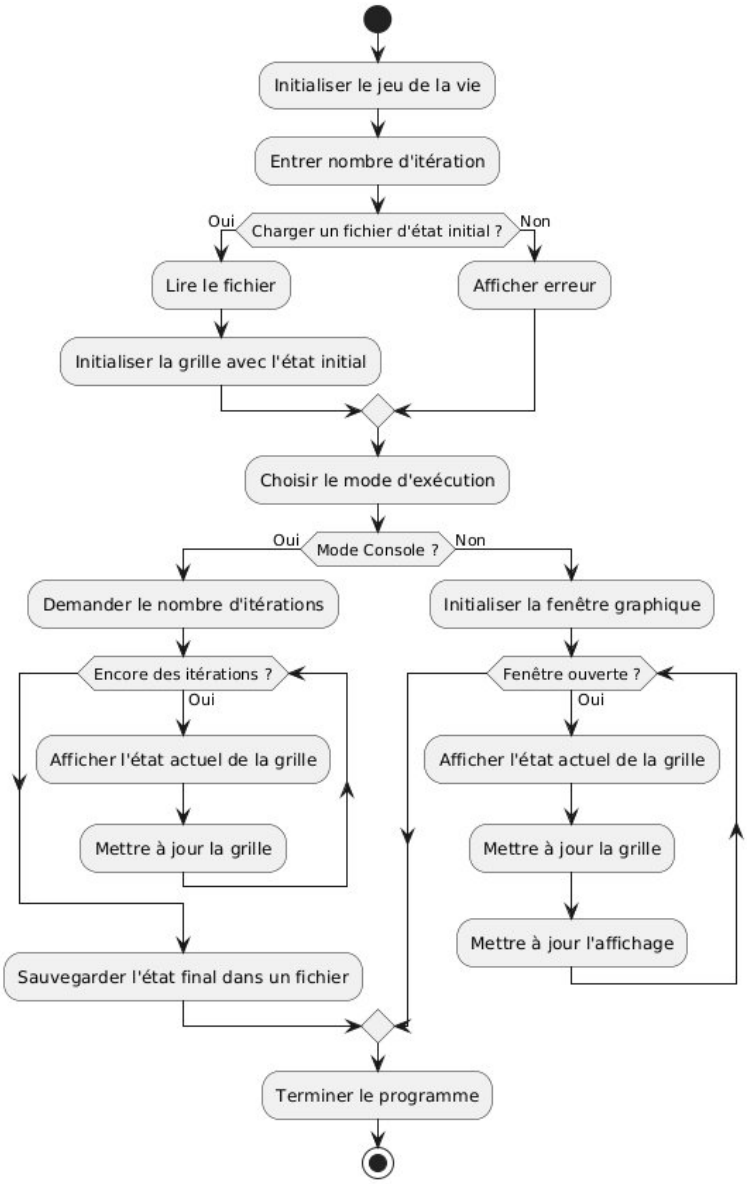
Diagramme de séquence (Mode graphique)

Les deux modes sont relativement similaires. Pour le mode console, l'utilisateur lance le programme et choisit le mode console à partir de la console. Le jeu se lance en initialisant une grille à l'état initial, puis met à jour la grille au fur et à mesure, à chaque itération, dans la console. Ensuite, il écrit et stocke l'état final dans un fichier que nous pouvons consulter afin d'obtenir une sauvegarde de l'état final du Jeu de la Vie.

En ce qui concerne le mode graphique, l'affichage ne se fait pas dans la console et nous n'avons pas un historique des itérations, mais l'évolution des itérations et le résultat directement sur l'interface graphique ; ce qui est plus joli et plus pratique que la version console.

4. Diagramme d'Activité

Enfin, nous terminons par le diagramme d'activité. Le diagramme d'activités permet de raffiner les cas d'utilisation en précisant les différentes activités/actions nécessaires à sa réalisation. Ici, notre diagramme illustre les différents modes de fonctionnement et les étapes pour passer d'un mode à l'autre.



Dans un premier temps, le jeu de la vie s'initialise puis demande à l'utilisateur d'entrer le nombre d'itération qu'il souhaite. Ensuite, le jeu demande si l'utilisateur souhaite charger un fichier d'état, s'il dit oui alors une lecture du fichier et une initialisation de la grille avec l'état initial sera fait, sinon le jeu affichera une erreur.

Dans un second temps, le jeu demande à l'utilisateur de choisir le mode d'exécution, c'est-à-dire l'activation du mode console. S'il dit oui alors le jeu demande à l'utilisateur d'entrer le nombre d'itérations puis si l'utilisateur souhaite davantage des itérations et s'il dit oui, le jeu va afficher l'état actuel de la grille et la mettre à jour et s'il dit non alors le jeu va sauvegarder l'état final dans un fichier. S'il dit non, le jeu initialise la fenêtre graphique et si une fenêtre alors, le jeu va afficher l'état actuel de la grille puis la mettre à jour et pour finir, mettre à jour l'affichage.

Au sein du diagramme d'activité, on y retrouve plusieurs éléments à prendre en compte afin de mieux le comprendre :

- Le cercle plein signifie le début
- Le cercle partiellement rempli signifie la fin
- Les flèches indiquent seulement l'étape d'après, il n'y a donc aucune relation d'héritage ou de dépendance
- Les rectangles arrondis représentent de simple étape du programme
- Les petits carrés représentent seulement une transition
- Les grands losanges représentent une condition « SI » offrant deux possibilités de sortie

III. Conclusion

A. Bilan Global

Un livrable qui nous permet principalement de nous lancer dans la réalisation du jeu de la vie avec la réalisation des diagrammes d'un côté, et de l'autre vouloir absolument, une optimisation afin d'alléger le code final.

B. Bilan personnel

Ramy : Une cohésion d'équipe au top du top même avec un peu de difficulté au début mais finalement, les diagrammes sont là et c'est une bonne chose de faite.

Gabriel : Un travail fastidieux qui semble être simple en apparence mais qui requiert tout de même une certaine organisation et une certaine réflexion qui n'est pas des plus négligeable pour la réalisation des diagrammes.

Georges : Une réalisation assez longue des diagrammes mais dans l'ensemble, je suis satisfait du travail que j'ai fourni.

Mickaël : Un diagramme assez sympathique à prendre en main comme les livrables du premier bloc, cependant toujours aussi long à faire