Mini-projet LU2IN002 - 2020-2021

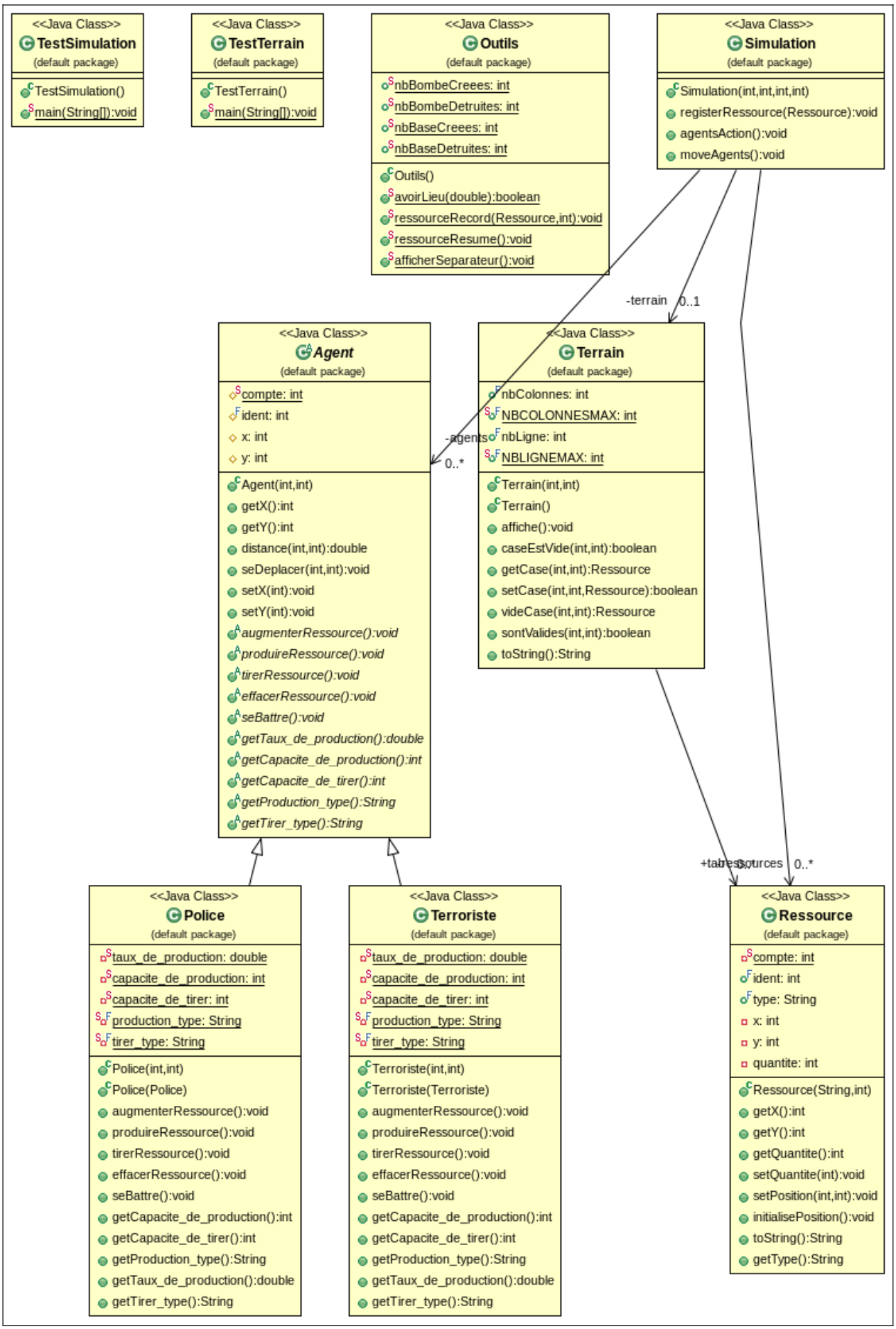
Nom : LI	*Nom :* ZENG
Prénom : Junji	*Prénom : Fanxiang*
N° étudiant : 28610187	*N° étudiant :* 28600693

Thème de simulation choisi (en 2 lignes max.)
Une bataille entre les polices et les terroriste sur un champ de bataille (le terrain)

Description des classes et de leur rôle dans la simulation (2 lignes max par classe)
Class Terrain: crée un tableau a deux dimension, qui sera affiché, avec des nom de ressources à l'intérieur. Ce dernier contient des méthodes pour vider ou mettre des ressource dans une case Class Ressource: crée des ressources, avec le constructeur qui donne son type (bombe ou base) et sa quantité. Sa position sera déterminer une méthode setPosition() Class Agent: class mère des classes Terroriste et Police, avec comme attribut sa position et sa morale class Terroriste: class fille de la class Agent, qui lui a son production type (bombe), et son tirer type (base), avec des autres paramètre (capacité de production et de tirer; taux de production) class Police: class fille de la class Agent, qui lui a son production type (base), et son tirer type (bombe), avec des autres paramètre (capacité de production et de tirer; taux de production) class Simulation: Simule le bataille entre police et terroriste qui construisent des bases et des bombes et les detruits sur un terrain class TestSimulation: la class qui contient le main qui teste la simulation class TestTerrain: la classs qui contient le main qui teste de notre terrain classe Outil : une class qui ne contient que des méthodes statique, qui fait l'affichage des ressources et fait le statistique

Décrire, dans les grandes lignes, ce qui se passe durant la simulation (max. 5-6 lignes)
Durant la simulation, les polices et les terroristes vont construire et enlever les type de ressource qui leur correspondent. Les terroristes vont contruire des bombes et détuire des bases, et les polices construire des bases et désactiver des bombes. Si un terroriste rencontre une case vide ou bien une bombe, il va construire une bombe sur le terrain, avec un pourcentage correspondant aux taux de production. Et si il renconcontre une base il va la détruire. Et c'est la même chose pour les polices qui lui fait le contraire.

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


<<Java Class>>
TestSimulation
(default package)
TestSimulation()
main(String[]):void
<<Java Class>>
TestTerrain
(default package)
TestTerrain()
main(String[]):void
<<Java Class>>
Outils
(default package)
nbBombeCreees: int
nbBombeDetruites: int
nbBaseCreees: int
nbBaseDetruites: int
Outils()
avoirLieu(double):boolean
ressourceRecord(Ressource,int):void
ressourceResume():void
afficherSeparateur():void
<<Java Class>>
Simulation
(default package)
Simulation(int,int,int,int)
registerRessource(Ressource):void
agentsAction():void
moveAgents():void
-terrain
0..1
-agents
0..*
<<Java Class>>
Agent
(default package)
compte: int
ident: int
x: int
y: int
Agent(int,int)
getX():int
getY():int
distance(int,int):double
seDeplacer(int,int):void
setX(int):void
setY(int):void
augmenterRessource():void
produireRessource():void
tirerRessource():void
effacerRessource():void
seBattre():void
getTaux_de_production():double
getCapacite_de_production():int
getCapacite_de_tirer():int
getProduction_type():String
getTirer_type():String
<<Java Class>>
Terrain
(default package)
nbColonnes: int
NBCOLONNESMAX: int
nbLigne: int
NBLIGNEMAX: int
Terrain(int,int)
Terrain()
affiche():void
caseEstVide(int,int):boolean
getCase(int,int):Ressource
setCase(int,int,Ressource):boolean
videCase(int,int):Ressource
sontValides(int,int):boolean
toString():String
+tabRessources
0..*
<<Java Class>>
Police
(default package)
taux_de_production: double
capacite_de_production: int
capacite_de_tirer: int
production_type: String
tirer_type: String
Police(int,int)
Police(Police)
augmenterRessource():void
produireRessource():void
tirerRessource():void
effacerRessource():void
seBattre():void
getCapacite_de_production():int
getCapacite_de_tirer():int
getProduction_type():String
getTaux_de_production():double
getTirer_type():String
<<Java Class>>
Terroriste
(default package)
taux_de_production: double
capacite_de_production: int
capacite_de_tirer: int
production_type: String
tirer_type: String
Terroriste(int,int)
Terroriste(Terroriste)
augmenterRessource():void
produireRessource():void
tirerRessource():void
effacerRessource():void
seBattre():void
getCapacite_de_production():int
getCapacite_de_tirer():int
getProduction_type():String
getTaux_de_production():double
getTirer_type():String
<<Java Class>>
Ressource
(default package)
compte: int
ident: int
type: String
x: int
y: int
quantite: int
Ressource(String,int)
getX():int
getY():int
getQuantite():int
setQuantite(int):void
setPosition(int,int):void
initialisePosition():void
toString():String
getType():String

Checklist des contraintes prises en compte:	*Nom(s) des classe(s) correspondante(s)*
Classe contenant un tableau ou une liste d'objets	public class Terrain public class Simulation
Classe statique contenant que des méthodes statiques	public class Outils
Héritage	public class Police extends Agent public class Terroriste extends Agent
Classe avec composition	public class Simulation public class Terrain
Classe avec un constructeur par copie ou clone()	public class Police extends Agent public class Terroriste extends Agent
Noms des classes créées (entre 4 et 10 classes)	public class Terrain public class Simulation public class Police extends Agent public class Terroriste extends Agent public class Ressource public abstract class Agent public class TestTerrain public class TestSimulation public class Outils

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

```
public abstract class Agent {
protected static int compte = 0;
protected final int ident;
protected int x;
protected int y;

public Agent(int x, int y) {
compte++;
ident = compte;
this.x = x;
this.y = y;
}

public int getX() {
return x;
}

public int getY() {
return y;
}

public double distance(int x, int y){
double dx = this.x - x;
double dy = this.y - y;
return (Math.sqrt(dx*dx + dy*dy));
}

public void seDeplacer(int newx, int newy){
this.x = newx;
this.y = newy;
}

public void setX(int x) {
this.x = x;
}

public void setY(int y) {
this.y = y;
}

public abstract void augmenterRessource();
public abstract void produireRessource();

public abstract void tirerRessource();
public abstract void effacerRessource();

public abstract void seBattre();

public abstract double getTaux_de_production();

public abstract int getCapacite_de_production();

public abstract int getCapacite_de_tirer();

public abstract String getProduction_type();
```

```
public abstract String getTirer_type();
}



public class Outils {
public static int nbBombeCreees = 0;
public static int nbBombeDetruites = 0;
public static int nbBaseCreees = 0;
public static int nbBaseDetruites = 0;

// une methode pour definir un evenement se passe ou non
public static boolean avoirLieu(double seuil){
return (Math.random()<seuil);
}
/**
*
* @param ress la ressource specifique qui fait l'objet
* @param plusOuMoins 0 si on va diminuer sa quantite, 1 pour l'augmenter
*/
public static void ressourceRecord(Ressource ress, int plusOuMoins) {
if(ress.getType() == "bombe"){
if(plusOuMoins == 1){
nbBombeCreees++;
}else{
nbBombeDetruites++;
}
}else{
if(plusOuMoins == 1){
nbBaseCreees++;
}else{
nbBaseDetruites++;
}
}
}
// faire une resume pour les ressources deja creees et detruites
public static void ressourceResume() {
System.out.println(String.format("Dans cette guerre, %d bombes et %d bases sont creees.\n %dodg bombes et
%d bases sont detruites.\n",
nbBombeCreees, nbBaseCreees, nbBombeDetruites, nbBaseDetruites));
}
// separateur de chaque affichage
public static void afficherSeparateur(){
for(int i = 0; i < 3; i++){
System.out.println("##########");
}
System.out.println("\n");
}
}

public class Police extends Agent {

private static double taux_de_production = 0.15; // le pourcentage pour creer des ressources
private static int capacite_de_production = 1; // le nombre de ressource a creer si possible
private static int capacite_de_tirer = 2; // le nombre de ressource a destruire si y en a
private static final String production_type = "base"; // le type de ressource que la classe peut produire
private static final String tirer_type = "bombe"; // le type de ressource que la classe peut detruire

public Police(int x, int y) {
super(x, y);
}
```

```
public Police (Police p1){
super(p1.x,p1.y);
}



// les 5 methodes ci-dessous ne servent que d'afficher des informations
@Override
public void augmenterRessource() {
System.out.println("Une nouvelle base est construite!\n");
}
@Override
public void produireRessource() {
System.out.println("Une base de police est construite!\n");
}
@Override
public void tirerRessource() {
System.out.println("Nous avons desactivés des bombees.\n");
}

@Override
public void effacerRessource() {
System.out.println("Les bombees sont désactivées, zone sérurisé.\n");
}

@Override
public void seBattre() {
System.out.println("Police : Un combat a lieu quelque part! Nous avons besoin de soutien!\n");
}

// les methodes ci-dessous ne servent que d'obtenir des valeurs
@Override
public int getCapacite_de_production() {
return capacite_de_production;
}

@Override
public int getCapacite_de_tirer() {
return capacite_de_tirer;
}

@Override
public String getProduction_type() {
return production_type;
}

@Override
public double getTaux_de_production() {
return taux_de_production;
}

@Override
public String getTirer_type() {
return tirer_type;
}
}

public class Ressource{
private static int compte = 0;
public final int ident;
public final String type;
private int x;
```

```
private int y;
private int quantite;

public Ressource(String type, int quantite) {
compte++;
this.ident = compte;
this.type = type;
this.quantite = quantite;
}

public int getX() {
return x;
}

public int getY() {
return y;
}

public int getQuantite() {
return quantite;
}

public void setQuantite(int quantite) {
this.quantite = quantite;
}
public void setPosition(int lig, int col){
this.x = lig;
this.y = col;
}

public void initialisePosition(){
this.x = -1;
this.y = -1;
}

@Override
public String toString() {
return type + " [id=" + ident + ", quantite=" + quantite +
"] en position (" + x + ", " + y + ") \n";
}

/*
* @return le type de Ressource sous format de string
*/
public String getType() {
return type;
}
}

import java.util.ArrayList;

public class Simulation {
// Les composants de base pour une simulation
private Terrain terrain;
private ArrayList<Ressource> ressources;
private ArrayList<Agent> agents;
// Ci-dessous des nombres pour compter les ressources creees et detruites

public Simulation(int lig, int col, int m, int n) {
this.terrain = new Terrain(lig, col);
```

```
ressources = new ArrayList<Ressource>(); // creation des ressource
for(int i = 0; i < m/2; i++){
Ressource base = new Ressource("base", 1); // creation d'une ressource
base.setPosition((int) (Math.random()*lig), (int) (Math.random()*col)); // definition de saposition
ressources.add(base); // ajout dans la liste
registerRessource(base); // enregistrement dans le terrain
Ressource bombe = new Ressource("bombe", 1);
bombe.setPosition((int) (Math.random()*lig), (int) (Math.random()*col));
ressources.add(bombe);
registerRessource(bombe);
}
if(m % 2 == 1){ // pour assurer que le nombre des ressources est bon
ressources.add(new Ressource("base", 1));
}

// compter les ressources et enrigistrer leur nombre cree
for(Ressource ress : ressources){
Outils.ressourceRecord(ress, 1);
}

// creation des agents
agents = new ArrayList<Agent>();
for(int i = 0; i < n/2; i++){
agents.add(new Police((int) (Math.random()*terrain.nbLigne), (int) (Math.random()*terrain.nbLigne)));
agents.add(new Terroriste((int) (Math.random()*terrain.nbLigne), (int) (Math.random()*terrain.nbLigne)));
}
if(n % 2 == 1){
agents.add(new Terroriste((int) (Math.random()*terrain.nbLigne), (int) (Math.random()*terrain.nbLigne)));
}
}

// enregistrer ou supprimer les ressources sur le terrain et dans la liste selon leur quantite
public void registerRessource(Ressource ress){
int x = ress.getX();
int y = ress.getY();
Ressource ter = terrain.tab[x][y];
// si la case est vide, ajouter la ressource
if(terrain.caseEstVide(x, y)){
terrain.setCase(x, y, ress);
}
// si la case a le meme type de ressource, augmenter la quantite
else if(ter.getType() == ress.getType()){
ter.setQuantite(ter.getQuantite()+1);
// si le type de ressource est different de celle qui existe dans la case, diminuer la quantite de cette derniere
}else{
ter.setQuantite(ter.getQuantite()-1);
// si la quantite reduit a 0, la ressource est supprimee
if (ter.getQuantite() == 0) {
terrain.videCase(x, y);
ressources.remove(ress);
}
}
}



public void agentsAction() {

for(Agent agent : agents){ // une cercle pour tous les agents existants
int x = agent.getX();
int y = agent.getY();
```

```
Ressource ress;

// si la case est vide, il y a une possibilite que l'agent depose sa ressource correspondante
if (terrain.caseEstVide(x, y)) {
if (Outils.avoirLieu(agent.getTaux_de_production())) {
agent.produireRessource();
ress = new Ressource(agent.getProduction_type(), agent.getCapacite_de_production());
ress.setPosition(x, y);
terrain.setCase(x, y, ress);
System.out.println(ress.toString());
Outils.ressourceRecord(ress, 1);
terrain.affiche();
}
} else {// sinon, voir le type de ressource
ress = terrain.getCase(x, y);
if (ress.getType() == agent.getProduction_type()) { // si la case contient le meme type de ressource que l'agent
peut creer, sa quantite incremente
agent.augmenterRessource();
System.out.println(ress.toString());
ress.setQuantite(ress.getQuantite() + agent.getCapacite_de_production());
for(int i = 0; i < agent.getCapacite_de_production(); i++){
Outils.ressourceRecord(ress, 1);
}
} else { // sinon, on l'enleve
int quantite = ress.getQuantite();
if (quantite > agent.getCapacite_de_tirer()) {
// si la quantite de ressource est superieure au nombre qu'un agent peut enlever
// on faire diminuer ce nombre
agent.tirerRessource();
System.out.println(ress.toString());
ress.setQuantite(quantite - agent.getCapacite_de_tirer());
for(int i = 0; i < agent.getCapacite_de_tirer(); i++){
Outils.ressourceRecord(ress, 0);
}
} else { // sinon, on supprimer la ressource
agent.effacerRessource();
ress.setQuantite(0);
ress = terrain.videCase(x, y);
System.out.println(ress.toString());
for (int i = 0; i < quantite; i++) {
Outils.ressourceRecord(ress, 0);
}
ress.initialisePosition();
ressources.remove(ress);
terrain.affiche();
}
}
}
// il y a une possibilite que les deux camps se battent, mais cela ne modifie pas aucune coordonnee
if(Outils.avoirLieu(terrain.nbLigne*terrain.nbColonnes/(agents.size()/2.)/100)){
agent.seBattre();
}
}
}
public void moveAgents(){ // deplacer tous les agents
for(Agent agent : agents){
agent.seDeplacer((int)Math.random()*terrain.nbLigne, (int)Math.random()*terrain.nbColonnes);
}
}


}
```

```java
/**
*
*/

/**
* @author Christophe Marsala, LI Junji, ZENG Fanxiang (LU2IN002 2020oct)
*
*
*/
public class TestTerrain {

/**
* @param args
*/
public static void main(String[] args) {
// Exemple de création de terrain
Terrain t = new Terrain(5,6);
// Terrain initial : il est vide
t.affiche();
// Informations sur le terrain
System.out.println("Informations sur le terrain:\n"+t);
// On créé une ressource
Ressource e1 = new Ressource("bombe",5);
// et on la place sur le terrain
if (t.setCase(2,3,e1))
System.out.println("Ajout de " +e1+" valide !");
else
System.out.println("Ajout incorrect: problème de coordonnées !");
// On créé une autre ressource
Ressource e2 = new Ressource("base",4);
// On rajoute la ressource sur le terrain
if (t.setCase(0,2,e2))
System.out.println("Ajout de " +e2+" valide !");
else
System.out.println("Ajout incorrect: problème de coordonnées !");
// On rajoute une autre ressource et on la met sur le terrain
if (t.setCase(4,1,new Ressource("base",3)))
System.out.println("Ajout valide !");
else
System.out.println("Ajout incorrect: problème de coordonnées !");

// Affichage du terrain avec les ressources ajoutées
t.affiche();
// Informations sur le terrain
System.out.println("Informations sur le terrain:\n"+t);
// Contenu d'une case:
System.out.println("Dans la case (1,4): "+t.getCase(1,4));
// Contenu d'une case:
System.out.println("Dans la case (0,2): "+t.getCase(0,2));
// Vidage d'une case:
System.out.println("Vidage d'une case:");
Ressource etaitDansLaCase = t.videCase(0,2);
if (etaitDansLaCase == null)
System.out.println("La case était déjà vide.");
else
System.out.println("La case contenait : "+etaitDansLaCase);
// Affichage du terrain avec les ressources ajoutées
t.affiche();
}

}
```

```
public class Terrain {
public final int nbColonnes;
public static final int NBCOLONNESMAX = 20;
public final int nbLigne;
public static final int NBLIGNEMAX= 20;
public Ressource[][] tab;

public Terrain(int nbLigne, int nbColonnes) {
super();
this.nbColonnes = nbColonnes;
this.nbLigne = nbLigne;
tab = new Ressource[nbLigne][nbColonnes];
if(nbColonnes<NBCOLONNESMAX || nbLigne <NBLIGNEMAX || nbColonnes> 1 || nbLigne >1 ) {
for(int i = 0 ; i < 1;i++) {
for(int j = 0 ; j < 1 ; j++) {
tab[i][j]= null;
}
}
}

for(int i = 0 ; i < nbLigne;i++) {
for(int j = 0 ; j < nbColonnes ; j++) {
tab[i][j]= null;
}
}
}
public Terrain() {
this(NBCOLONNESMAX,NBLIGNEMAX);
}
/*
* affiche le terrain
*/
public void affiche() {
for(int i = 0 ; i < nbLigne; i++) {
for(int j = 0; j<nbColonnes;j++) {
System.out.print(":-----");
}
System.out.print(":");
System.out.println();
for(int j =0 ; j< nbColonnes;j++){
if(!caseEstVide(i, j)){
String tmp = tab[i][j].getType();
// affichage des ressources dans la case
System.out.print(String.format("|%-5s", tmp));
}
else{
System.out.print("| ");
}
}
System.out.print("|");
System.out.println();
}
for (int i =0 ; i< nbColonnes;i++) {
System.out.print(":-----");
}
System.out.print(":\n\n");
}
public boolean caseEstVide(int lig,int col){
return tab[lig][col] == null;
}
```

```
public Ressource getCase(int lig, int col){
return tab[lig][col];
}

public boolean setCase(int lig, int col, Ressource ress){
if(caseEstVide(lig, col)&& sontValides(lig, col)){
tab[lig][col] = ress;
}
return caseEstVide(lig,col) && sontValides(lig, col);

}
public Ressource videCase (int lig, int col){
if(caseEstVide(lig,col)){
return null;
}
else{
Ressource tmp = tab[lig][col];
tab[lig][col] = null ;
return tmp ;
}
}



public boolean sontValides(int lig, int col) {
return col<NBCOLONNESMAX || lig <NBLIGNEMAX || col> 1 || lig >1;
}

@Override
public String toString() {
return "Terrain [nbColonnes=" + nbColonnes + ", nbLigne=" + nbLigne + "]";
}

}


public class Terroriste extends Agent {
private static double taux_de_production = 0.3; // le pourcentage pour creer des ressources
private static int capacite_de_production = 1; // le nombre de ressource a creer si possible
private static int capacite_de_tirer = 1; // le nombre de ressource a destruire si y en a
private static final String production_type = "bombe"; // le type de ressource que la classe peut produire
private static final String tirer_type = "base"; // le type de ressource que la classe peut detruire

public Terroriste(int x, int y) {
super(x, y);
}
public Terroriste(Terroriste t1){
super(t1.x, t1.y);
}

// les 5 methodes ci-dessous ne servent que d'afficher des informations
@Override
public void augmenterRessource() {
System.out.println("A new bombe is set!\n");
}
@Override
public void produireRessource() {
System.out.println("A bombe is set in the zone!\n");
}
@Override
public void tirerRessource() {
System.out.println("A police's base is destroyed!\n");
```

```
}

@Override
public void effacerRessource() {
System.out.println("No police's base is around, zone cleared.\n");
}

@Override
public void seBattre() {
System.out.println("Terroriste : We've met the police somewhere! Call for support!\n");
}
// les methodes ci-dessous ne servent que d'obtenir des valeurs
@Override
public int getCapacite_de_production() {
return capacite_de_production;
}

@Override
public int getCapacite_de_tirer() {
return capacite_de_tirer;
}

@Override
public String getProduction_type() {
return production_type;
}

@Override
public double getTaux_de_production() {
return taux_de_production;
}

@Override
public String getTirer_type() {
return tirer_type;
}
}

public class TestSimulation {
public static void main(String[] args) {
Simulation s1 = new Simulation(4, 5 , 100, 20);
for (int i = 0; i < 10; i++) {
s1.agentsAction();
s1.moveAgents();
Outils.ressourceResume();
Outils.afficherSeparateur();
}
}
}
```

public class Terrain

public class Simulation

public class Police extends Agent

public class Terroriste extends Agent

public class Ressource

public abstract class Agent

public class TestTerrain

public class TestSimulation

public class Outils