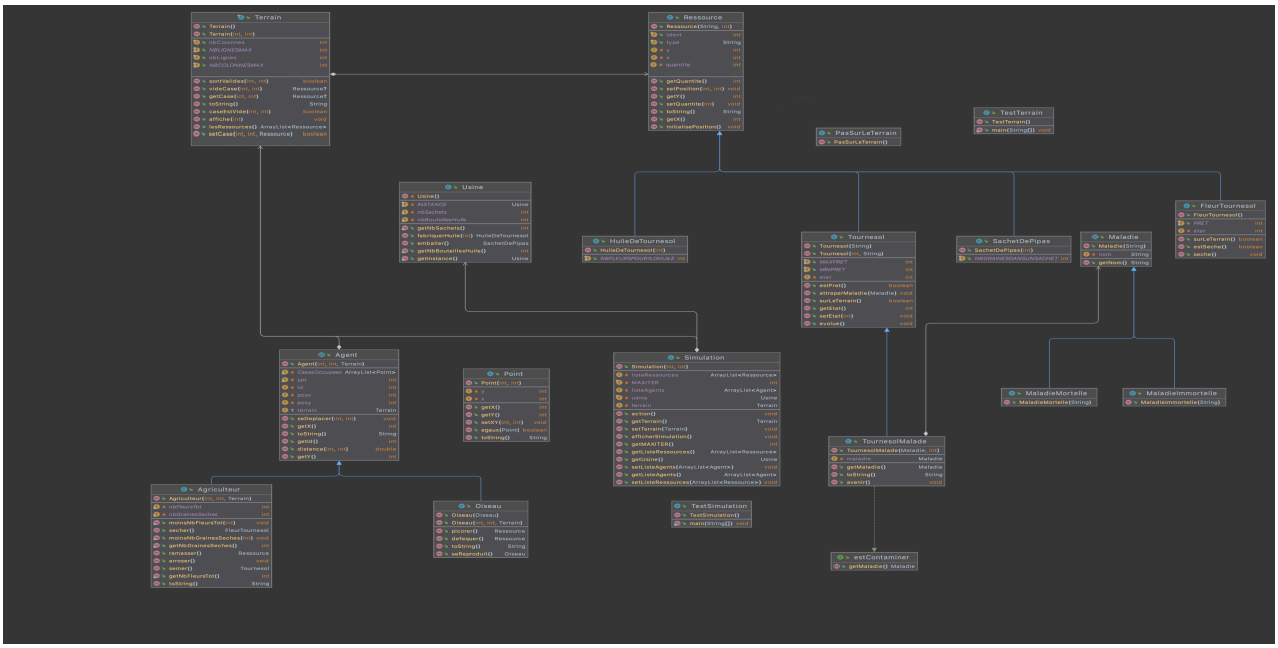
Projet LU2IN002 - 2022-2023

Numéro du groupe de TD/TME : 11

Nom : DAHER	*Nom :* CHEMALI
Prénom : Sarah	*Prénom :* Maïssa
N° étudiant : 21100791	*N° étudiant :* 28722554

Thème choisi (en 2 lignes max.)
Simulation de la récolte dans un champ de tournesol et la transformation des fleurs de tournesol en huile et emballage des graines dans des sachets de pipas dans l'usine.

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	Agent / Simulation
Classe avec membres et méthodes statiques	Agent / Agriculteur / FleurTournesol / HuileDeTournesol/ SachetDePipas / Tournesol / Usine
Classe abstraite et méthode abstraite	Agent / Maladie
Interface	estContaminer

Classe avec un constructeur par copie ou clone()	Oiseau
Définition de classe étendant Exception	PasSurLeTerrain
Gestion des exceptions	Simulation / Agriculteur / FleurTournesol / Oiseau / Tournesol / TournesolMalade

Présentation brève de votre projet (max. 10 lignes) : texte libre expliquant en quoi consiste votre projet.

Notre projet simule une récolte dans un champ de tournesol et la croissance de ces fleurs (graines (état 0) -> fleurs (état 5)). Au-delà de l'état 5, la fleur meurt et redevient une graine (état 0). L'agriculteur arrose (etat++), sème les graines de tournesol, cueille les fleurs, et il peut soit les sécher si il tombe sur une case vide du terrain soit les garder et les transférer vers l'usine qui fabrique de l'huile. Quand les fleurs sont sèches, il les ramasse ce qui lui procure des graines de tournesol qu'il pourra transférer à l'usine pour les emballer en sachet de pipas. Malheureusement, nos tournesols peuvent attraper des maladies (si celles-ci sont mortelles, elles sont retirées du terrain sinon leur état régresse de 1). Des oiseaux sont aussi présents sur notre terrain, ils peuvent se reproduire, ils picorent le contenu de la case si elle n'est pas vide, sinon il se déplace. Ils peuvent aussi déféquer, en l'occurence, notre tournesol évolue mais notre fleur est jetée.

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

Agent.java:

```
import java.util.ArrayList;
public abstract class Agent{
  //attributs
    private int id;
    private int posx;
    private int posy;
    protected Terrain terrain;
    private static ArrayList<Point> CasesOccupees = new ArrayList<Point>();
    private static int cpt = 0;

  // constructeur
    public Agent(int x, int y, Terrain t){
        if (!t.sontValides(x,y) ){
            System.out.println("coordonnees saisies non valides");
            return;
        }
        Point newp = new Point(x,y);
        for(Point p : CasesOccupees){
            if(newp.egaux(p)){
                System.out.println("il y a deja un agent sur cette case (" +x+","+y+")");
                return;
            }
        }
        posx=x;
        posy=y;
        CasesOccupees.add(new Point(x,y));
```

```
        terrain=t;
        cpt++;
        id=cpt;

    }
  //methodes
   public double distance(int x, int y){
        return Math.sqrt((posx-x)*(posx-x) + (posy-y)*(posy-y));
    }

    public void seDeplacer(int newx, int newy){
        if(this.terrain != null){
        if (!terrain.sontValides(newx,newy) ){
            System.out.println("coordonnees saisies non valides");
            return;
        }
        Point newp = new Point(newx,newy);
        Point thisp = new Point(posx,posy);
        int indice = 0;
        for(Point p : CasesOccupees){
            if(newp.egaux(p)){
                System.out.println("il y a deja un agent sur la case" + p);
                return;
            }
            if(thisp.egaux(p)){
                indice =CasesOccupees.indexOf(p);
            }
        }
        posx = newx;
        posy = newy;
        System.out.println("se deplace en ("+ newx + ","+ newy +")");
        CasesOccupees.remove(indice);
        CasesOccupees.add(newp);

        }
    }
    public String toString(){
        return "(agent " + id + " ) en position ("+ posx + ","+ posy +")";
    }

  // accesseurs
    public int getId(){
        return id;
    }
    public int getX(){
        return posx;
    }
    public int getY(){
        return posy;
    }
}
```

Agriculteur:

```
import java.util.ArrayList;

public class Agriculteur extends Agent {
  //attributs
  private static int nbFleursTot = 0;
  private static int nbGrainesSeches = 0;

//constructeur
  public Agriculteur(int x, int y, Terrain t) {
    super(x, y, t);

  }

  //methodes
  public Tournesol semer() {
    if(this.terrain== null) return null;
    System.out.println(this.toString() + " seme graines de tournesol");
    Tournesol t = new Tournesol("tournesol");
    t.setPosition(getX(), getY());
    terrain.setCase(getX(), getY(), t);
    return t;
  }

  public void arroser() throws PasSurLeTerrain {
    if(this.terrain == null) return ;
    if (terrain.caseEstVide(getX(), getY())) {System.out.println(this.toString() + " n'a rien a
arroser"); return;}
    Ressource r = terrain.getCase(getX(),getY());
    if(r instanceof Tournesol){
       ((Tournesol)r).evolue();
       System.out.println(this.toString() + " arrose " + r.toString());
    }
  }

  public Ressource ramasser() throws PasSurLeTerrain {
    if(this.terrain == null) return null;
    if (terrain.caseEstVide(getX(), getY())){ System.out.println(this.toString() + " n'a rien a
recuperer"); return null;}
    Ressource r = terrain.getCase(getX(), getY());
      if (r instanceof Tournesol) {
        if (((Tournesol) r).estPret()) {
          System.out.println(this.toString() + " cueille fleur de tournesol");

          r.initialisePosition();
          nbFleursTot++;
          return terrain.videCase(getX(), getY());
        }
        else {
          System.out.println(this.toString() + " ne cueille rien car " + r.toString() + " n'est pas pret");
          return null;
        }
```

```
    } else if (r instanceof FleurTournesol) {
       if (((FleurTournesol) r).estSeche()) {
         System.out.println(this.toString() + " recupere les graines seches de tournesol");
         r.initialisePosition();
         nbGrainesSeches += 100;
         return terrain.videCase(getX(), getY());
       } else {
          System.out.println(this.toString() + " ne recupere rien car " + r.toString() + " n'est pas
seche");
          return null;
         }
      }
   return null;
  }

  public FleurTournesol secher() throws PasSurLeTerrain {
   if (this.terrain==null) return null;
   if (terrain.caseEstVide(getX(), getY())  && nbFleursTot>0 ) {
     System.out.println(this.toString() + " fait secher fleur de tournesol");
     FleurTournesol f =new FleurTournesol();
     f.setPosition(getX(), getY());
     terrain.setCase(getX(), getY(), f);
     nbFleursTot--;
     return f;
   }
   return null;
  }

  public static void moinsNbFleursTot(int nb) {
   nbFleursTot -= nb;
  }

  public static void moinsNbGrainesSeches(int n) {
   nbGrainesSeches -= n;
  }

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

  //accesseurs
  public static int getNbGrainesSeches() {
   return nbGrainesSeches;
  }

  public static int getNbFleursTot() {
   return nbFleursTot;
  }
}
```

estContaminer:

```
public interface estContaminer{
```

```
  public Maladie getMaladie();
}
```

FleurTournesol:

```
public class FleurTournesol extends Ressource{
    public static final int PRET=5;
    private int etat; //etat correspondant a l'avancee de l'assechement de la fleur

  //constructeur
    public FleurTournesol(){
        super("fleur de tournesol",1);
        etat=0;
    }

  // methodes
    public void seche() throws PasSurLeTerrain{
      if (!this.surLeTerrain()) throw new PasSurLeTerrain();
      etat++;
      System.out.println(super.toString()+" seche");
    }
    public boolean estSeche(){
        return (etat >=PRET);
    }
    public boolean surLeTerrain() {
      return this.getX() != -1 && this.getY() != -1;
    }
}
```

HuileDeTournesol:

```
public class HuileDeTournesol extends Ressource{
  //attribut
  public static final int NBFLEURSPOUR1LDHUILE = 1;

  //constructeur
  public HuileDeTournesol(int n){
    super("Huile de tournesol", n);
  }
}
```

Maladie:

```
public abstract class Maladie {
  //attribut
  private String nom;

  //constructeur
  public Maladie(String n){
    nom = n;
  }

  //accesseur
  public String getNom(){
    return nom;
```

```
  }
}
```

MaladieImmortelle:

```
public class MaladieImmortelle extends Maladie{

  public MaladieImmortelle(String n){
    super(n);
  }
}
```

MaladieMortelle:

```
public class MaladieMortelle extends Maladie{

  public MaladieMortelle(String n){
    super(n);
  }
}
```

Oiseau:

```
public class Oiseau extends Agent {
//constructeurs
  public Oiseau(int x, int y, Terrain t) {
    super(x, y, t);
  }

  public Oiseau(Oiseau o) {
    super((int) (Math.random() * (o.terrain).nbLignes), (int) (Math.random() *
(o.terrain).nbColonnes), o.terrain);
  }

  //methodes
  public Ressource picorer() throws PasSurLeTerrain{

    if (this.terrain == null) return null;
    if (!(terrain.caseEstVide(getX(), getY()))) {
        System.out.println(this.toString() + " picore " + terrain.getCase(getX(), getY()).toString());
        (terrain.getCase(getX(), getY())).initialisePosition();
        return terrain.videCase(getX(), getY());
    } else {
        System.out
            .println("Rien à faire, l'oiseau ne picore pas car la case (" + getX() + "," + getY() + ")  est
vide");
        System.out.println("L'oiseau se déplace");
        this.seDeplacer((int) (Math.random() * terrain.nbLignes), (int) (Math.random() *
terrain.nbColonnes));
        return null;
      }
  }

  public Ressource defequer() throws PasSurLeTerrain{
    if (this.terrain == null) return null;
```

```
    System.out.println(this.toString() + " fait ses besoins, la terre est nourrie avec cet engrais ");
    if (!terrain.caseEstVide(getX(), getY())) {
      if (terrain.getCase(getX(), getY()) instanceof Tournesol){ ((Tournesol) terrain.getCase(getX(),
getY())).evolue(); return null;}
      else if (terrain.getCase(getX(), getY()) instanceof FleurTournesol) {
          System.out.println((terrain.getCase(getX(), getY())).toString() + " est souillé ; doit etre
jete");
          (terrain.getCase(getX(), getY())).initialisePosition();
          return terrain.videCase(getX(), getY());
        }
    }
    return null;
  }

  public Oiseau seReproduit() {
    if (this.terrain != null){
      System.out.println(this.toString() + " se reproduit");
      return new Oiseau(this);
    }else return null;
  }

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

PasSurLeTerrain:

```
public class PasSurLeTerrain extends Exception{
  public PasSurLeTerrain(){
    super("la ressource n'est pas sur le terrain : pas d'action a realiser");
  }
}
```

Point:

```
public class Point{
  //attributs
    private int x;
    private int y;

  //constructeur
    public Point(int x,int y){
        this.x=x;
        this.y=y;
    }
    public boolean egaux(Point p){
        return x==p.x && y==p.y;
    }

    public String toString(){
      return "(" + x + "," + y + ")";
    }
  //accesseurs
```

```
  public int getX(){
    return x;
  }
  public int getY(){
    return y;
  }
 //assesseurs
  public void setXY(int newx, int newy){
    x = newx;
    y = newy;
  }
}
```

SachetDePipas:

```
public class SachetDePipas extends Ressource{
 public static final int NBGRAINESDANSUNSACHET = 70;

 public SachetDePipas(int n){
  super("Sachet de pipas", n);
 }
}
```

Simulation:

```
import java.util.ArrayList;

public class Simulation {
 //attributs
 private Terrain terrain;
 private final int MAXITER = 20;
 private ArrayList<Agent> listeAgents;
 private ArrayList<Ressource> listeRessources;
 private final Usine usine = Usine.getInstance();

 //constructeur
 public Simulation(int m, int n) {
  terrain = new Terrain(6, 6);
  listeRessources = new ArrayList<Ressource>();
  listeAgents = new ArrayList<Agent>();
  for (int i = 0; i < m; i++) {
   Ressource r;
   int x, y;
   r = new Tournesol("tournesol"); // au debut les tournesols de sont pas malades
   x = (int) (Math.random() * terrain.nbLignes);
   y = (int) (Math.random() * terrain.nbColonnes);
   listeRessources.add(r);
   r.setPosition(x, y);
   terrain.setCase(x, y, r);
  }
  for (int i = 0; i < n; i++) {
   Agent a;
   int x, y;
   x = (int) (Math.random() * terrain.nbLignes);
```

```
    y = (int) (Math.random() * terrain.nbColonnes);
    if (Math.random() < 0.3)  a = new Oiseau(x, y, terrain);
    else  a = new Agriculteur(x, y, terrain);
    if(a.getId()>0) listeAgents.add(a);
   }
  }
  //methodes
  public void action() {
   int cpt = 1;
   System.out.println();
   System.out.println("TOUR " + 0);
   terrain.affiche(30);
   while (cpt <= MAXITER) {
    System.out.println();
    System.out.println("TOUR " + cpt);

    for (int i = 0; i < listeAgents.size(); i++) {
     try{
      int alea;
      if (listeAgents.get(i) instanceof Agriculteur) {
       alea = (int) (Math.random() * 10);
       if (alea == 0) listeRessources.add(((Agriculteur) (listeAgents.get(i))).semer());
       if (alea == 6) ((Agriculteur) (listeAgents.get(i))).arroser();
       if (alea < 6) {
        Ressource r =((Agriculteur) (listeAgents.get(i))).ramasser();
        if(r!=null) listeRessources.remove(listeRessources.indexOf(r));
       }
       if ((6 <= alea) && (alea <= 10)) listeRessources.add(((Agriculteur)
(listeAgents.get(i))).secher());

      }
      else if (listeAgents.get(i) instanceof Oiseau) {
       alea = (int) (Math.random() * 3);
       if (alea == 0) {
        Ressource r =((Oiseau) (listeAgents.get(i))).picorer();
        if(r!=null) listeRessources.remove(listeRessources.indexOf(r));
       }
       if (alea == 1) {
        Ressource r =((Oiseau) (listeAgents.get(i))).defequer();
        if(r!=null) listeRessources.remove(listeRessources.indexOf(r));
       }
       if (alea == 2){
        Oiseau nvo =( (Oiseau) (listeAgents.get(i))).seReproduit();
        if(nvo!=null && nvo.getId()>0) listeAgents.add(nvo);
       }
      }
     }catch (PasSurLeTerrain e){ System.out.println(e);}

     if (Math.random() < 0.6) {
      int x = (int) (Math.random() * terrain.nbLignes);
      int y = (int) (Math.random() * terrain.nbColonnes);
```

```
    if ((listeAgents.get(i)) != null) (listeAgents.get(i)).seDeplacer(x, y);
   }
  }


  boolean soleil = false;
  if (Math.random() < 0.6) {
   System.out.println("Il y a du soleil !");
   soleil = true;
  }
  else System.out.println("Il n'y a pas de soleil aujourd'hui");
  double p = Math.random();

  for (int i = 0; i < listeRessources.size(); i++) {
     Ressource r = listeRessources.get(i);
     if(r==null){listeRessources.remove(i); continue;}
     try{
     if (r instanceof Tournesol){
        if(soleil) ((Tournesol) r).evolue();
        int x=r.getX(), y =r.getY(), etat = ((Tournesol) r).getEtat();

        if ((p < 0.1)&&(!(r instanceof TournesolMalade))) {
          Maladie m;
          if(p<0.5) m = new MaladieImmortelle("Mindiou");
          else m = new MaladieMortelle("Phoma");
          ((Tournesol)r).attraperMaladie(m);

          r.initialisePosition();
          terrain.videCase(x,y);
          listeRessources.remove(i);
          TournesolMalade tm = new TournesolMalade(m,etat);
          tm.setPosition(x, y);
          terrain.setCase(x, y, tm);
          listeRessources.add(tm);

        }
        else if((r instanceof TournesolMalade)){
          if (((TournesolMalade)r).surLeTerrain()){
            ((TournesolMalade)r).avenir();
            if(((TournesolMalade)r).getMaladie() instanceof MaladieMortelle){
              r.initialisePosition();
              terrain.videCase(x,y);
              listeRessources.remove(i);
            }
          }
        }
   }
   else if (r instanceof FleurTournesol && soleil) ((FleurTournesol) r).seche();
   }catch (PasSurLeTerrain e){ System.out.println(e);}
   if(r==null){listeRessources.remove(i);}
  }
```

```
    SachetDePipas s=usine.emballer();
    if(s!=null) listeRessources.add(s);
    HuileDeTournesol h =
usine.fabriquerHuile((int)(Math.random()*Agriculteur.getNbFleursTot()));
    if(h!=null) listeRessources.add(h);
    terrain.affiche(30);
    cpt++;

   }
}
    //affichage
 public void afficherSimulation() {
   System.out.println("LA LISTE DES RESSOURCES :");
   for (Ressource r : listeRessources) {
    System.out.println(r);
   }
   System.out.println();
   System.out.println("LA LISTE DES AGENTS : ");
   for (Agent a : listeAgents) {
    System.out.println(a);
   }
 }


 //accesseurs
 public Terrain getTerrain() {
        return terrain;
 }



 public int getMAXITER() {
         return MAXITER;
 }

 public ArrayList<Agent> getListeAgents() {
         return listeAgents;
 }


 public ArrayList<Ressource> getListeRessources() {
         return listeRessources;
 }

 public Usine getUsine() {
         return usine;
 }
 //assesseurs
  public void setListeRessources(ArrayList<Ressource> listeRessources) {
         this.listeRessources = listeRessources;
 }
```

```
   public void setListeAgents(ArrayList<Agent> listeAgents) {
            this.listeAgents = listeAgents;
   }

   public void setTerrain(Terrain terrain) {
            this.terrain = terrain;
   }
}
```

TestSimulation:

```
public class TestSimulation{
  public static void main(String[] args){

     System.out.println("Bienvenue dans notre champ de tournesol");
    System.out.println("Commençons par placer nos ressources et nos agents et par les présenter");
     System.out.println();

    Simulation s = new Simulation(15, 20);

    System.out.println();
    s.afficherSimulation();

     System.out.println("Que la simulation commence !");
    s.action();

    System.out.println("FIN DE LA SIMULATION ! ");

   System.out.println("Nombre de sachets de pipas créés durant cette simulation " +
Usine.getNbSachets());
    System.out.println("Nombre de bouteilles d'huiles créées durant cette simulation " +
Usine.getNbBouteillesHuile());

    System.out.println("Nombre graines restantes: " + Agriculteur.getNbGrainesSeches());
     System.out.println("Nombre fleurs restantes :" + Agriculteur.getNbFleursTot());
  }
}
```

Tournesol:

```
public class Tournesol extends Ressource {
  public static final int MINPRET = 3;
  public static final int MAXPRET = 5;
  private int etat; // indique l'etat de vie du tournesol : 0-> graine, {1,..,MINPRET-1}-> etapes de
croissance, {MINPRET,..,MAXPRET}-> pret a etre cueilli, {MAXPRET+1,...}-> trop vieux
(deperit et repasse a l'etat 0)


  public Tournesol(String nom) {
    super(nom, 1);
    etat = 0;
}

  public Tournesol(int etat, String nom) {
```

```
  super(nom, 1);
  this.etat = etat;
 }

 public void evolue() throws PasSurLeTerrain {
  if (!this.surLeTerrain()) throw new PasSurLeTerrain();
  etat++;
  if (etat < MINPRET) System.out.println(super.toString() + " pousse !");
  else if (etat <=MAXPRET) System.out.println(super.toString() + " est pret a etre cueilli");
  else System.out.println("trop tard...  " + super.toString() + " redevient graine ");
 }

 public int getEtat() {
  return etat;
 }

 public void setEtat(int newEtat) {
  etat = newEtat;
 }

 public boolean estPret() {
  return (etat <= MAXPRET) && (etat>=MINPRET);
 }

 public boolean surLeTerrain() {
  return this.getX() != -1 && this.getY() != -1;
 }


 public void attraperMaladie(Maladie m) throws PasSurLeTerrain  {
  if (!this.surLeTerrain()) throw new PasSurLeTerrain();
  System.out.println(this.toString() + " tombe malade");
 }
}
```

TournesolMalade:

```
public class TournesolMalade extends Tournesol implements estContaminer{
 private Maladie maladie;

 public TournesolMalade(Maladie m, int n){
  super(n,"tournesol malade");
  maladie = m;
 }

 public void avenir() throws PasSurLeTerrain{
  if(!this.surLeTerrain()) throw new PasSurLeTerrain();
  if (this.maladie instanceof MaladieMortelle){
   System.out.println("Oh non ! " + this.toString() + " meurt \n");
   this.setEtat(0);
  }
  else if (this.maladie instanceof MaladieImmortelle){
   if(this.getEtat()>0 ){
```

```
      this.setEtat(this.getEtat()-1);
      System.out.println(this.toString() + " regresse de 1 etat \n");
     }
    }
  }
  public Maladie getMaladie(){
    return maladie;
  }

  public String toString(){
    return  super.toString() + "malade de maladie " + maladie.getNom();
  }
}
```

Usine:

```
import java.util.ArrayList;
public class Usine {
  private static int nbSachets = 0;
  private static int nbBouteillesHuile = 0;
  private static final Usine INSTANCE = new Usine();

  private Usine() { }

  public static Usine getInstance() {
    return INSTANCE;
  }
  public static int  getNbSachets(){
    return nbSachets;
  }
  public static int  getNbBouteillesHuile(){
    return nbBouteillesHuile;
  }
  public SachetDePipas emballer() {
    int nb = 0;
    int grainesDispo = Agriculteur.getNbGrainesSeches();
    while (grainesDispo >= SachetDePipas.NBGRAINESDANSUNSACHET) {
      nb++;
      grainesDispo -= SachetDePipas.NBGRAINESDANSUNSACHET;

    }
      Agriculteur.moinsNbGrainesSeches(SachetDePipas.NBGRAINESDANSUNSACHET*nb);
    System.out.println("On a emballé " + nb + " sachets de pipas");
    nbSachets += nb;
    if(nb>0) return new SachetDePipas(nb);
    else return null;
  }

  public HuileDeTournesol fabriquerHuile(int nbFleurs){
    int nb = 0;
    int fleursDispo = nbFleurs;
    if(fleursDispo > Agriculteur.getNbFleursTot()){
      fleursDispo = Agriculteur.getNbFleursTot();
```

```
    }

    while (fleursDispo >=HuileDeTournesol.NBFLEURSPOUR1LDHUILE){

      nb++;
      fleursDispo -= HuileDeTournesol.NBFLEURSPOUR1LDHUILE;
    }

    Agriculteur.moinsNbFleursTot(HuileDeTournesol.NBFLEURSPOUR1LDHUILE*nb);

    System.out.println("On a fabrique " + nb + " bouteilles d'huile de tournesol");
    nbBouteillesHuile+=nb;
    if(nb>0) return new HuileDeTournesol(nb);
    else return null;
  }
}
```