SYSTÈME MULTI-AGENTS

Imene OUAHRANI

12/11/2020

TABLES DE MATIÈRES

Introduction

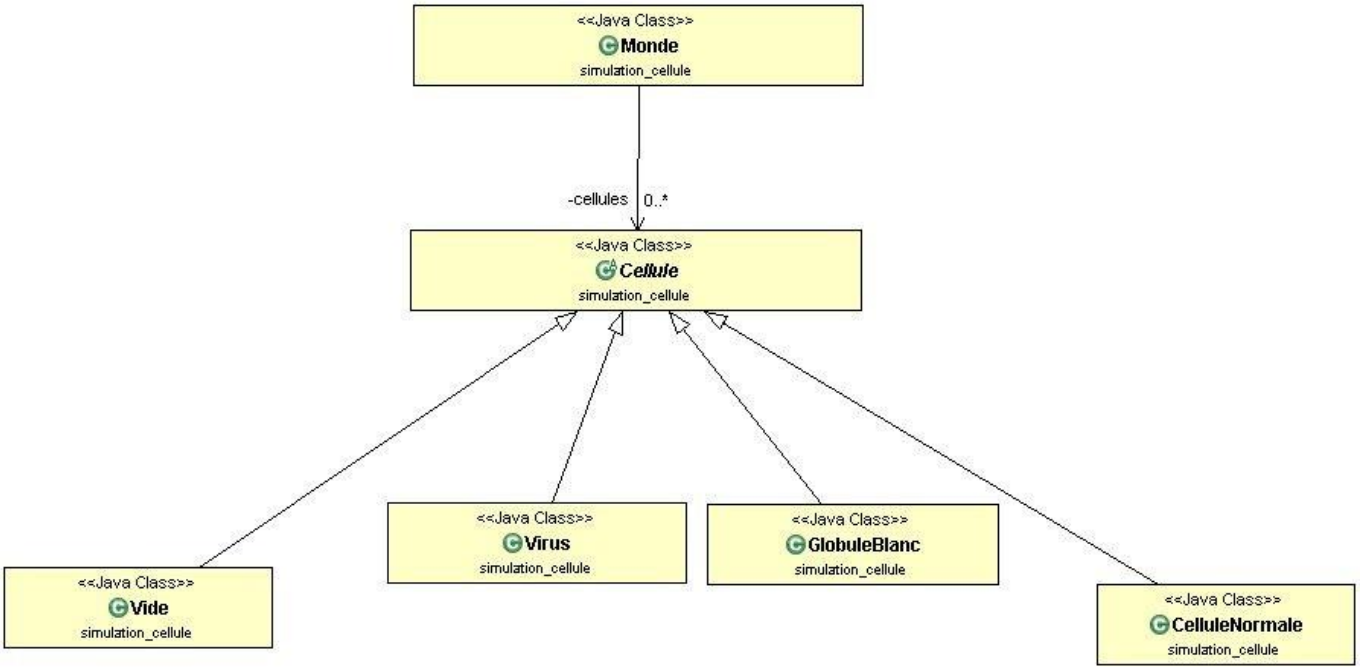
Afin de réaliser notre projet de simulation, nous avons mis en pratique un système multi-agents, implémenté en un langage de programmation orienté objet JAVA, l'un des langages qui nous est très utile et qui répond exactement à nos besoins. Nous avons également eu recours à un générateur de nombres aléatoires implémenté par Mersenne Twister.

Nous avons choisi le domaine de la biologie dans le but de simuler le phénomène des cellules sanguines en interaction avec les virus qui les affectent. Dans cette étude, nous avons modéliser le déplacement des virus par une explosion contaminant toute cellule voisine (les trois premières cellules voisines deviennent malades à leurs tours), et seuls les globules blancs restent inchangeables et continuent à jouer leurs rôles dans la défense de l'organisme contre les agents étrangers (les virus).

Ce rapport présente les comportements résultants de chaque test effectué sur les différentes fonctionnalités du SMA implémentés.

A. Conception de l'univers Multi Agents

Afin de répondre aux attentes demandées dans ce TP, nous avons mis en place un Système Multi Agent qui se rapproche plus ou moins au fonctionnement des cellules biologiques dans le corps humain. Le diagramme de classe ci-dessous illustre les différentes classes implémentées. nous avons regroupé dans une classe abstraite appelée ***Cellule*** les différents types de cellules ou agents manipulés, qui sont représentés par la classe ***Matrice***.



i. L'univers des cellules

Nous avons modélisé les déplacements des cellules ainsi que l'explosion des virus dans un monde cellulaire représenté par une matrice de dimensions modifiables. Les tests de comparaison sont effectués sur une matrice de taille 10*10, tandis que le nombre des cellules varie selon l'expérience et le temps d'exécution souhaité. Le positionnement de chacun de nos agents est choisi aléatoirement.

ii. Les agents

Les agents choisis pour le développement de ce système sont des entités abstraites capables d'agir sur elles-mêmes ainsi que sur ce qui les entourent. Chacune de ces entités a le droit de se déplacer arbitrairement suivant les trois cases qui lui sont voisines tout en respectant certaines conditions imposées.

1. **Les globules blancs:**ce type de cellules appartient au système immunitaire, qui a pour mission de lutter contre les infections. En cas de détection d'un virus, le nombre de globules blancs se multiplient en se déplaçant aléatoirement sur les cases adjacentes pour répondre à leur rôle qui celui d'éliminer les agents étrangers.

2. **Les cellules normales:** connues par des unités biologiques. Ce type de cellules dans notre projet font partie des agents immobiles qui peuvent être contaminés par des virus dans le cas où elles se situent dans l'une des cases voisines du germe. A noter qu'une cellule contaminée est une cellule dont les coordonnées sont remplacées par celles du virus contaminant.

3. **Les cellules vides**: notées par des points pour désigner le vide dans le monde cellulaire.

4. **Les virus:** Le concept des virus consiste à s'exploser après un temps de vie déterminé tout en contaminant trois cases parmis les huits cases voisines. Lors de son déplacement arbitraire, le virus vise à affecter seulement les cellules normales et vides, mais sans toucher aux globules blancs.

Notre étude s'est reposée sur le même principe que dans le monde réel; dans le cas où le nombre de virus dépassent un seuil déterminé, toutes les cellules restantes seront malades ce qui implique la mort de l'être humain.

B. Implémentation du SMA

Lors de notre apprentissage des langages de programmation, nous avons eu un aperçu sur JAVA. Ce qui nous a incité à le choisir pour l'implémentation de ce projet est d'une part pour mieux maîtriser ce langage et étudier son fonctionnement sous plusieurs angles et d'autre part il nous est bénéfique pour la réalisation de cette étude grâce à son concept d'objet.

i. Implémentation des virus

Les germes sont des agents infectieux, ayant leur propre classe qui hérite de la classe abstraite ***Cellule***. A la naissance du virus, sa durée de vie est initialisée à zéro et elle est incrémenté durant l'expérience. Le virus est reconnu dans la matrice grâce à son nom indiqué à 'V' et son état 'true' tant qu'il est vivant.

1. Explosion du virus

Quand le temps de déflagration d'un micro-organisme est égal à celui de sa vie, il explose tout en contaminant trois cellules des huit de ses voisins choisis aléatoirement par l'implémentation de la fonction *explosion, t*out en respectant une condition qui est celle de ne pas dépasser les extrémités de la matrice. Si un virus se situe au bord de cette dernière et que son explosion la dépasse, ceci n'aura aucune conséquence ce qui implique qu'aucune cellule ne sera contaminée.

A noter que lorsqu'un virus s'explose, meurt. Par conséquent son état devient 'False' ce qui entraîne sa disparition de la matrice (voir la figure ci-dessous).

t = n t = n+1

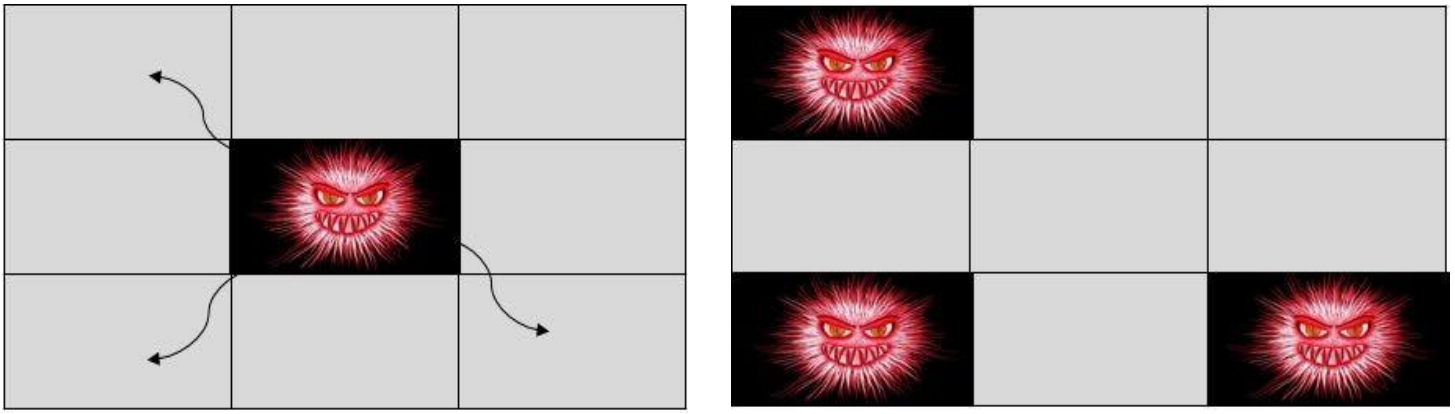


Figure: Explosion d'un virus.

ii. Implémentation des cellules

Comme il est déjà mentionné dans les parties précédentes, toutes les cellules que ce soit des globules blancs, normales ou vides héritent de la même classe abstraite ***Cellule***, le nom, la position, et l'état.

Nous avons initialisé tous les états des cellules à 'true' ce qui signifie qu'elles sont encore en vie. Quant au positionnement, leur première position est fixée de manière aléatoire pour les cellules normales ainsi qu'aux virus, et en ce qui concerne l'appellation, chaque cellule prend un caractère désignant la première lettre soit de son nom, soit de sa couleur. Les globules blancs sont représentés dans la matrice par la lettre W, les cellules normales par N, les cellules vides par un point, et pour finir, les virus par un V.

Lors de l'implémentation des cellules, nous avons initialisé notre monde cellulaire par des cellules vides, puis en appelant la fonction *ajouterCellule,* nous

remplaçons arbitrairement certains points par d'autres cellules selon le nombre *n* passé en paramètre (*code en figure ci-dessous*).

Cette fonction prend en paramètres: la cellule que nous souhaitons ajouté à la matrice, une liste où stocker ces cellules, et le nombre de cellules désiré. Elle est appelée au début de notre programme principale, pour initialiser notre matrice et commencer l'expérience de simulation.

```
public void ajouterCellule( Class<? extends Cellule> c, ArrayList<? super Cellule> cells, int n)
{
    Cellule cell = null;
    int x;
    int y;

    for( int i = 0; i < n ; i++)
    {
        do
        {
            x = ran.nextInt( this.ligne );
            y = ran.nextInt( this.colonne);

            if( !cellules[x][y].getClass().equals(c.getClass()) && cellules[x][y].toString().equals(".") )
            {

                break;
            }
        }
        while( true );

        try {
            cell = (Cellule)c.newInstance();
        } catch (InstantiationException | IllegalAccessException e) {
            e.printStackTrace();
        }

        if( cell != null )
        {
            cell.setx(x);
            cell.sety(y);

            cellules[x][y] = cell;
            cells.add( cell );
        }
    }
}
```

1. Les conditions de leurs déplacements

Comme nous l'avons évoqué précédemment, les cellules normales et vides ne se déplacent pas dans la matrice.Lors de l'exécution, la première impression que nous avons est que les virus circulent sans cesse, en revanche ce n'est pas le cas, c'est leur explosion qui nous fait prétendre cela, ils sont également des agents immobiles.

Contrairement aux autres cellules, les globules blancs ont la capacité de circuler de manière aléatoire dans les huit cases tout en respectant certaines conditions: la première interdise le fait de dépasser les extrémités du *monde*

cellulaire et la deuxième concerne le choix arbitraire des trois parmi les huits cases adjacentes.

Dans le cas où le positionnement de cet agent coïncide avec les extrémités de la matrice **(**la première\dernière ligne ou colonne**),** celui-ci se déplace suivant les cases qui les entourent et qui doivent obligatoirement appartenir à notre monde cellulaire, ce qui est implémenté avec la fonction *deplacer* (*figure suivante*).

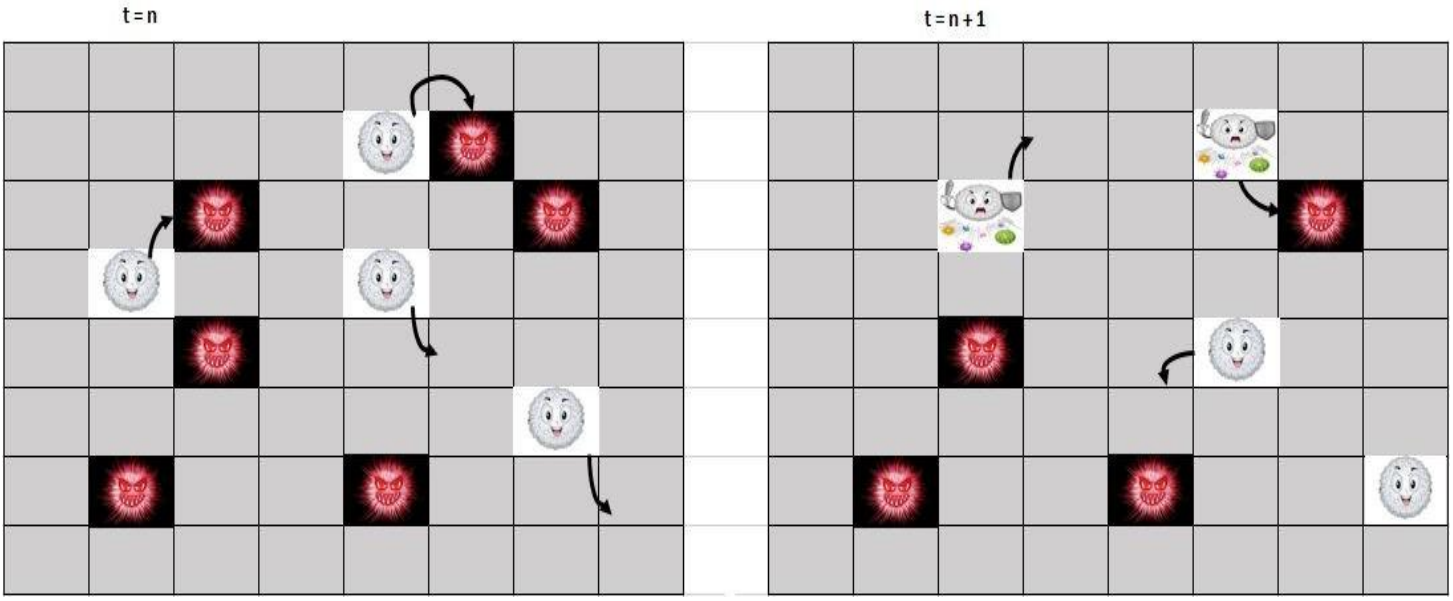
```
public int[] deplacer()
{
    int[] prochaine_cas = new int[2];
    int r = ran.nextInt( 16 );
    int[] tab_voisins = this.affichevoisins(); //retourne tous les voisins de cette case

    if( (r%2) == 0 )
    {
        prochaine_cas[0] = tab_voisins[r];
        prochaine_cas[1] = tab_voisins[r+1];
    }
    else
    {
        prochaine_cas[0] = tab_voisins[r-1];
        prochaine_cas[1] = tab_voisins[r];
    }

    return prochaine_cas;
}
```

2. Le rôle des globules blancs

Les globules blancs correspondent à des cellules du système immunitaire qui jouent un rôle important dans la défense de l'organisme *Monde cellulaire* contre les agents étrangers. Lors d’une infection, ils se dirigent alors où ils doivent aller, se multiplient puis détruisent et occupent la position du virus trouvé.



Pour implémenter le rôle des globules blancs, nous avons utilisé la fonction *division* qui a pour but de diviser ces leucocytes en deux dans le cas où le nombre de virus dépasse les deux tiers du total des cellules.

```
public int[] division()
{
    int[] prochaine_cas = new int[2];
    int r = ran.nextInt( 16 );
    int[] tab_voisins = this.affichevoisins();


    if( (r%2) == 0 )
    {
        prochaine_cas[0] = tab_voisins[r];
        prochaine_cas[1] = tab_voisins[r+1];
    }
    else
    {
        prochaine_cas[0] = tab_voisins[r-1];
        prochaine_cas[1] = tab_voisins[r];
    }

    return prochaine_cas;
}
```

C. Analyse du comportement du SMA

Nous avons étudié le comportement des globules blancs sur les autres cellules sanguines ainsi qu'aux virus qui les contaminent.

D’après ce qui était abordé précédemment, si le nombre de virus dépassent les deux tiers de la totalité des cellules, les globules blancs commencent à se diviser tout en éliminant les germes qui leur sont voisins.

```
Nombre de Globule Blanc est :   5
Nombre de Virus est :           15
Nombre de Cellule Normale est : 50
t = 0
************************************

          0     1     2     3     4     5     6     7     8     9
    0     .     N     N     V     N     N     N     N     N     N
    1     .     .     .     N     .     .     .     N     .     N
    2     N     N     .     N     V     N     .     .     .     N
    3     V     N     N     N     N     .     V     .     .     N
    4     N     N     V     .     N     .     W     N     V     N
    5     .     .     .     N     V     N     N     N     N     N
    6     V     .     .     N     .     W     V     .     N     .
    7     W     N     N     N     N     N     V     .     N     W
    8     N     N     N     .     N     .     N     V     V     N
    9     V     W     N     N     .     V     .     V     N     N
```

```
Nombre de Globule Blanc est :   5
Nombre de Virus est :           44
Nombre de Cellule Normale est : 3
t = 88
************************************

          0     1     2     3     4     5     6     7     8     9
    0     W     .     V     .     .     .     V     .     .     .
    1     .     .     .     .     V     V     .     V     .     V
    2     W     .     .     V     V     V     V     V     V     .
    3     .     .     V     .     .     V     .     V     V     .
    4     .     V     .     V     V     V     .     .     .     V
    5     .     V     .     V     V     V     V     V     V     V
    6     .     V     .     .     V     .     V     V     .     .
    7     .     .     V     V     V     V     .     .     .     V
    8     W     .     N     .     .     V     .     V     .     .
    9     .     W     N     V     V     V     V     .     W     N
```

```
Nombre de Globule Blanc est :   5
Nombre de Virus est :           35
Nombre de Cellule Normale est : 1
t = 141
**********************************

          0   1   2   3   4   5   6   7   8   9
    0     W   .   .   V   .   V   .   V   V   .
    1     W   .   .   V   .   V   .   V   .   V
    2     .   V   .   V   V   V   V   .   V   V
    3     .   .   .   V   V   .   .   W   V   V
    4     .   W   V   .   .   .   .   .   V   .
    5     .   .   .   .   V   .   V   .   V   .
    6     .   .   .   V   .   .   .   V   V   .
    7     .   .   V   .   W   .   .   .   V   .
    8     .   .   V   V   .   .   .   .   .   .
    9     .   .   N   V   .   V   V   V   .   .

DEAD
```

i. L’effet de la durée de l’explosion du virus sur les cellules

D’après les tests effectués, nous remarquons que lorsque le nombre de cellules normales tend vers zéro, l'humain meurt. nous avons également constaté que plus l’interval du temps d’explosion est élevé, plus la simulation prendra du temps pour que la personne guérisse ou meurt
Dans le cas évoqué juste avant l’individu a plus de chance de se rétablir

ii. L’effet du nombre de virus sur l’issue de la simulation

Nous avons commencé nos tests par un nombre de germes égale à cinquante, mais cela ne donnait pas une bonne simulation à cause de la quantité des globules blancs qui augmentait rapidement, ce qui nous a poussé à réduire le total des virus au deux tiers du global des cellules pour que ça soit plus visible (pour pouvoir examiner son état de mort ou de guérison).

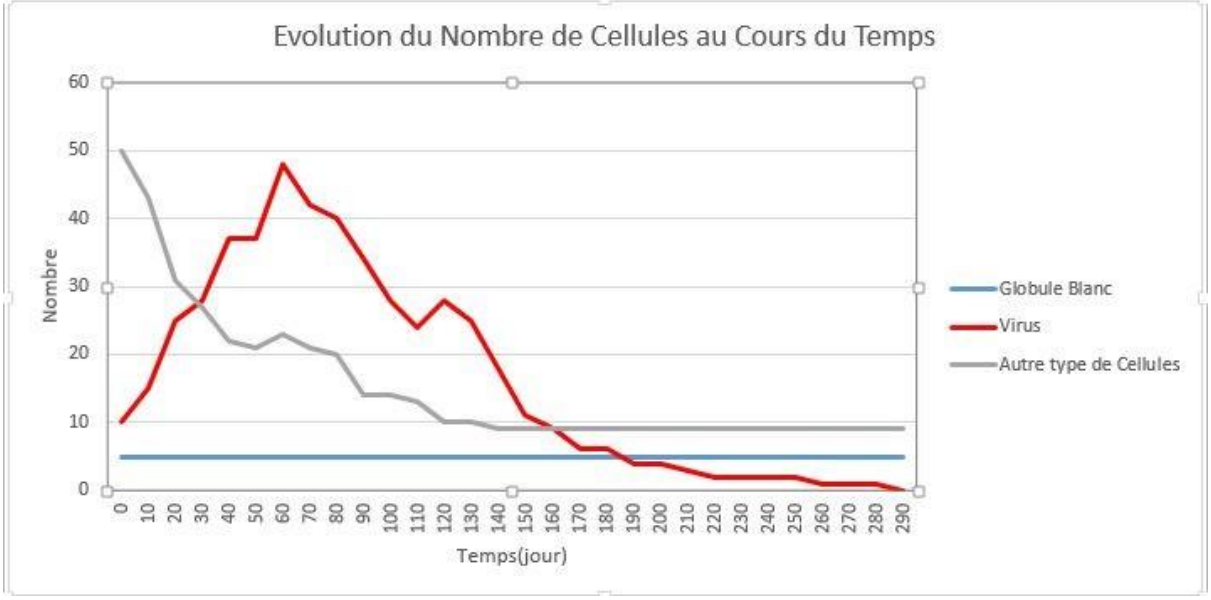
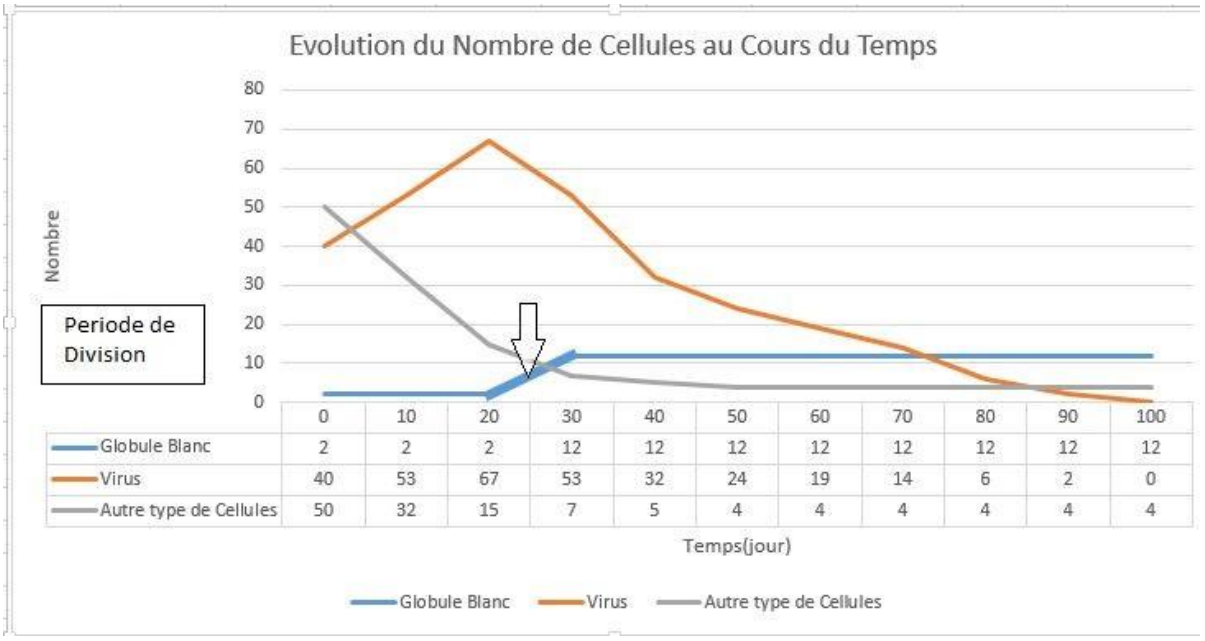


Figure : Exemple de division des globules blancs.



	0	10	20	30	40	50	60	70	80	90	100
Globule Blanc	2	2	2	12	12	12	12	12	12	12	12
Virus	40	53	67	53	32	24	19	14	6	2	0
Autre type de Cellules	50	32	15	7	5	4	4	4	4	4	4

Figure : Période de division des globules blancs.

Nous pouvons observer d'après la figure ci-dessus, que la quantité des globules blancs a connu une augmentation remarquable au niveau de l'intervalle [20, 30]; la période où ce nombre a été multiplié trois fois. Ces résultats sont dûs au

principe de la division des leucocytes, qui se multiplient une fois que le nombre des virus atteint 60 ou le dépasse, ce qui est égale à ⅔ du nombre totale des cellules.

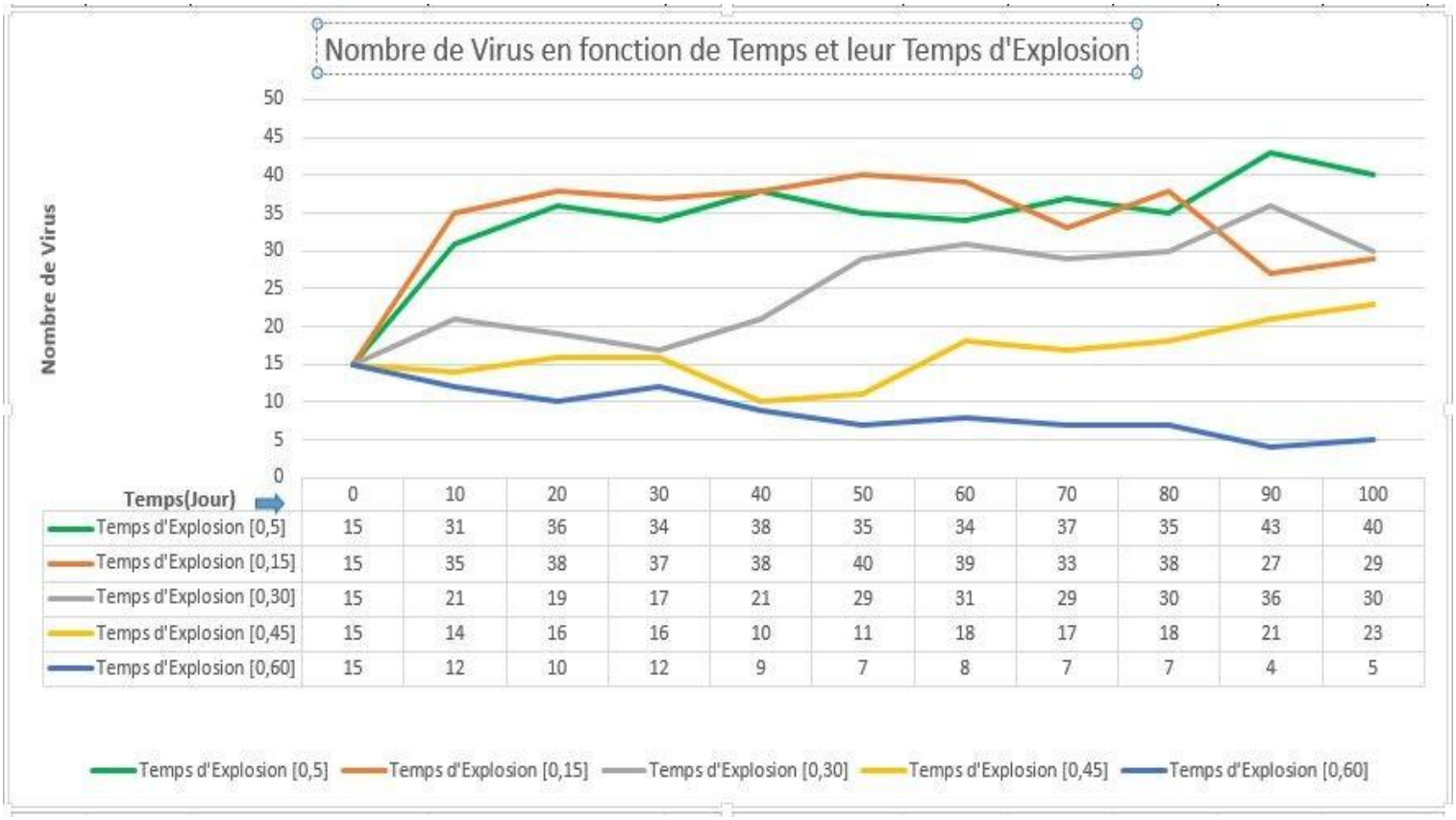


Figure: L'effet de l'intervalle du temps d'explosion sur l'évolution des virus en fonction du temps.

le graphe ci-dessus illustre l'évolution des virus en fonction de leur temps d'explosion avec un nombre fixe de globules blancs égale à cinq tout au long de l'expérience. Nous remarquons que plus l'intervalle de la durée de déflagration est petit plus le nombre des germes augmente, le contraire est également vrai.

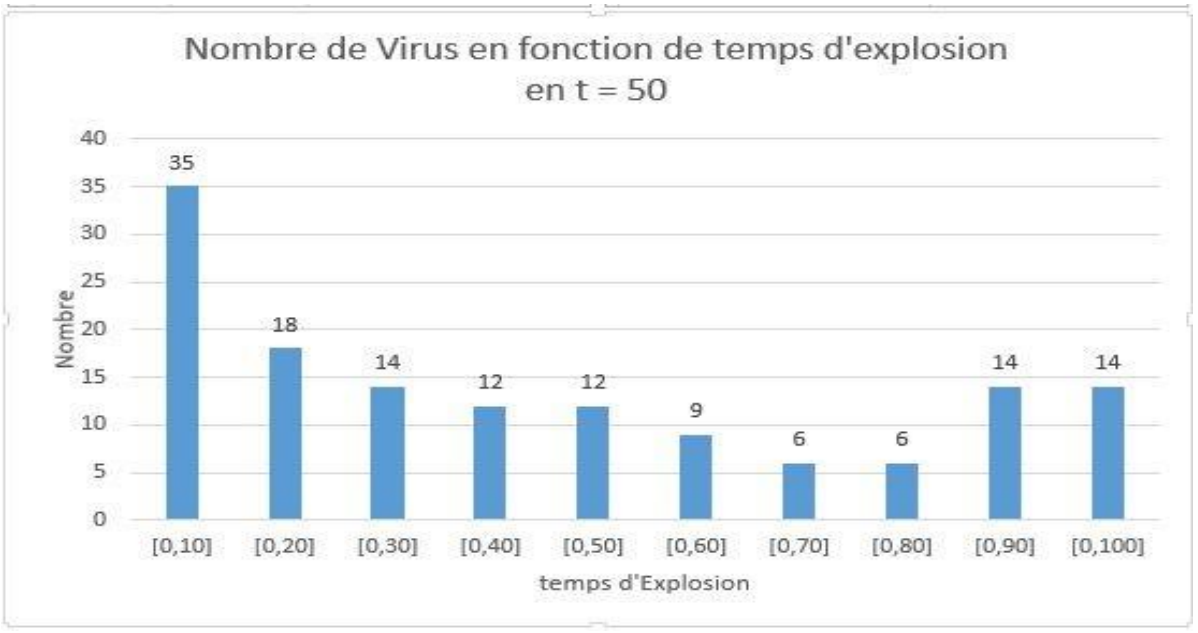


Figure: Augmentation des virus par rapport au temps d'explosion

Conclusion

Les virus que nous contactons quotidiennement et la façon avec laquelle notre organisme se défend, nous a inspiré pour l'implémentation de ce Système Multi Agent.

D'une part, cette étude nous a donné la possibilité de se cultiver et mieux apprendre sur le corps humain, voire le monde cellulaire.

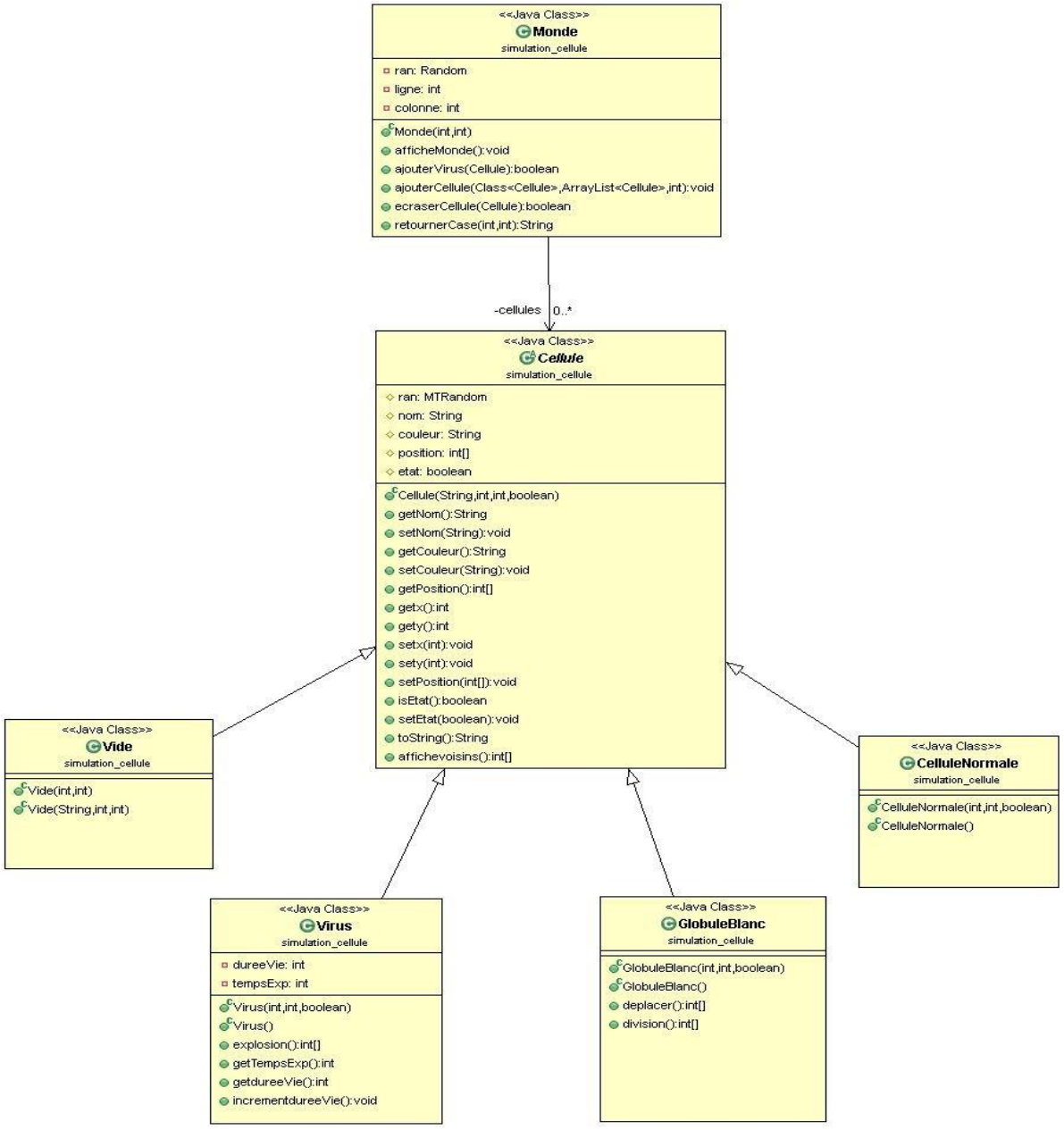
D'autre part, la réalisation de ce projet en JAVA nous a aidé à mieux pratiquer les différents concepts de ce langage, assimiler les notions abordées en cours comme le choix convenable du générateur, et même penser à développer une application traitant le sujet de notre projet.

Les conditions imposées au début de ce projet, nous ont résulté une bonne simulation, surtout que le corps humain guérisse après un nombre d'itérations moyen.

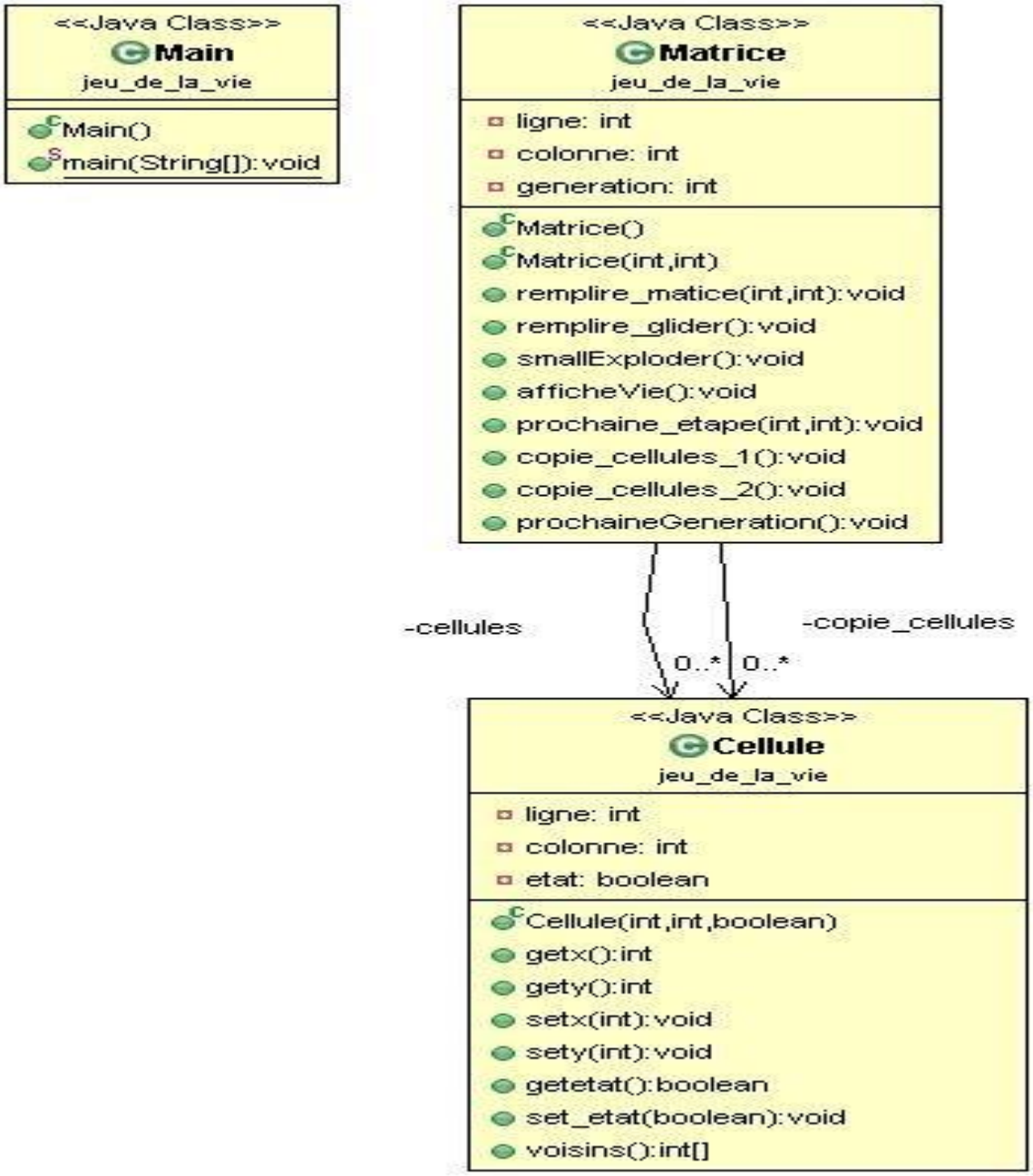
Répartition des tâches:

BENLOULOU Sarra s'est occupée des codes, celui du jeu de la vie ainsi que celui du SMA, tout en commentant les différentes fonctions utilisées. Tandis que NEKKAA Yousra et AKHTEN Fatima-Ezzohra, se sont chargées de la rédaction du rapport, en essayant d'expliquer les fonctionnalités du SMA. Nous avons également donné un petit coup de main à Sarra, en ce qui concerne l'implémentation de la classe *CelluleNormale,* et nous avons toutes contribuées à la création et la génération de l'UML.

Annexe

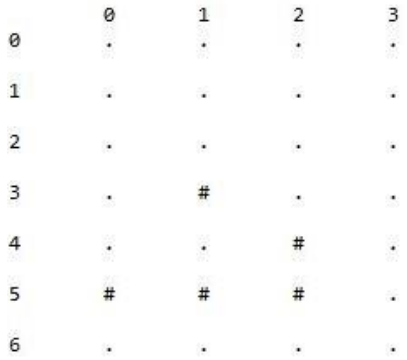

<<Java Class>>
Monde
simulation_cellule
ran: Random
ligne: int
colonne: int
Monde(int,int)
afficheMonde():void
ajouterVirus(Cellule):boolean
ajouterCellule(Class<Cellule>,ArrayList<Cellule>,int):void
ecraserCellule(Cellule):boolean
retournerCase(int,int):String
-cellules
0..*
<<Java Class>>
Cellule
simulation_cellule
ran: MTRandom
nom: String
couleur: String
position: int[]
etat: boolean
Cellule(String,int,int,boolean)
getNom():String
setNom(String):void
getCouleur():String
setCouleur(String):void
getPosition():int[]
getx():int
gety():int
setx(int):void
sety(int):void
setPosition(int[]):void
isEtat():boolean
setEtat(boolean):void
toString():String
affichevoisins():int[]
<<Java Class>>
Vide
simulation_cellule
Vide(int,int)
Vide(String,int,int)
<<Java Class>>
CelluleNormale
simulation_cellule
CelluleNormale(int,int,boolean)
CelluleNormale()
<<Java Class>>
Virus
simulation_cellule
dureeVie: int
tempsExp: int
Virus(int,int,boolean)
Virus()
explosion():int[]
getTempsExp():int
getdureeVie():int
incrementdureeVie():void
<<Java Class>>
GlobuleBlanc
simulation_cellule
GlobuleBlanc(int,int,boolean)
GlobuleBlanc()
deplacer():int[]
division():int[]

JEU DE LA VIE

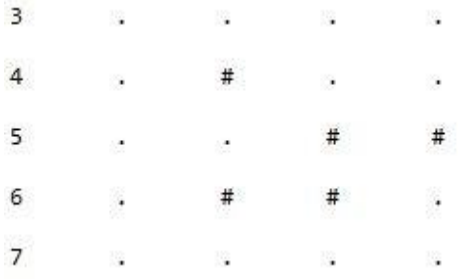

<<Java Class>>
Main
jeu_de_la_vie
Main()
main(String[]):void
<<Java Class>>
Matrice
jeu_de_la_vie
ligne: int
colonne: int
generation: int
Matrice()
Matrice(int,int)
remplire_matice(int,int):void
remplire_glider():void
smallExploder():void
afficheVie():void
prochaine_etape(int,int):void
copie_cellules_1():void
copie_cellules_2():void
prochaineGeneration():void
-cellules
-copie_cellules
0..*
0..*
<<Java Class>>
Cellule
jeu_de_la_vie
ligne: int
colonne: int
etat: boolean
Cellule(int,int,boolean)
getx():int
gety():int
setx(int):void
sety(int):void
getetat():boolean
set_etat(boolean):void
voisins():int[]

**Implémentation de GLIDER :

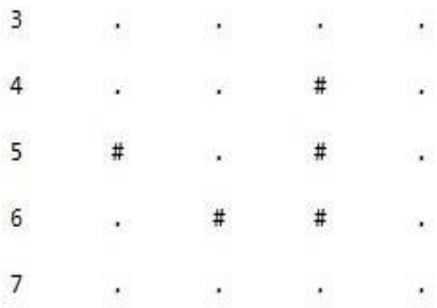
t = 0



t = 1



t = 2



t = 3