Mini-projet LU2IN002 - 2020-2021

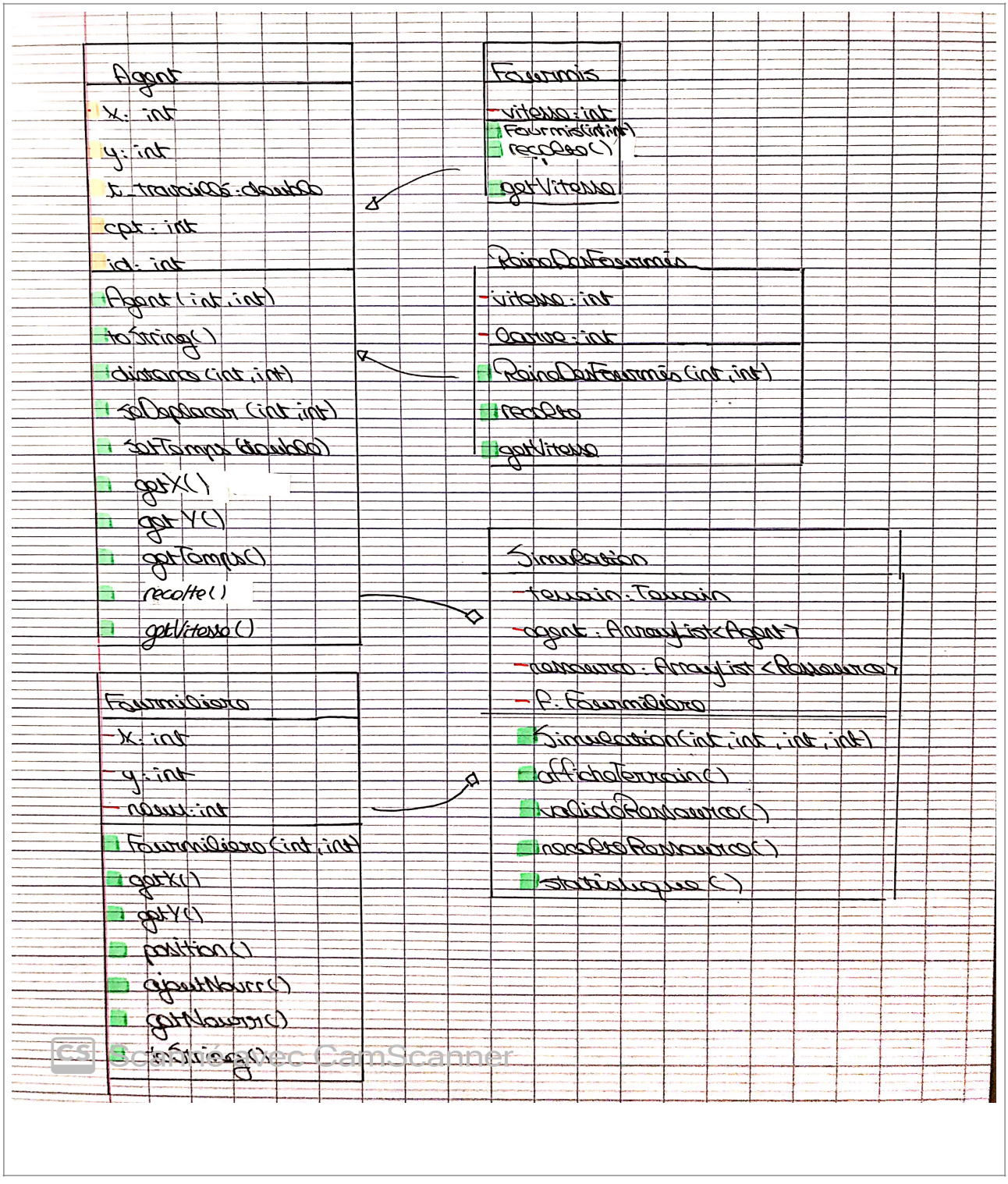
Nom : Zhang	*Nom :*
Prénom : Yilu	*Prénom :*
N° étudiant : 28601009	*N° étudiant :*

Thème de simulation choisi (en 2 lignes max.)
Récoltes de nourriture et production de larve par les fourmis d'une fourmilière..

Description des classes et de leur rôle dans la simulation (2 lignes max par classe)
Class Ressource: créer des ressource composé de id, type ("Nourriture") et quantité (aléatoire entre 0 et 3). Son rôle est de mettre des nourriture dans le terrain. Class Terrain: créer un terrain composé de nblignes et nb colonnes. Le rôle dans la simulation est de créer et d'afficher un terrain. Class Agent: créer des agents composé de id, nbcolonne et nbligne et le temps travallé, son rôle est de créer une liste d'agent qui fait la récolte et la production. Class Fourimis: class hérite de Agent, créer des fourmis en position (x,y) avec vitesse de travaille, faire la récolte dans la simulation. Class ReineDesFourmis: class hérite de Agent, créer un reine des fourmis en position (x, y) avec la vitesse d'accouchement, faire la production dans simulation. Class Fourmiliere: créer une fourmiere en position (x,y) et le nombre de nourriture dans cette fourmiliere, fait la partie de statistique. Class Simulation: faire une simulation avec un terrain, une liste de ressource, une liste d'agent, et une fourmiliere. L'endroit où on fait la simulation. Class TestTerrain: contient le main pour tester Terrain. Class TestSimulation: contient le main pour tester Simulation.

Décrire, dans les grandes lignes, ce qui se passe durant la simulation (max. 5-6 lignes)
Pendant la simulation, on a un terrain de taille x*y(10*10), les fourmis ont commencé de travailler et donc chaque fouemis sont sur le terrainet cherche une ressource plus proche que lui. Il l'emène dans la fourmilière et la reine des fourmis va avoir des larves nés grâce à ces nourriture. Chacun d'entre ces fourmis travaillent dans un temps limité (10min), puis rentrer pour reposer, jusqu'à quand tous les fourmis sont entrain de reposer ou qu'il n'y a plus de nourriture.

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



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

Héritage	Class Fourmis extendes Agent Class ReineDesFourmis extendes Agent
Classe avec composition	Class Simulation
Classe avec un constructeur par copie ou clone()	Class Fourmis extendes Agent Class ReineDesFourmis extendes
Noms des classes créées (entre 4 et 10 classes)	Class Ressource Class Terrain Class Agent Class Fourimis Class ReineDesFourmis Class Fourmiliere Class Simulation Class TestTerrain Class TestSimulation

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

```
public abstract class Agent {
    protected int x;
    protected int y;
    protected double t_travaille;
    protected static int cpt=0;
    protected final int id;

    public Agent(int x, int y) {
        cpt++;
        id=cpt;
        t_travaille=0.0;
        this.x=x;
        this.y=y;
    }

    public String toString() {
        return "L'agent "+id+" est en position ("+x+", "+y+")";
    }

    public double distance(int x, int y) { //la distance entre agent et (x,y)
        return (Math.sqrt((this.x-x)*(this.x-x)+(this.y-y)*(this.y-y)));
    }

    public void seDeplacer(int xnew, int ynew) { //Agent deplacer vers(xnew, ynew)
        this.x=xnew;
        this.y=xnew;
    }

    public void setTemps(double t) { //temps travaille de l'agent
        t_travaille+=t;
    }

    public int getX() {
        return x;
    }

    public int getY() {
        return y;
    }

    public double getTemps() {
        return t_travaille;
    }

    public abstract String recolte();
    public abstract int getVitesse();
}
```

```
public class Fourmis extends Agent {
    private static int vitesse=2;   //la vitesse de chaque fourmis/min.

    public Fourmis (int x, int y) {
        super(x, y);
    }

    public Fourmis (Fourmis fm) {
        super(fm.x, fm.y);
    }

    @Override
    public String recolte() {
        return "Le fourmis "+super.id+" a trouve de nourriture";
    }

    @Override
    public int getVitesse() {
        return vitesse;
    }
}


public class ReineDesFourmis extends Agent {
    private int vitesse; //le nombre d'enfant né quand il y a de nourriture
    private int larve;

    public ReineDesFourmis (int x, int y) {
        super(x, y);
        vitesse=(int)(Math.random()+1);
        larve=0;
    }

    public ReineDesFourmis (ReineDesFourmis rm) {
        super(rm.x, rm.y);
    }

    @Override
    public String recolte() {
        larve+=vitesse;
        return "La reine des fourmis a donner naissance à "+larve+" enfants.";
    }

    @Override
    public int getVitesse() {
        return vitesse;
    }
}


public class Fourmiliere {
```

```
    private int x, y, nourr;

    public Fourmiliere(int x, int y){
        this.x=x;
        this.y=y;
        nourr=0; //0 nourriture au debut
    }

    public int getX() {
        return x;
    }

    public int getY() {
        return y;
    }

    public String position() {
        return "La position de la fourmiliere et la reine est ("+x+", "+y+").";
    }

    public void ajoutNourr(int n) {
        nourr+=n;
    }

    public int getNourr() {
        return nourr;
    }

    public String toString() {
        return "Dans le fourmiliere, le quantite de nourriture recupere est "+nourr;
    }
}


import java.util.ArrayList;

public class Simulation {
    //Les composant de simulation
    private Terrain terrain;
    private ArrayList<Agent> agent;
    private ArrayList<Ressource> ressource;
    private Fourmiliere f;

    public Simulation (int m, int n, int x, int y) {
        //Initialisation de terrain
        terrain=new Terrain(x, y);

        //Initialisation du tableau de agent
        agent=new ArrayList<Agent>();
        for (int i=0; i<n-1; i++) { //Ajout des fourmis dans le tableau
            agent.add(new Fourmis(((int)(Math.random()*x)), ((int)(Math.random()*y))));
        }
```

```
    agent.add(new ReineDesFourmis(0, 0)); //Ajout de la reine des fourmis

    //Initialisation du tableau de ressource
    ressource=new ArrayList<Ressource>();
    for (int i=0; i<m; i++) { //Ajout des ressource dans la liste et le terrain
        Ressource r=new Ressource("Nourriture", ((int)(Math.random()*2+1)));
        r.setPosition((int)(Math.random()*x), (int)(Math.random()*y));
        ressource.add(r);
        terrain.setCase(r.getX(), r.getY(), r);
    }

    //Inicialisation de fourmiliere
    f=new Fourmiliere(0, 0);
}

public void afficheTerrain() { //Affichage de terrain avec des ressources
    terrain.affiche();
    System.out.println ("Informations sur le terrain:\n"+terrain);
}

public void valideRessource() { //Verification
    for (Ressource ress: ressource) {
        int rx=ress.getX();
        int ry=ress.getY();

        if (!terrain.sontValides(rx, ry))
            System.out.println("Incorrect: problème de coordonnées ("+rx+", "+ry+")!");
    }
}

public void recolteRessource() {
    System.out.println(f.position());

    for (Ressource ress: ressource) { //Pour chaque nourriture
        int rx=ress.getX();
        int ry=ress.getY();

        int p=0; //Creer une variable p: le plus proche avec le nourriture

        //Boucle pour tous les fourmis sauf la reine qui doit rester chez eux
        for (int i=0; i<(agent.size())-1; i++) {
            while (agent.get(i).getTemps()>10) { //Repos
                System.out.println(agent.get(i)+". Il travaille trop, il va rentre pour reposer.");
                agent.remove(i);
            }

            //Pour trouver le fourmis le plus proche
            if (agent!=null) {
                if (agent.get(p).distance(rx, ry)>agent.get(i).distance(rx, ry))
                    p=i;
                System.out.println(agent.get(p)+" est le plus proche que "+ress+".");
                double d=agent.get(p).distance(rx, ry); //Calcul de distance avec ress
```

```
                double temps1=d/agent.get(p).getVitesse(); //Calcul de temps pour arriver
                agent.get(p).seDeplacer(rx, ry); //Deplacer vers les ress
                System.out.println("Il prend "+temps1+"min pour arriver.");

                agent.get(p).setTemps(temps1); //Calcul du temps travalle

                terrain.videCase(rx,ry); //recuperer des ress donc vider la case
                System.out.println(agent.get(p).recolte());

                System.out.println("Il decide de le ramener dans le fourmiliere.");
                d=agent.get(p).distance(f.getX(), f.getY()); //Calcul de distance avec f
                double temps2=d/agent.get(p).getVitesse(); //Calcul de temps pour rentrer
                agent.get(p).seDeplacer(f.getX(), f.getY()); //Deplacer vers f
                System.out.println("Il prend "+temps2+"min pour rentrer.");

                agent.get(p).setTemps(temps2); //Calcul de temps total de travailler

                f.ajoutNourr(ress.getQuantite()); //Ajout de nourriture chez les fourmis
                System.out.println("Grace au nourriture, "+agent.get(agent.size()-1).recolte());

System.out.println("================================================");

System.out.println("================================================");
            }
          }
       }
    }


    public void statistique () {
       System.out.println(f.toString()); //Le nb total de ress recupere
       System.out.println(agent.get(agent.size()-1).recolte()); //Le nb total de larve ne
    }
}



public class TestSimulation {
    public static void main(String[] args) {
       Simulation simulation=new Simulation(10, 2, 10, 10);

       System.out.println("Il est temps de travailler pour les fourmis.");
       simulation.afficheTerrain();

System.out.println("================================================");
       simulation.valideRessource();
       simulation.recolteRessource();

       simulation.afficheTerrain();

       simulation.statistique();
    }
```

```
}
```