Mini-projet LU2IN002 - 2020-2021

Nom : Ramdani	*Nom :* Elquez
Prénom : Cyrena	*Prénom : Miguel*
N° étudiant : 3805942	*N° étudiant :* 28600492

Thème de simulation choisi (en 2 lignes max.)
Le thème, inspiré de « Il était une fois la vie » correspond à notre organisme. Les globines acheminent l'O2 au Cœur pour se reproduire pendant que les pathogènes transforme l'O2 se nourrissent d'O2 .

Description des classes et de leur rôle dans la simulation (2 lignes max par classe)

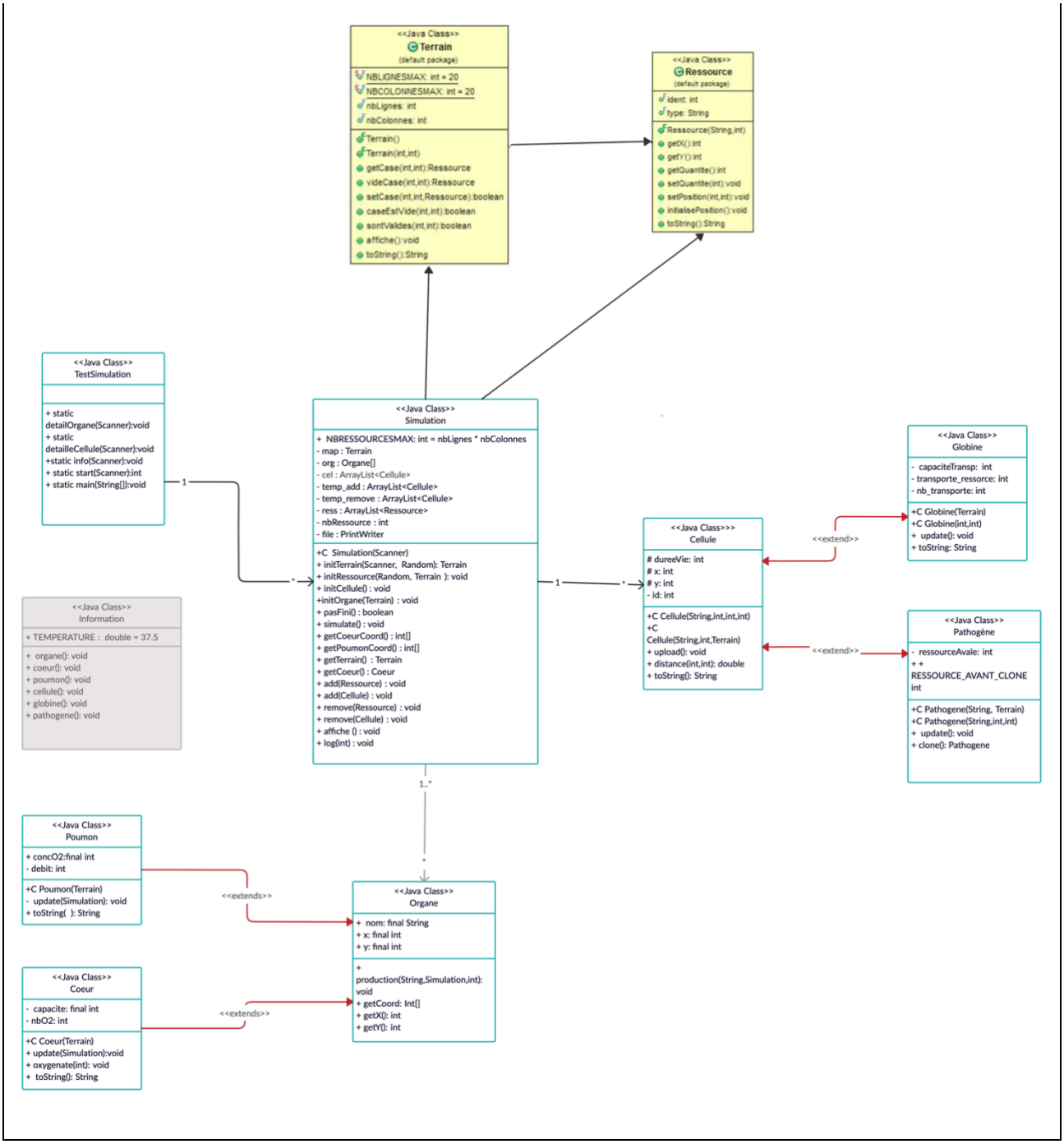
- **TestSimulation** : Permet d'avoir des informations la simulation et ses principaux acteurs, commencer la simulation ou quitter.
- **Simulation** : Initialise le Terrain les Organes, les Cellules et démarre la simulation. S'arrête si le CO2 surpasse l'O2, s'il n'y a plus de globines ou si on dépasse 1000 tours.
- **Information** : Donne les informations sur le rôle de chaque élément de la simulation. Contient la constante de la température du corps humain.
- **Cellule** : Ce sont des Agents, ils ont des coordonnés une durée de vie et un type. Il y a des Cellules bénéfiques (Globine) et nocives (Pathogène).
- **Globine** : Les Globines transportent l'O2 jusqu'au Cœur, le CO2 aux Poumons et se déplacent aléatoirement quand elles ne transportent rien. Avec une capacité de transport compris entre 1 et 5.
- **Pathogène** : Les Pathogènes se nourrissent d'O2 pour se cloner et produisent du CO2. Ils se déplacent de manière aléatoire.
- **Organe** : N'est pas un agent mais permet la production d'O2 et de CO2.
- **Poumon** : Produit de l'O2 avec un certain débit et permet aux globines de se débarrasser de leur CO2 transporté.
- **Cœur** : Produit des globines une fois suffisamment d'O2 consommé et rejette du CO2.

Décrire, dans les grandes lignes, ce qui se passe durant la simulation (max. 5-6 lignes)

Les organes sont placés de manière fixe sur le Terrain. On place de façon aléatoire les Ressources et Cellules de façon à avoir des simulations intéressantes. On remarque que si on commence avec beaucoup d'O2 au départ les Pathogènes et le CO2 augmente de manière exponentielle. A la fin de la simulation crée un fichier Simulation.log et affiche une modélisation faite sur gnuplot dans le terminal.

Schéma UML fournisseur des classes (dessin "à la main" scanné ou photo acceptés)

- Les flèches noires : Composition
- Les flèches rouges : Hérédité
- Classes en gris : Statique



Checklist des contraintes prises en compte:	*Nom(s) des classe(s) correspondante(s)*
Classe contenant un tableau ou une liste d'objets	- Simulation
Classe statique contenant que des méthodes statiques	- Information

Héritage	- Cellule:
	  • Globine
	  • Pathogène
	- Organe
	  • Coeur
	  • Poumon

Classe avec composition	- Simulation
Classe avec un constructeur par copie ou clone()	- Pathogene
Noms des classes créées (entre 4 et 10 classes)	- TestSimulation - Simulation - Organes - Coeur - Poumon - Cellules - Globine - Pathogene - Temperature

Copier / coller de vos classes à partir d'ici :

```
class Information{
   private static final double  TEMPERATURE = 37.2;

   private Information(){}

   public static void simulation(){
      System.out.println("Bienvenue dans la simulation \"Il etait une fois... la Vie\"! ");
      System.out.println("Inspiré de la serie televise \"La fabuleuse histoire du corps humain\".");
      System.out.println("Vous allez pouvoir apprecier les interactions entre les divers organes du corps humains avec leurs resources.");
      System.out.println("Le tout sans oublier les organismes externes qui nous pourrissent bien la vie (cf année 2020~).");
      System.out.println("\nCommencer la Simulation va proceder par plusieurs etapes :");
      System.out.println("1. Creer un terrain, avec un choix de la taille");
      System.out.println("2. Pose aleatoire des ressources");
      System.out.println("3. Creation d'Organes (Coeur, Poumons) et Cellules (Globines, Pathogenes)");
      System.out.println("4. La simulation commence et on peut suivre le mouvement des Cellules grace aux logs.");
      System.out.println("5. La situation se termine uniquement si le nombre de tour atteint le max ou si le nombre de CO2 excede celui de l'O2\n");
   }

   public static void organe(){
      System.out.println("\nLes Organes composent le corps. On s'interesse ici uniquement au Coeur et au Poumon.\n");

   }
   public static void coeur(){
      System.out.println("Le Coeur assure la circulation sanguine. Il recoit de l'O2 grace aux globines.");
      System.out.println("Une fois un certain nombre d'O2 recut il produit de nouvelles globines et du CO2.\n");
   }

   public static void poumon(){
      System.out.println("Le Poumon assure l'approvisionnement d'O2 dans le corps.");
      System.out.println("Il produit de l'O2 avec un certain debit et permet aux globines de les debarasser du CO2.\n");
   }

   public static void cellule(){
      System.out.println("Les cellules sont des micro-organismes qui assure se déplace dans le corps humain. Ils peuvent etre benefique pour le corps.\n");
   }

   public static void globine(){
      System.out.println("Les globines sont des agents qui assurent l'acheminement des ressources entre les divers organes du corps humain.\n");

   }

   public static void pathogene(){
      System.out.println("Les pathogenes sont des agents qui se nourissent des ressources et se multiplient. Ils transforment l'O2 en CO2\n");
   }
}
```

```
import java.util.*; //need Random and Scanner

class TestSimulation{

   private TestSimulation(){}

   public static void detailOrgane(Scanner sc){

      Information.organe();
      System.out.printf("Which Organe information do you want?\n");
      System.out.printf("\t1. Coeur\n");
      System.out.printf("\t2. Poumon\n");
      System.out.printf("\t3. Quit\n");
      //try/catch affiche un message d'erreur si un int n'est pas rentré dans ligne de commande
      try{
         int choice = sc.nextInt();
         if (choice == 1)
            Information.coeur();
         else if (choice == 2)
            Information.poumon();
         else
            return;
      } catch(Exception e){
         throw new RuntimeException("Error.\nYou need to Enter a number");
      }
   }

   public static void detailCellule(Scanner sc){

      Information.cellule();
      System.out.printf("Which Cellule information do you want?\n");
      System.out.printf("\t1. Globine\n");
      System.out.printf("\t2. Pathogene\n");
      System.out.printf("\t3. Quit\n");
      //try/catch affiche un message d'erreur si un int n'est pas rentré dans ligne de commande
      try{
         int choice = sc.nextInt();
         if (choice == 1)
            Information.globine();
         else if (choice == 2)
            Information.pathogene();
         else
            return;
      } catch(Exception e){
         throw new RuntimeException("Error.\nYou need to Enter a number");
      }
   }
   public static void info(Scanner sc){
      System.out.printf("Which information do you want?\n");
      System.out.printf("\t1. All about the simulation\n");
      System.out.printf("\t2. Organes\n");
      System.out.printf("\t3. Cellules\n");
      System.out.printf("\t4. Quit\n");
      //try/catch affiche un message d'erreur si un int n'est pas rentré dans ligne de commande
      try{
         int choice = sc.nextInt();
         if (choice == 1)
            Information.simulation();
         else if (choice == 2)
            detailOrgane(sc);
         else if (choice == 3)
```

```
            detailCellule(sc);
         else
            return;
      } catch(Exception e){
         throw new RuntimeException("Error.\nYou need to Enter a number");
      }
   }

   //permet de choisir si on veut ou non commencer la simulation
   public static int start(Scanner sc){
      System.out.printf("Which simulation do you want?\n");
      System.out.printf("\t1. La vie ~\n");
      System.out.printf("\t2. Information\n");
      //mon projet initial mais les contraintes de production m'ont empeché de m'amuser
      // System.out.printf("\t2. Chunin Game\n");
      System.out.printf("\t3. Quit\n");
      //try/catch affiche un message d'erreur si un int n'est pas rentré dans ligne de commande
      try{
         int choice = sc.nextInt();
         if (choice == 2){
            info(sc);
            Start(sc);
         }
         return choice;
      } catch(Exception e){
         throw new RuntimeException("Error.\nYou need to Enter a number");
      }
   }

   public static void main(String args[]){
      Scanner sc = new Scanner(System.in);
      int jeu = start(sc);
      System.out.printf("Welcome!\n\n");

      if (jeu == 1 ) {
         Simulation s = new Simulation(sc);
         s.simulate();
      } else
         System.out.println("You chose to quit the Simulation.\nSee you soon!");
   }
}
```

```
import java.util.*; //need Random and Scanner and ArrayList
import java.io.*;

class Simulation{
   public final int NBRESSOURCESMAX; //nb max de ressources sur le terrain
   private Terrain map;
   private ArrayList<Cellule> cel;
   private Organe[] org;
   private ArrayList<Ressource> ress;
   private ArrayList<Cellule> temp_add;
   private ArrayList<Cellule> temp_remove;
   private int nbRessource;
   private PrintWriter file;

   //lance les methode d'initialisation de
   //  Terrain
   //  Agent=Cellule(Globine,Leucocyte,Pathogene)
   //  Organe = Coeur et Poumon
   public Simulation(Scanner sc){
```

```
    Random rnd = new Random();
    System.out.printf("You picked : La vie ~ ");
    map = initTerrain(sc, rnd);
    NBRESSOURCESMAX = map.nbColonnes * map.nbLignes <= 0 ? 1 : map.nbColonnes * map.nbLignes;
    initRessource(rnd, map); // position le coeur et les poumons
    initOrgane(map); // position le coeur et les poumons
    initCellule();

    map.affiche();
    System.out.printf(map.toString());
    System.out.println("\n" + org[0].toString() + "\n" + org[1].toString());

    try {
        file = new PrintWriter("simulation.log", "UTF-8");
    } catch(Exception e) {
        throw new RuntimeException("Error on log file");
    }

}

public Terrain initTerrain(Scanner sc, Random rnd){
    System.out.printf("Do you want to choose the field size ?\n");
    System.out.printf("\t1. Yes ~\n");
    System.out.printf("\t2. No - random \n");
    System.out.printf("\t3. No - max \n");

    // try-catch affiche un message d'erreur si un int n'est pas rentré dans ligne de commande
    try{
        int pickSize = sc.nextInt();

        System.out.printf("\nStarting field : \n");
        if (pickSize == 1)
            return new Terrain(sc.nextInt(), sc.nextInt());
        else if (pickSize == 2)
            return new Terrain( rnd.nextInt(Terrain.NBLIGNESMAX) ,rnd.nextInt(Terrain.NBCOLONNESMAX));
        else
            return new Terrain();
    } catch(Exception e){
        throw new RuntimeException("Error.\nYou need to Enter a number");
    }
}

public void initRessource(Random rnd, Terrain t){
    int ressourceStart = (int)((rnd.nextInt(NBRESSOURCESMAX) + 1)/2);
    ress = new ArrayList<Ressource>();
    boolean success;
    Ressource tmpR;
    int x, y;
    System.out.println("nb ressource start : " + ressourceStart);
    for (int i = 0; i < ressourceStart; i++){
        y = rnd.nextInt(t.nbLignes);
        x = rnd.nextInt(t.nbColonnes);
        tmpR =new Ressource("O2", rnd.nextInt(3) + 1);
        ress.add(tmpR);
        success = t.setCase(y, x, tmpR);
    }
}

private void initCellule() {
    cel = new ArrayList<Cellule>();
    cel.add(new Pathogene("Bactérie",map));
    for (int i=0;i<100;i++) {
        cel.add(new Globine(map));
```

```
    }
}

public void initOrgane(Terrain t){
    org = new Organe[2];
    org[0] = new Coeur(t);
    org[1] = new Poumon(t);
}

public boolean pasFini() {
    int o2 = 0;
    int co2 = 0;
    for (int i=0;i<map.nbLignes;i++) {
        for (int j=0;j<map.nbColonnes;j++) {
            if (map.getCase(i,j) != null) {
                if (map.getCase(i,j).type.equals("O2")) {
                    o2 += map.getCase(i,j).getQuantite();
                } else {
                    co2 += map.getCase(i,j).getQuantite();
                }
            }
        }
    }
    System.err.println("" + co2 + " " + o2);
    o2 = 0;
    co2 = 0;
    for (Ressource res: ress) {
        if (res.type.equals("O2")) {
            o2 += res.getQuantite();
        } else {
            co2 += res.getQuantite();
        }
    }
    System.err.println("" + co2 + " " + o2);
    if (co2 > o2) {
        return false;
    }
    boolean glob = false;
    for (Cellule c:cel) {
        if (c instanceof Globine) {
            glob = true;
            break;
        }
    }
    /*if (!glob) {
        return false;
    }*/
    return (cel.size() != 0);
}

public void simulate() {
    int n = 0;
    while (pasFini() && n < 1000) {
        temp_add = new ArrayList<Cellule>();
        temp_remove = new ArrayList<Cellule>();
        for (Cellule cellule :cel) {
            cellule.update(this);
        }
        for (Organe org_:org) {
            org_.update(this);
        }

        for (Cellule cellule:temp_remove) {
```

```
            cel.remove(cellule);
        }
        System.out.println(temp_add.size());
        for (Cellule cellule:temp_add) {
            cel.add(cellule);
        }
        //cel.addAll(temp_add);



        affiche();
        log(n);
        n++;
        System.out.println("\n\n");
    }
    file.close();
}

public int[] getCoeurCoord() {
    return org[0].getCoord();
}

public int[] getPoumonCoord() {
    return org[1].getCoord();
}

public Terrain getTerrain() {
    return map;
}

public Coeur getCoeur() {
    return (Coeur) org[0];
}

public void add(Ressource newRessource){
    if (map.caseEstVide(newRessource.getX(), newRessource.getY())){
        ress.add(newRessource);
        map.setCase(newRessource.getX(), newRessource.getY(), newRessource);
    } else {
        Ressource tmp = map.getCase(newRessource.getX(), newRessource.getY());
        if (newRessource.type.equals(tmp.type))
            tmp.setQuantite(tmp.getQuantite() + newRessource.getQuantite());
    }
}

public void add(Cellule newCell) {
    temp_add.add(newCell);
}

public void remove(Ressource oldRess){
    ress.remove(map.videCase(oldRess.getX(), oldRess.getY()));
}

public void remove(Cellule oldCel){
    temp_remove.add(oldCel);
}

public void affiche() {
    map.affiche();
    System.out.println(this.getCoeur());
    System.out.println(org[1]);
    for (Cellule cel_ : cel) {
        System.out.println(cel_);
```

```
        }
    }

    public void log(int n) {
        int co2 = 0;
        int o2 = 0;
        for (Ressource res: ress) {
            if (res.type.equals("O2")) {
                o2 += res.getQuantite();
            } else {
                co2 += res.getQuantite();
            }
        }
        int glob = 0;
        int path = 0;
        for (Cellule c:cel) {
            if (c instanceof Globine) {
                glob++;
            } else {
                path++;
            }
        }

        file.println("" + n + " " + o2 + " "+co2+" "+glob+" "+path);
    }
}
```

```
import java.util.*; //need Random and Scanner

abstract class Organe{
    public final String nom;
    public final int x;
    public final int y;

    public Organe(String nom, Terrain t){
        Random rnd = new Random();
        this.nom = nom;
        x = rnd.nextInt(t.nbColonnes);
        y = rnd.nextInt(t.nbLignes);
    }

    abstract public void update(Simulation sim);

    public String toString(){
        return nom + ": [" + y + ", " + x + "]";
    }

    public void production(String nom, Simulation sim, int quantite){
        Random rnd = new Random();
        Ressource newRes = new Ressource(nom, quantite);
        newRes.setPosition(rnd.nextInt(sim.getTerrain().nbLignes) , rnd.nextInt(sim.getTerrain().nbColonnes));
        sim.add(newRes);
    }

    public int getX(){
        return x;
    }

    public int getY(){
        return y;
    }
```

```
    public int[] getCoord() {
        int[] res = {x,y};
        return res;
    }
}
```

```
class Coeur extends Organe{
    public final int capacite;
    private int nbO2;

    public Coeur(Terrain t){
        super("Coeur", t);
        capacite = 10;
        nbO2 = 0;
    }

    public void update(Simulation sim){
        while (capacite <= nbO2){
            System.out.println("Production");
            super.production("CO2", sim, 5);
            sim.add(new Globine(super.x, super.y));
            nbO2 -= capacite;
        }
    }

    public void oxygenate(int nb) {
        System.out.println("        Coeur" + nb);
        nbO2 += nb;
    }

    public String toString() {
        return super.toString() + "(" + nbO2 +")";
    }

}
```

```
import java.util.*;

//Le poumon est un organe qui rejette le CO2
//et injecte de l'O2 dans l'organisme
class Poumon extends Organe{
    public static final double POURCENTAGE_MAP = 0.2;//poucentage de map a ajouter O2
    public final int concO2;//concentration O2 = quantite d'O2 ajouter par ressource
    private int debit;

    public Poumon(Terrain map){
        super("Poumon", map);
        debit = map.nbColonnes * map.nbLignes <= 0 ? 1 : (int)(map.nbColonnes * map.nbLignes *
POURCENTAGE_MAP);
        concO2 = 4;
    }


    public void update(Simulation sim){
        Random rnd = new Random();
        int ajoutO2 = rnd.nextInt(debit+1);//quantite O2 ajouté aleatoirement

        for (int i = 0; i < ajoutO2; i++){
            super.production("O2", sim, concO2);
        }
```

```
    }

    public String toString(){
        return super.toString() + " a un debit de " + debit + " O2/tour.";
    }
}
```

```
import java.util.*;

abstract class Cellule{
    protected final String type;
    private static int cpt=0;
    protected int dureeVie;
    protected int y;//ligne
    protected int x;//colonne
    private int id;

    public Cellule(String type, int dureeVie , int y, int x){
        this.type = type;
        this.dureeVie = dureeVie;
        this.y = y;
        this.x = x;
        id = cpt++;
    }



    public Cellule(String type, int dureeVie, Terrain t){
        this(type,dureeVie,(int)(Math.random()*t.nbLignes),(int)(Math.random() * t.nbColonnes));
    }



    public boolean estMort(){
        return dureeVie <= 0;
    }


    public double distance(int lig, int col){
        return Math.sqrt(Math.pow(lig - y, 2) + Math.pow(col - x, 2));
    }

    //xnew:nouvelle colonne
    //ynew:nouvelle ligne
    public void seDeplacer(int xnew, int ynew){
        //System.out.println(""+this+" "+xnew+","+ynew);
        x = xnew;
        y = ynew;

    }

    public void update(Simulation sim) {
        dureeVie--;
        if (estMort()) {
            sim.remove(this);
        }
    }

    public String toString() {
        return type + "("+id+") : [" + y + "," + x + "]";
    }
```

```
}
```

```
import java.util.*;

class Globine extends Cellule{
   private int capaciteTransp;
   private int transporte_ressource;
   private int nb_transporte;

   public Globine(Terrain t){
      super("Globule rouge", 20, t);
      Random rnd = new Random();
      capaciteTransp = rnd.nextInt(5) + 1;
   }

   public Globine(int x, int y) {
      super("Globule rouge",20,x,y);
   }


   public void update(Simulation sim) {
      super.update(sim);
      if (estMort()) return;


      if (transporte_ressource == 0) { //ne transporte pas de ressources, donc bouge aléatoirement
         int new_x = x + (int)(Math.random() * 3) - 1;
         int new_y = y + (int)(Math.random() * 3) - 1;
         while (!sim.getTerrain().sontValides(new_y,new_x)) {
            new_x = x + (int)(Math.random() * 3) - 1;
            new_y = y + (int)(Math.random() * 3) - 1;
         }
         super.seDeplacer(new_x,new_y); //FIXME: vérifier qu'on déborde pas

      } else {

         int[] coords;
         if (transporte_ressource == 1) { // O2
            coords =  sim.getCoeurCoord();
         } else  { //CO2
            coords = sim.getPoumonCoord();
         }
         int dx = x - coords[0];
         int dy = y - coords[1];
         int mv_x,mv_y;
         if (dx > 0)
            mv_x = -1;
         else if (dx < 0)
            mv_x = 1;
         else
            mv_x = 0;

         if (dy > 0)
            mv_y = -1;
         else if (dy < 0)
            mv_y = 1;
         else
            mv_y = 0;
         super.seDeplacer(x+mv_x, y+mv_y);

         if (x == coords[0] && y == coords[1]) { //
```

```
            if (transporte_ressource == 1) {
                sim.getCoeur().oxygenate(nb_transporte);

            }
            transporte_ressource = 0;
            nb_transporte = 0;
        }
    }

    Ressource res =  sim.getTerrain().getCase(y,x);
    if (res != null) {
        if (transporte_ressource == 0) {
            if (res.type == "O2") transporte_ressource = 1;
            if (res.type == "CO2") transporte_ressource = 2;
            if (nb_transporte + res.getQuantite() < capaciteTransp) {
                nb_transporte += res.getQuantite();
                sim.remove(res);
            }

        } else if (res.type == "O2" && transporte_ressource == 1
            || res.type == "CO2" && transporte_ressource == 2) {

            if (nb_transporte + res.getQuantite() <= capaciteTransp) {
                nb_transporte += res.getQuantite();
                sim.remove(res);
            } else {
                res.setQuantite(res.getQuantite() - (capaciteTransp - nb_transporte)); //La globine se remplit au max, et
laisse le reste sur place.
                nb_transporte = capaciteTransp;
            }
        }
    }
}

public String toString() {
    String s;
    if (transporte_ressource == 0) {
        s = "";
    } else if (transporte_ressource == 1) {
        s = "  O2 (" + nb_transporte+")";
    } else {
        s = "  CO2 (" + nb_transporte + ")";
    }
    return super.toString() + s;
}
}
```

```
class Pathogene extends Cellule{
    private int ressourceAvale;
    public static final int RESSOURCE_AVANT_CLONE = 15;

    public Pathogene(String type, Terrain t){
        super(type, 20, t);
        ressourceAvale = 0;
    }

    public Pathogene(String type, int x, int y) {
        super(type,20,x,y);
    }

    public Pathogene clone(){
        return new Pathogene(super.type, this.x, this.y);
    }
```

```
    public void update(Simulation sim) {
        super.update(sim);
        if (estMort()) return;

        int new_x = x + (int)(Math.random() * 3) - 1;
        int new_y = y + (int)(Math.random() * 3) - 1;
        while (!sim.getTerrain().sontValides(new_y,new_x)) {
            new_x = x + (int)(Math.random() * 3) - 1;
            new_y = y + (int)(Math.random() * 3) - 1;
        }
        super.seDeplacer(new_x,new_y); //FIXME: vérifier qu'on déborde pas


        Ressource res =  sim.getTerrain().getCase(y,x);
        if (res != null && res.type == "O2") {
            if (res.getQuantite() == 0) {
                throw new RuntimeException("test");
            }
            Ressource tmp = new Ressource("CO2",res.getQuantite());

            tmp.setPosition(res.getY(),res.getX());
            sim.add(tmp);
            sim.remove(res);
            ressourceAvale += tmp.getQuantite();
            if (ressourceAvale >= RESSOURCE_AVANT_CLONE) {
                Pathogene new_pat = this.clone();
                new_pat.ressourceAvale /= 2;
                ressourceAvale = ressourceAvale - new_pat.ressourceAvale;
                sim.add(new_pat);
            }
        }
    }
}
```