Mini-projet LU2IN002 - 2020-2021

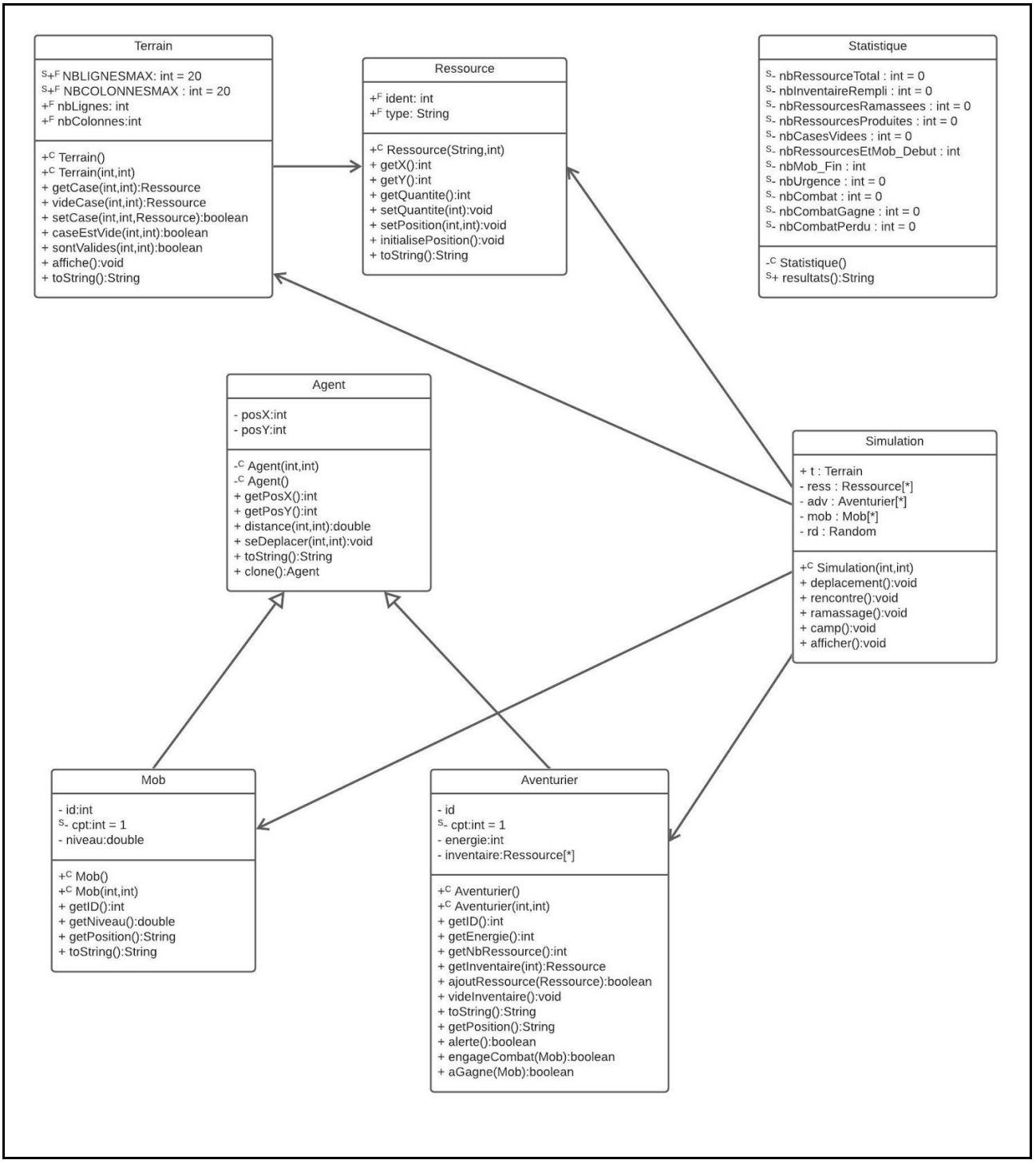
Nom : SINGH	*Nom :*
Prénom : Nawelle	*Prénom :*
N° étudiant : 28600503	*N° étudiant :*

Thème de simulation choisi (en 2 lignes max.)
Des aventuriers doivent ramasser des minerais et le ramener dans un camp. Chaque groupement de minerais est protégé par des monstres.

Description des classes et de leur rôle dans la simulation (2 lignes max par classe)
● Agent : classe mère de Aventurier et Mob. Elle possède un constructeur qui initialise la position d'un agent. ● Aventurier : classe fille de Agent. Elle possède toutes les méthodes pour gérer son inventaire et pour engager des combats contre les monstres. ● Mob : classe fille de Agent. Elle possède un attribut "niveau" qui jauge la puissance du monstre, qui est comparé dans la classe Combat. ● Simulation : classe qui initialise le terrain et tous les objets souhaités. Il instaure les règles de la simulation. ● Statistique : classe avec une méthode statique qui affiche uniquement les données statistiques de la simulation. ● TestSimulation : classe main.

Décrire, dans les grandes lignes, ce qui se passe durant la simulation (max. 5-6 lignes)
Lors d'un tour de simulation, les aventuriers se déplacent d'une case aléatoirement, sauf s'ils sont dans un cas d'urgence, c'est-à-dire que leur point de vie est en dessous de 30, dans ce cas, ils se dirigent vers le camp en position [0,0]. Ensuite, si les aventuriers croisent un monstre, ils le combattent. S'ils gagnent, ils peuvent ramasser les minerais. Au camp, les minerais sont transformés en lingot de fer et les aventuriers ont leurs points de vie remontés à 100.

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	Aventurier Simulation
Classe statique contenant que des méthodes statiques	Statistique

Héritage	Agent ← Aventurier Agent ← Mob
Classe avec composition	Simulation
Classe avec un constructeur par copie ou clone()	Agent
Noms des classes créées (entre 4 et 10 classes)	Agent Aventurier Mob Simulation Statistique TestSimulation

```java
import java.util.Random;

public class Agent{
	private int posX;
	private int posY;

	protected Agent(int x, int y){
		posX = x;
		posY = y;
	}

	protected Agent(){
		Random random = new Random();
		posX = random.nextInt(Terrain.NBLIGNESMAX);
		posY = random.nextInt(Terrain.NBCOLONNESMAX);
	}

	public int getPosX(){
		return posX;
	}

	public int getPosY(){
		return posY;
	}

	public double distance(int x, int y){
		return Math.sqrt(Math.pow(posX-x,2.0) + Math.pow(posY-y,2.0));
	}

	public void seDeplacer(int xnew, int ynew){
		posX = xnew;
		posY = ynew;
	}

	public String toString(){
		return "[" + posX + "," + posY + "]";
	}

	public Agent clone(){
		return new Agent(this.posX,this.posY);
	}

}
```

```java
public class Aventurier extends Agent{
	private int id;
	private static int cpt = 1;
	protected int energie;
	private Ressource[] inventaire = new Ressource[5];


	public Aventurier(){
		super(0,0);
		id = cpt;
		cpt++;
		energie = 100;
	}

	public Aventurier(int posX, int posY){
		super(posX,posY);
		id = cpt;
		cpt++;
		energie = 100;
	}

	public int getID(){
		return id;
	}

	public int getEnergie(){
		return energie;
	}

	public int getNbRessource(){
		int cpt = 0;
		for (int i = 0 ; i<inventaire.length ; i++){
			if (inventaire[i] != null){
				cpt +=1;
			}
		}
		return cpt;
	}

	public Ressource getInventaire(int i){
		return inventaire[i];
	}

	public boolean ajoutRessource(Ressource ress){
		for (int i = 0; i < inventaire.length ; i++){
			if (inventaire[i] == null){
				inventaire[i] = ress;
				return true;
			}
		}
```

```
            Statistique.nbInventaireRempli += 1;
            return false;
      }

      public void videInventaire(){
            for (int i = 0; i< inventaire.length ; i++){
                  if (inventaire[i]!=null){
                        inventaire[i] = null;
                  }
            }
      }


      public String toString(){
            String s = "** Aventurier" + id + " -- PV: "+ energie + " - position:"+
super.toString() + " - inventaire: ";
            if (inventaire[0]==null){
                  return s + "rien";
            }

            s = s + "\n";
            for (int i = 0; i<inventaire.length; i++){
                  s = s + "* ---" + inventaire[i] + "\n";
            }

            return s;
      }

      public String getPosition(){
            return super.toString();
      }


      //Methodes pour les combats
      public boolean alerte(){ //permet de savoir si le heros engagne le combat ou
pas
            if (energie <= 30){
                  Statistique.nbUrgence += 1;
                  return true;
            }
            return false;
      }

      public boolean engageCombat(Mob mob){
            if (this.alerte()) return false;
            System.out.println("-> Aventurier" + this.getID() + " VS " + "Mob" +
mob.getID() + " " + mob.getPosition());
            Statistique.nbCombat += 1;
            return true;
      }
```

```
	public boolean aGagne(Mob mob){
		/*s'il gagne, il perd 10 PV et retourne true, sinon il perd 30 PV et retourne false*/
		if((Math.random()*50)+50.0 > mob.getNiveau()){
			Statistique.nbCombatGagne += 1;
			this.energie -= 10;
			mob.seDeplacer(-1,-1);
			System.out.println("VS: Mob" + mob.getID() + " a perdu le combat. PV-10.");
			return true;
		}
		else{
			Statistique.nbCombatPerdu += 1;
			this.energie -= 30;
			System.out.println("VS: Aventurier"+ this.getID() + " a perdu le combat. PV-30");
			return false;
		}
	}
}
```

```java
public class Mob extends Agent{
	private int id;
	private static int cpt = 1;
	private double niveau;

	public Mob(){
		super();
		id = cpt;
		cpt++;
		niveau = Math.random()*100;
	}

	public Mob(int posX, int posY){
		super(posX,posY);
		id = cpt;
		cpt++;
		niveau = Math.random()*100;
	}

	public int getID(){
		return id;
	}

	public double getNiveau(){
		return niveau;
	}

	public String getPosition(){
		return super.toString();
	}

	public String toString(){
		return "xx Mob" + id + " -- position:"+super.toString();
	}

}
```

```java
import java.util.Random;

public class Simulation{
	public Terrain t;
	private Ressource[] ress;
	private Aventurier[] adv;
	private Mob[] mob;
	private Random rd = new Random();


	public Simulation(int m, int n){

		t = new Terrain();
		ress = new Ressource[m];
		mob = new Mob[m];
		adv = new Aventurier[n];

		Statistique.nbRessourceEtMob_Debut = m;
		Statistique.nbRessourceEtMob_Fin = m;

		System.out.println("... Initialisation des ressources et des mobs
sur le terrain, les mobs sont associés à chaque ressource ...");
		for(int i = 0 ; i < ress.length ; i++){
			int x = rd.nextInt(t.NBLIGNESMAX);
			int y = rd.nextInt(t.NBCOLONNESMAX);
			while(!t.caseEstVide(x,y) || ((x==0)&&(y==0))){
				x = rd.nextInt(t.NBLIGNESMAX);
				y = rd.nextInt(t.NBCOLONNESMAX);
			}
			ress[i] = new Ressource("Minerai",rd.nextInt(3)+1);
			Statistique.nbRessourceTotal += ress[i].getQuantite();
			t.setCase(x,y,ress[i]);


			mob[i] = new Mob();
			mob[i].seDeplacer(x,y);
		}

		for(int i = 0 ; i < ress.length ; i++){
			System.out.println(ress[i]);
		}
		for(int i = 0 ; i < mob.length ; i++){
			System.out.println(mob[i]);
		}

		System.out.println("... Initialisation des aventuriers au camp de
position:[0,0] ...");
		for(int i =0 ; i < adv.length ; i++){
			adv[i] = new Aventurier();
			System.out.println(adv[i]);
```

```
            }

            t.affiche();
      }

      public void deplacement(){
            /*Aventurier se deplace de facon aleatoire, sauf s'il est en etat
d'urgence : il part alors vers le camp en position [0,0] ou alors lorsque son
inventaire est plein*/
            for(int i = 0 ; i < adv.length ; i++){

                  System.out.print(">>> Aventurier" + adv[i].getID() + " " +
adv[i].getPosition() + " -> ");

                  if(adv[i].alerte() || adv[i].getNbRessource()==5){
                        if (adv[i].getPosX() == 0){
                              adv[i].seDeplacer(adv[i].getPosX(),
adv[i].getPosY()-1);
                              System.out.print(adv[i].getPosition() + "\n");
                        }
                        else{
                              adv[i].seDeplacer(adv[i].getPosX()-1,
adv[i].getPosY());
                              System.out.print(adv[i].getPosition() + "\n");
                        }
                  }
                  else{
                        int x = adv[i].getPosX();
                        int y = adv[i].getPosY();

                        if ((x == 0)&&(y == 0)){

adv[i].seDeplacer(x+(rd.nextInt(2)),y+(rd.nextInt(2)));
                              System.out.print(adv[i].getPosition() + "\n");
                              continue;
                        }
                        if((x == t.NBLIGNESMAX-1)&&(y==t.NBCOLONNESMAX-1)){

adv[i].seDeplacer(x-rd.nextInt(2),y-rd.nextInt(2));
                              System.out.print(adv[i].getPosition() + "\n");
                              continue;
                        }
                        if((x == 0)&&(y == t.NBCOLONNESMAX-1)){

adv[i].seDeplacer(x+rd.nextInt(2),y-rd.nextInt(2));
                              System.out.print(adv[i].getPosition() + "\n");
                              continue;
                        }
                        if((x == t.NBLIGNESMAX-1)&&(y == 0)){

adv[i].seDeplacer(x-rd.nextInt(2),y+rd.nextInt(2));
```

```
                                    System.out.print(adv[i].getPosition() + "\n");
                                    continue;
                              }
                              if (x == 0){

adv[i].seDeplacer(x+(rd.nextInt(2)),y+(rd.nextInt(3)-1));
                                    System.out.print(adv[i].getPosition() + "\n");
                                    continue;
                              }
                              if (y == 0){

adv[i].seDeplacer(x+(rd.nextInt(3)-1),y+(rd.nextInt(2)));
                                    System.out.print(adv[i].getPosition() + "\n");
                                    continue;
                              }
                              if (x == t.NBLIGNESMAX-1){

adv[i].seDeplacer(x-rd.nextInt(2),y+(rd.nextInt(3)-1));
                                    System.out.print(adv[i].getPosition() + "\n");
                                    continue;
                              }
                              if (y == t.NBCOLONNESMAX-1){

adv[i].seDeplacer(x+(rd.nextInt(3)-1),y-rd.nextInt(2));
                                    System.out.print(adv[i].getPosition() + "\n");
                                    continue;
                              }

adv[i].seDeplacer(x+(rd.nextInt(3)-1),y+(rd.nextInt(3)-1));
                              System.out.print(adv[i].getPosition() + "\n");
                        }

                  }
            }

            public void rencontre(){
                  /*Aventurier rencontre un Mob et ils se battent. Si l'aventurier
gagne, alors il recupere les mineraux que le mob gardait. Si son inventaire est
plein, il laisse le minerai la ou il l'a trouve.*/
                  for(int j = 0 ; j < adv.length ; j++){

                        for (int i = 0 ; i < mob.length ; i++){

                              if((mob[i].getPosX()==adv[j].getPosX()) &&
(mob[i].getPosY()==adv[j].getPosY())){

                                    if(adv[j].engageCombat(mob[i])){
                                          if(adv[j].aGagne(mob[i])){
                                                Statistique.nbRessourceEtMob_Fin
-= 1;
```

```
if(adv[j].ajoutRessource(ress[i])){
                                                      System.out.println("++ " +
ress[i].type + ress[i].ident + " a ete ramasse par Aventurier" + adv[j].getID()
+ "\n" + adv[j].toString());

Statistique.nbRessourcesRamassees += ress[i].getQuantite();
                                                      Ressource poubelle =
t.videCase(ress[i].getX(),ress[i].getY());

ress[i].initialisePosition();
                                                      Statistique.nbCasesVidees
+= 1;
                                                      break;
                                                }
                                                else{
                                                      System.out.println("Il n'y
a plus de place dans l'inventaire !");
                                                }
                                          }
                                          break;
                                    }
                                    break;
                              }
                        }
                  }
            }

      public void ramassage(){
            /*S'il n'y a pas de Mob, alors l'aventurier ramasse le minerai,
sauf si son inventaire est plein*/
            for(int i = 0 ; i < adv.length ; i++){
                  for(int j = 0 ; j < ress.length ; j++){
                        if(((adv[i].getPosX() == ress[j].getX()) &&
(adv[i].getPosY() == ress[j].getY())) && ((adv[i].getPosX() !=
mob[j].getPosX())||(adv[i].getPosY() != mob[j].getPosY()))){
                              if(adv[i].ajoutRessource(ress[j])){
                                    System.out.println("++ " + ress[j].type
+ ress[j].ident + " a ete ramasse par Aventurier" + adv[i].getID() + "\n" +
adv[i].toString());
                                    Statistique.nbRessourcesRamassees +=
ress[j].getQuantite();
                                    Ressource poubelle =
t.videCase(ress[j].getX(),ress[j].getY());
                                    ress[j].initialisePosition();
                                    Statistique.nbCasesVidees += 1;
                                    break;
                              }
                              else{
                                    System.out.println("Il n'y a plus de
place dans l'inventaire !");
                              }
```

```
                        }
                }
        }
    }

    public void camp(){
        /*l'aventurier voit ses PV remontes a 100 et depose les mineraux en
vidant son inventaire. Le camp s'occupe de transformer les mineraux en lingot de
fer.*/
        int cpt = 0;
        for(int i = 0 ; i < adv.length ; i++){
            if((adv[i].getPosX()==0)&&(adv[i].getPosY()==0)){
                System.out.println("&& Guerison de Aventurier" +
adv[i].getID());
                adv[i].energie = 100;
                cpt += adv[i].getNbRessource();
                adv[i].videInventaire();
            }
        }

        while (cpt > 0){
            System.out.println("@@ Creation d'un lingot de fer.");
            Statistique.nbRessourcesProduites += 1;
            cpt--;
        }
    }

    public void afficher(){
        System.out.println("~~~~~~ Affichage graphique de la situation
~~~~~~");
        for(int i = 0 ; i < adv.length ; i++){
            System.out.println(adv[i]);
        }
    }

}
```

```
public class Statistique{
	public static int nbRessourceTotal = 0;
	public static int nbInventaireRempli = 0;
	public static int nbRessourcesRamassees = 0;
	public static int nbRessourcesProduites = 0;
	public static int nbCasesVidees = 0;
	public static int nbRessourceEtMob_Debut;
	public static int nbMob_Fin;
	public static int nbUrgence = 0;
	public static int nbCombat = 0;
	public static int nbCombatGagne = 0;
	public static int nbCombatPerdu = 0;

	private Statistique(){}

	public static String resultats(){
		String s = "";
		s = "########## STATISTIQUE ##########\n\n";
		s += "~~~~~~ Ramassage ~~~~~~\n";
		s += "== Quantite de minerais totale : " + nbRessourceTotal + "\n";
		s += "== Nombre de cases qui ont ete videes : " + nbCasesVidees +
"\n";
		s += "== Nombre de minerais ramasses : "+ nbRessourcesRamassees +
"\n";
		s += "== Nombre de lingots produits : "+ nbRessourcesProduites +
"\n";
		s += "== Nombre de fois où l'inventaire a ete rempli : " +
nbInventaireRempli + "\n";
		s += "==> Au total, il y a eu "+
String.format("%.2f",(((float)nbRessourcesRamassees)/nbRessourceTotal)*100.0) +
"% de minerais qui ont ete ramasses, dont " +
String.format("%.2f",((float)nbRessourcesProduites)/nbRessourcesRamassees*100.0)
+ "% qui ont ete transformes en lingot.\n";
		s += "~~~~~~ Population de monstres ~~~~~~\n";
		s += "== Nombre de monstres au debut : " + nbRessourceEtMob_Debut +
"\n";
		s += "== Nombre de monstres a la fin : " + nbMob_Fin + "\n";
		s += "==> " + String.format("%.2f",((float)(nbRessourceEtMob_Debut
- nbMob_Fin))/nbRessourceEtMob_Debut*100.0) + "% de monstres qui ont ete
extermines.\n";
		s += "~~~~~~ Combats ~~~~~~\n";
		s += "== Nombre d'urgence : " + nbUrgence + "\n";
		s += "== Nombre de combats : " + nbCombat + "\n";
		s += "== Nombre de combats gagnes : " + nbCombatGagne + " / perdus
: " + nbCombatPerdu + "\n";
		s += "==> " +
String.format("%.2f",((float)nbCombatGagne)/nbCombat*100.0) + "% ont ete
gagnes.\n";
		s += "################################\n";
		return s;
```

```
        }
}
```

```java
public class TestSimulation{
	public static void main(String[]args){
		Simulation s1 = new Simulation(250,10); //premiere valeur: nombre de ressource //deuxieme valeur: nombre d'aventuriers

		int tour = 100; //modifiable à souhait

		for(int i = 0; i < tour ; i++){
			System.out.println("------ TOUR " + (i+1) + " --------------------");
			System.out.println("~~~~~~ Deplacement des aventuriers ~~~~~~");
			s1.deplacement();
			System.out.println("~~~~~~ Combat ~~~~~~");
			s1.rencontre();
			System.out.println("~~~~~~ Ramassage ~~~~~~");
			s1.ramassage();
			System.out.println("~~~~~~ Camp ~~~~~~");
			s1.camp();
			s1.afficher();
			System.out.println("------ FIN DU TOUR " + (i+1) + " -------------");
		}

		s1.t.affiche();

		System.out.println(Statistique.resultats());

	}
}
```