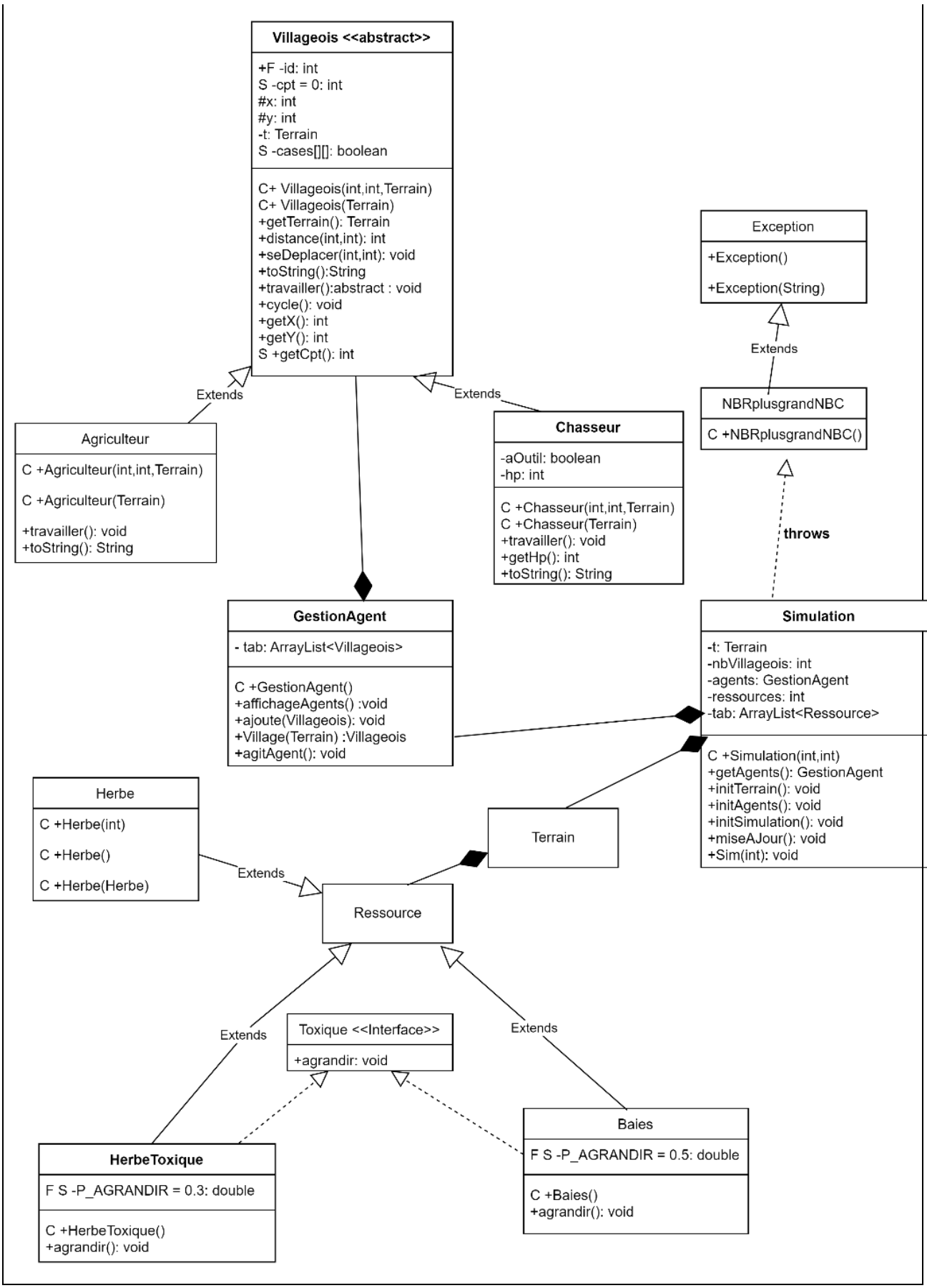
Projet LU2IN002 - 2022-2023

Numéro du groupe de TD/TME : groupe 7

Nom : Elhussieny	*Nom :* Kandil
Prénom : Habiba	*Prénom :* Omar
N° étudiant : 21105223	*N° étudiant :* 21107819

Thème choisi (en 2 lignes max.)
Dans ce projet, on a d'abord un village avec deux types de villageois : des agriculteurs et des chasseurs. Chacun a sa propre fonctionnalité sur les ressources. Ces derniers sont des herbes normales, des baies, ou des herbes toxiques.

Schéma UML des classes vision fournisseur (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 ArrayList	GestionAgent Simulation
Classe avec membres et méthodes statiques	Baies HerbeToxique Villageois
Classe abstraite et méthode abstraite	Villageois travailler()
Interface	Toxique
Classe avec un constructeur par copie ou clone()	Herbe
Définition de classe étendant Exception	NBRplusgrandNBC
Gestion des exceptions	Simulation TestSimulation

Présentation brève de votre projet (max. 10 lignes) : texte libre expliquant en quoi consiste votre projet.
Dans un village, il y a des villageois qui travaillent. Il y a aussi différentes ressources telle que des herbes normales, des herbes toxiques et des baies. Les ressources toxiques ont une caractéristique qu'ils peuvent s'agrandir tout seul. Tandis que pour les herbes, ce sont les agriculteurs qui doivent intervenir. Ces agriculteurs sont des villageois, ils doivent donc travailler. Ainsi, sur un terrain, si l'agriculteur est sur une ressource toxique alors il ne fait rien. Mais, si c'est une ressource normale, alors il augmente sa quantité de 1. De plus, si c'est une herbe dont la quantité a dépassée 4, elle devient une herbe toxique. En outre, un chasseur a deux caractéristiques : s'il possède ou pas un outil de chasse et sa santé. S'il se trouve sur une case ou la ressource n'est pas toxique : s'il possède un outil alors il mange la ressource et diminue sa quantité de 2. Sinon, il diminue sa quantité de 1. Cependant, si la ressource est toxique alors sa santé diminue de 1. Lorsque la santé devient inférieure à 0, le chasseur meurt. Un cycle c'est quand un villageois travaille pour 8 heures.

Copier / coller vos classes et interfaces à partir d'ici :

La classe Villageois :

```
public abstract class Villageois {
    private final int id;
    private static int cpt =0;
    protected int x;
    protected int y;
    private Terrain t;
    private static boolean cases[][];
    public Villageois(int x,int y, Terrain t){
        this.x = x;
```

```
        this.y =y;
        this.t = t;
        id = cpt;
        cpt++;
        cases = new boolean[t.nbLignes][t.nbColonnes];
        for(int i = 0; i < t.nbLignes ; i++ ){
            for(int j =0 ; j < t.nbColonnes ; j++){
                if((i != x ) && (j!= y)){
                    cases[i][j] = true;
                }
                else{
                    cases[i][j] = false;
                }
            }
        }
    }
    public Terrain getTerrain(){
        return t;
    }
    public Villageois(Terrain t){
        this((int)(Math.random()*t.nbLignes),(int)(Math.random()*t.nbColonnes),t);
    }
    public int distance(int i,int j){
        //cette fonction retourne la distanvce en cases non pas la distance normal
avec un double;
        double temp = Math.sqrt((i-x)*(i-x)+(j-y)*(j-y));
        return (int)(Math.ceil(temp));
    }
    public void seDeplacer(int xnew,int ynew ){
        if(cases[xnew][ynew]){
            cases[x][y] = true;
            x = xnew;
            y = ynew;
            cases[x][y] = false;
        }
        else{
            System.out.println(" On ne peut pas placer le villageois car la case
est occupee");
        }

    }
    public String toString(){
        return "Villageois " + id + " ";
    }
    //Une heure de travail
    public abstract void travailler();

    //un cycle a.k.a une journee de travail pour un villageois
   public void cycle(){
        //il travaille 8 heures
        for(int i =0 ; i <8; i++){
```

```
            this.seDeplacer((int)(Math.random()*t.nbLignes),(int)(Math.random()*t.n
bColonnes));
            this.travailler();
        }
        System.out.println(this.toString() + " a travailler pour 8h");
    }
    public int getX(){
        return x;
    }
    public int getY(){
        return y;
    }
    public static int getCpt(){
        return cpt;
    }

}
```

La classe Agriculteur :

```
public class Agriculteur extends Villageois{
    public Agriculteur(Terrain t){
        super(t);
    }
    public Agriculteur(int x,int y,Terrain t){
        super(x,y,t);
    }

    public void travailler(){
        //gets ressource in (x,y) in terrain
        Terrain t = getTerrain();
        Ressource r = t.getCase(x,y);
        if(r == null){
            r = new HerbeNormal();
            t.setCase(x, y, r);
        }
        //augmente quantite par 1 si r n'est pas toxique
        if(!(r instanceof Toxique)){
            r.setQuantite(r.getQuantite() + 1);
            //if herbe and quantite devient >= 4 donc devient toxique
            if(r.getQuantite() >= 4){
                t.videCase(x, y);
                r = new HerbeToxique();
                r.setPosition(x, y);
                t.setCase(x, y, r);
            }
        }
        if((r instanceof Toxique)){
            System.out.println("L'agriculteur ne peut pas agrandir un herbe
toxique");
        }
```

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

La classe Chasseur :

```
public class Chasseur extends Villageois{
    //possede un outil ou non
    private boolean aOutil;
    //signifie sa sante
    private int hp;
    public Chasseur(int x,int y, Terrain t){
        super(x,y,t);
        aOutil = Math.random() < 0.7;
        hp = 10;
    }
    public Chasseur(Terrain t){
        super(t);
        aOutil = Math.random() < 0.7;
        hp = 10;
    }

    public void travailler(){
        //gets ressource in (x,y) in terrain
        Terrain t = getTerrain();
        Ressource r = t.getCase(x,y);
        if(r != null){
            //checks if not toxic
            if(!(r instanceof Toxique)){
            //gets ressources more effeciently if a Outil

            if(aOutil){
                if(r.getQuantite() > 2){
                    r.setQuantite(r.getQuantite() -2);
                    System.out.println("il mange");
                }
                else{
                    t.videCase(x, y);
                    System.out.println("le chasseur enleve la ressource de la
case");
                }
            }
            //moins efficace (n'a pas outil)
            else{
                if(r.getQuantite() > 0){
                    r.setQuantite(r.getQuantite() -1);
```

```
                }
                else{
                    t.videCase(x, y);
                    System.out.println("le chasseur enleve la ressource de la
case");
                }
            }
        }
        //eats toxic
        if(r instanceof Toxique){
            //si c'est toxique, le fait qu'il possede un outil ou non n'affecte pas
sa chasse d'herbe toxique
            hp = hp -1;
            System.out.println(this.toString() +" a mange quelque chose
toxique...");
            if(r.getQuantite() > 0){
                r.setQuantite(r.getQuantite() -1);
            }
            else{
                t.videCase(x, y);
                System.out.println("le chasseur enleve la ressource toxique de la
case");
            }

        }
    }
    }
    public int getHp(){
        return hp;
    }
    public String toString(){
        return super.toString() + " chasseur";
    }
}
```

La classe Baies :

```
public class Baies extends Ressource  implements Toxique{
    //les baies apparaissent naturellement != (Herbe et HerbeToxique)
    private final static double P_AGRANDIR = 0.5;
    public Baies(){
        super("Baies",0);
    }

    //agrandir tout seul
    public void agrandir(){
        if(Math.random() > P_AGRANDIR){
            setQuantite(super.getQuantite()+ 1);
        }
    }
```

```
}
```

La classe HerbeNormal :

```
public class HerbeNormal extends Ressource{
    public HerbeNormal(int quantite){
        super("Herbe normal",quantite);
    }
    public HerbeNormal(){
        this(0);
    }
    public HerbeNormal(HerbeNormal h){
        this(h.getQuantite());
    }
}
```

La classe HerbeToxique :

```
public class HerbeToxique extends Ressource implements Toxique{
    private final static double P_AGRANDIR = 0.3;
    public HerbeToxique(){
        super("Herbe Toxique",5);
    }
    //agrandir tout seul
    public void agrandir(){
        if(Math.random() < P_AGRANDIR){
            setQuantite(getQuantite()+1);
        }
    }
}
```

La classe Toxique :

```
public interface Toxique{
    //agrandir tout seul
    public void agrandir();
}
```

La classe GestionAgents :

```
import java.util.ArrayList;
public class GestionAgent {
    private ArrayList<Villageois> tab;
    public GestionAgent(){
        tab = new ArrayList<Villageois>();
    }
    public void afficheAgents(){
        for(Villageois v : tab){
            System.out.println(v.toString());
        }
    }
    public void ajoute(Villageois v){
```

```
        tab.add((Villageois)v);
    }
    //creer un villageois aleatoire
    public Villageois Village(Terrain t){
        Villageois v;
        double k = Math.random();
        if(k < 0.5){
            v = new Chasseur(t);
        }
        else{
            v = new Agriculteur(t);
        }
        return v;
    }

    //les agents agissent et font un cycle de travail
    public void agitAgent(){
        for(int i= 0; i< tab.size(); i++){
            tab.get(i).cycle();
            if(tab.get(i) instanceof Chasseur){
                //if chasseur mange beaucoup de ressources toxiques il meurt
                int k = ((Chasseur)tab.get(i)).getHp();
                if(k < 0){
                    System.out.println((tab.get(i)).toString()+" meurt");
                    tab.remove(tab.get(i));
                }
            }
        }
    }
}
```

La classe NBRplusgrandNBC :

```
//NBR = nombre de ressources
//NBC = nombre de cases
public class NBRplusgrandNBC extends Exception{
    public NBRplusgrandNBC(){
        super("nombre de ressources plus grand que nombres de cases");
    }
}
```

La classe Simulation :

```
import java.util.ArrayList;
public class Simulation {
    private Terrain t;
    private int nbVillageois;
    private GestionAgent agents;
    private int ressources;
    private ArrayList<Ressource> tab;

    public Simulation(int nbVillageois, int ressources){
```

```
        this.nbVillageois = nbVillageois;
        this.ressources = ressources;
        t = new Terrain();
        agents = new GestionAgent();
        tab = new ArrayList<Ressource>();

    }
    public GestionAgent getAgents(){
        return agents;
    }
    //on initialise le terrain
    public void initTerrain() throws NBRplusgrandNBC{
        int n = ressources;
        if(n > t.nbLignes*t.nbColonnes){
            throw new NBRplusgrandNBC();
        }
        while( n > 0){
            int i = (int)(Math.random()*t.nbLignes);
            int j = (int)(Math.random()*t.nbColonnes);

            if(t.caseEstVide(i,j)){
                // 1% des ressources sont des baies
                if(Math.random() > 0.8){
                    t.setCase(i,j,(new Baies()));
                }
                else{
                    t.setCase(i,j,(new HerbeNormal()));
                }
                n--;
            }
        }
    }

    //initialisation des agents
    public void initAgents(){
        for(int i =0; i<nbVillageois;i++ ){
            agents.ajoute(agents.Village(t));
        }
    }
    public void initSimulation() throws NBRplusgrandNBC{
        initAgents();
        initTerrain();
    }

    //mise a jour du terrain
    public void miseAJour(){
        agents.agitAgent();
        tab = t.lesRessources();
        for(Ressource r : tab){
            if(r instanceof Toxique){
                ((Toxique)r).agrandir();
```

```
            }
        }
        System.out.println(t.toString());
    }
    public void Sim(int x){
        for(int i = 0; i<x; i++){
            System.out.println("Cycle:"+ i);
            miseAJour();
        }
    }
}
```

La classe TestSimulation :

```
public class TestSimulation {
    public static void main(String []args){
        Simulation s = new Simulation(6,57);
        try{ s.initSimulation();
        } catch (NBRplusgrandNBC erreur){
            System.out.println(erreur.toString());
        }

        (s.getAgents()).afficheAgents();
        s.Sim(8);
        (s.getAgents()).afficheAgents();
    }
}
```