Programmation Orientée Objet

Révision2 (solution 1)

Abdelwahab Naji **Enseignant chercheur**
Abdelwahab.naji@gmail.com

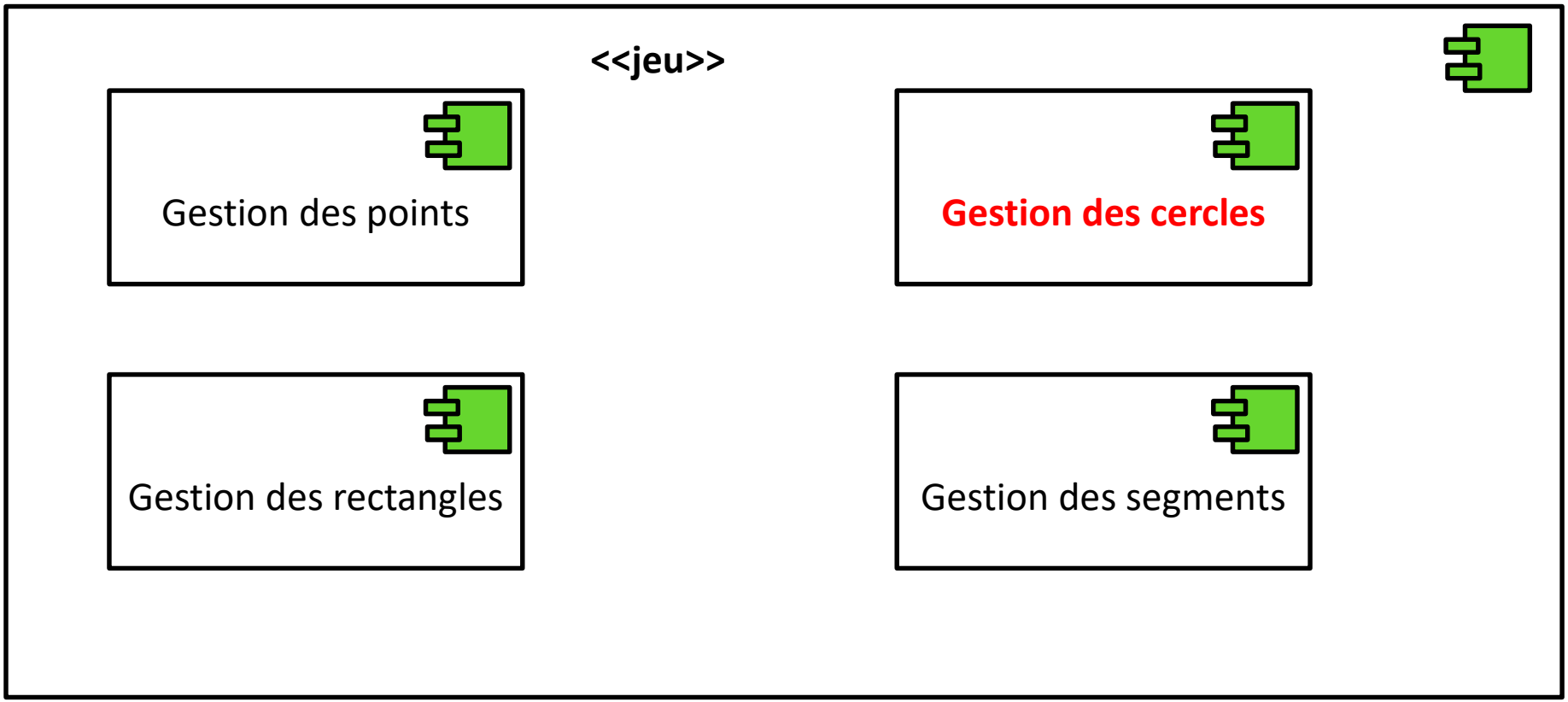
Application 1

- Dans l'application créée pour déplacer les points,
  - En utilisant votre propre LinkedList<T> générique, mémoriser les positions (x,y) d'un Point p pendant sont déplacement. (utiliser itérations)
  - Afficher toutes les positions
  - En utilisant votre propre LinkedList<T> générique, créer tous les segments composés de deux points successifs (deux déplacements)
    - Mémoriser l'ensemble des segments
    - Afficher les segments

- La proposition suivante contient une analyse, une conception et une implémentation de la partie en bleu

Application (jeu) : les modules de l'application

- L'application (jeu) à développer contient 4 modules
- Gestion des points, gestion des segments, gestion des rectangles, **gestion des cercles**
- La démarche à suivre:
  - Modéliser chaque module à part
  - Implémenter le module



Gestion des cercles: diagramme de cas d'utilisation

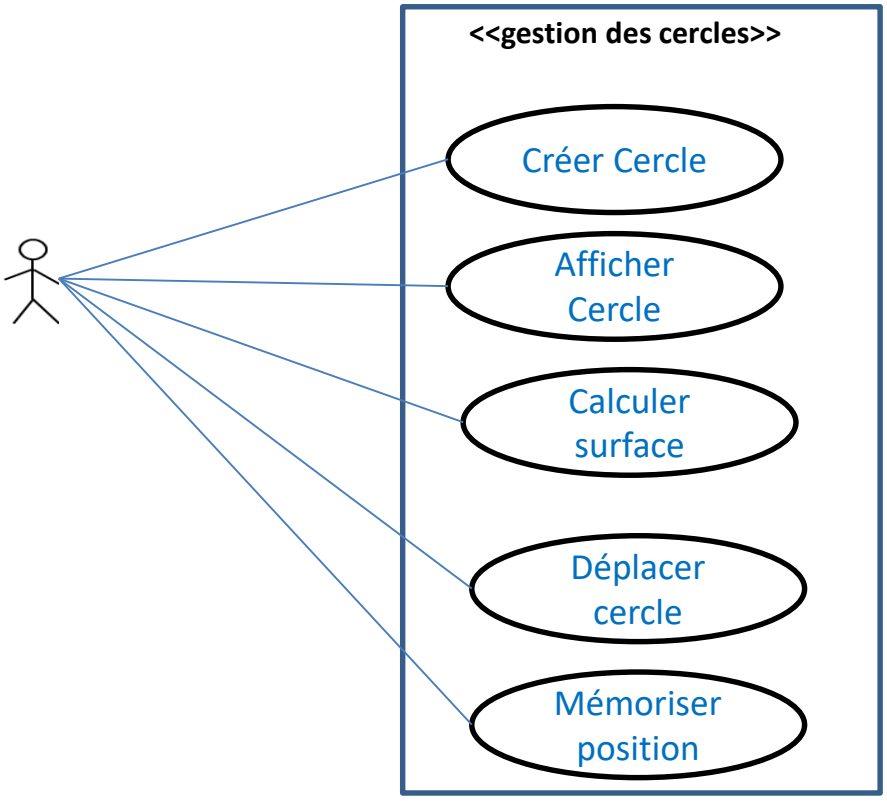


Diagramme de cas d'utilisation

Pour établir le diagramme de classe du domaine (d'analyse), pour chaque cas d'utilisation , il faut poser la question: quelles sont les données à gérer?

Gestion des cercles: Diagramme de classes d'analyse (V1)

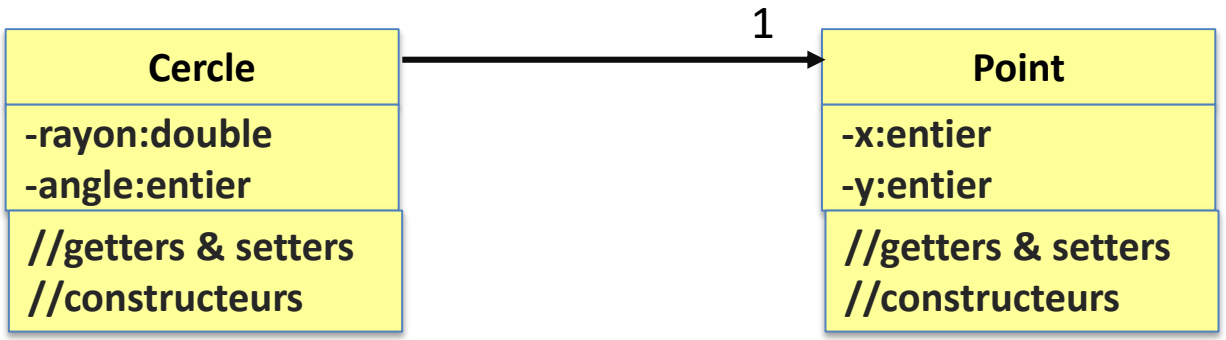


Diagramme de classes d'analyse V1

- **Le diagramme de classes d'analyse (V1) ne contient que les données à gérer**
- **La relation entre la classe Cercle et la classe Point est une association orientée**
- **Cette relation signifie: un objet cercle est associé à un et un seul point**

Gestion des cercles: architecture 2

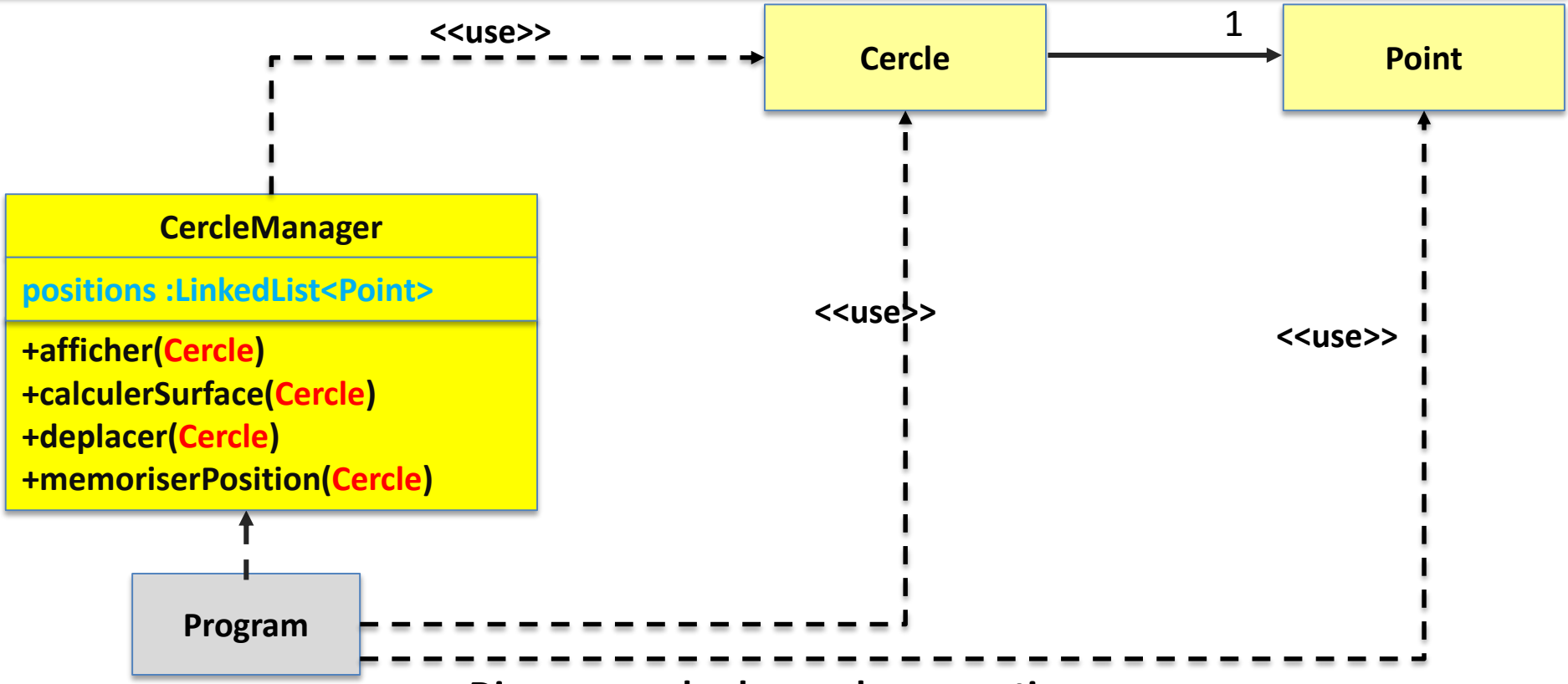
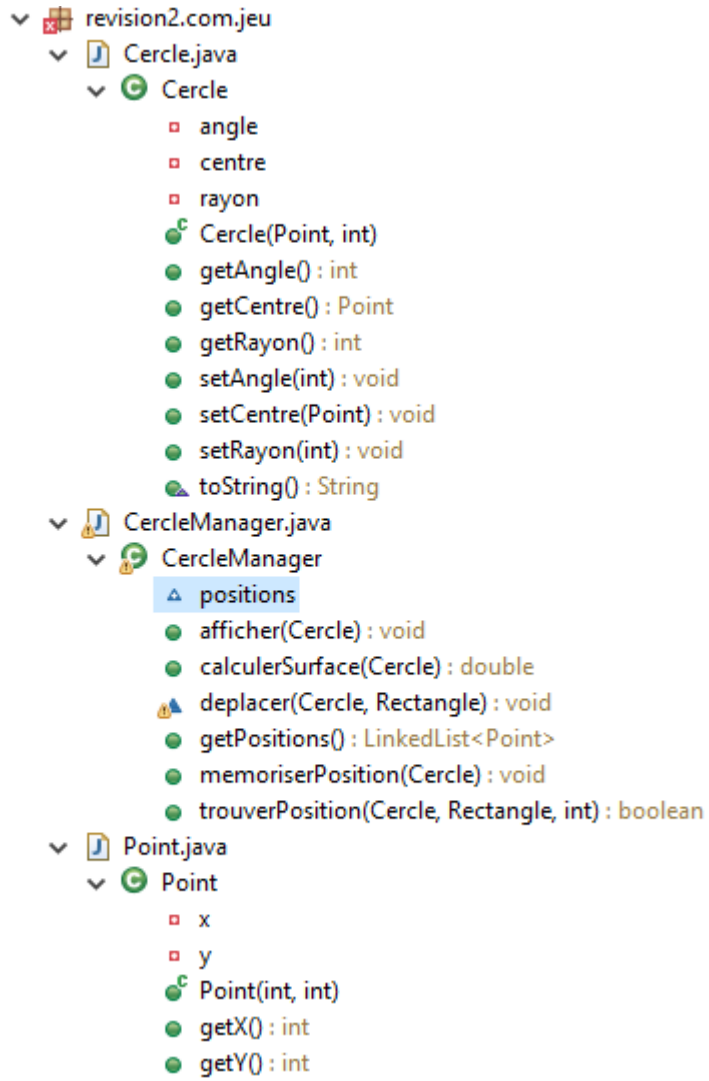


Diagramme de classes de conception

- Le diagramme de classes de conception définit les classes qui vont assurer la gestion
- CercleManager permet de:
  - Afficher un cercle
  - Calculer la surface d'un cercle
  - Déplacer un cercle
  - Mémoriser la position d'un cercle
- La classe Program définit le scénario d'exécution
- Proposer un diagramme de séquences

Architecture du projet



Code source: Point & Cercle

```
package revision2.com.jeu;
//créer le type de données Point
public class Point {
//structure & getters & setters & constructeurs
private int x;
private int y;
//getters & setters
public Point(int x, int y) {
this.x = x;
this.y = y;
}
@Override
public String toString() {
//(x,y)
return ("("+x+","+y+")");
}
}
```

```
package revision2.com.jeu;

public class Cercle {
private Point centre;
private int angle=180;
private int rayon;
//getters & setters
public Cercle(Point centre, int rayon) {
this.centre = centre;
this.rayon = rayon;
}
@Override
public String toString() {
return ("["+centre.toString()+ ", rayon="+rayon);
}
}
```

- Le diagramme de classes de conception définit les classes qui vont assurer la gestion
- CercleManager permet de:
  - Afficher un cercle
  - Calculer la surface d'un cercle
  - Déplacer un cercle
  - Mémoriser la position d'un cercle
- La classe Program définit le scénario d'exécution
- Proposer un diagramme de séquences

Code source: CercleManager

```
package revision2.com.jeu;
import collections.generic.LinkedList;
public class CercleManager {
LinkedList<Point> positions=new LinkedList<>();
public void afficher(Cercle c){
System.out.println(c.getCentre().toString() + "
"+c.getRayon()+ " Surface:"+calculerSurface(c));
}
public boolean trouverPosition(Cercle cercle,Rectangle
rect,int angle2) {
int x2 = (int) (cercle.getCentre().getX() + cercle.getRayon() *
Math.cos(Math.PI * angle2 / 180));
int y2 = (int) (cercle.getCentre().getY() + cercle.getRayon() *
Math.sin(Math.PI * angle2 / 180));
int xMin, yMin, xMax, yMax;
xMax = rect.getP2().getX();
xMin = rect.getP1().getX();
yMax = rect.getP2().getY();
yMin = rect.getP1().getY();
if ((x2 > xMax) || (x2 < xMin)|| (y2 < yMin) || (y2 > yMax)) {
return false;
}
cercle.setAngle(angle2);
cercle.getCentre().setX(x2);
cercle.getCentre().setY(y2);
return true;
}
```

```
public double calculerSurface(Cercle c){
double r=c.getRayon();
return Math.PI*r*r;
}
void deplacer(Cercle cercle, Rectangle rect){
int i=0;
boolean ok;
int angle2;
int x2,y2;//coordonnées nouveau Point
angle2=cercle.getAngle()+(int)(-45+Math.random()*90);
ok=trouverPosition(cercle,rect,angle2);
while(ok==false){
angle2=angle2+10;
ok=trouverPosition(cercle,rect,angle2);
}
}
public void memoriserPosition(Cercle cercle){
positions.add(cercle.getCentre());
}
public LinkedList<Point> getPositions(){
return positions;
}
}
```

- Le code en rouge ne mémorise pas les positions
- Pourquoi?
- Quelles sont les modifications à faire pour mémoriser les positions ?

LinkedList<T>

```
package collections.generic;
public class LinkedList<T>{
T val;
LinkedList<T> next;
public void add(T v){
LinkedList<T> nouv=new LinkedList();
nouv.val=v;
if(this.val==null){
this.val=v; return;
}
LinkedList<T> l=this;
while(l.next!=null) l=l.next;
l.next=nouv;
}

public void display(){
LinkedList<T> p=this;
while(p!=null){
System.out.println(p.val);
p=p.next;
}
}
}
```

- La classe générique LinkedList<T> appartient à un fichier LinkedList.java

Program

```
package revision2.com.jeu;
import collections.generic.LinkedList;
public class Program {
public static void main(String[] args) {
CercleManager cm=new CercleManager();
//----------créer un Point et l'afficher
Point point1=new Point(100,100);
Point point2=new Point(100,100);
System.out.println("Cercles----------");

Cercle cercle=new Cercle(point1, 20);
cm.afficher(cercle);

System.out.println("Rectangles----------");
Point p3=new Point(10,10);
Point p4=new Point(400,400);
Rectangle rect=new Rectangle(p3, p4, Color.red);
```

```
//deplacer  cercle dans rect
int i=0;
while(i<200){
cm.deplacer(cercle,rect);
cm.memoriserPosition(cercle);
cm.afficher(cercle);
i++;
}
//récupérer toutes les positions
System.out.println("liste des positions----------");
LinkedList<Point> pos=cm.getPositions();
pos.display();

}
}
```

Gestion des cercles: Diagramme de classe du domaine(V2)

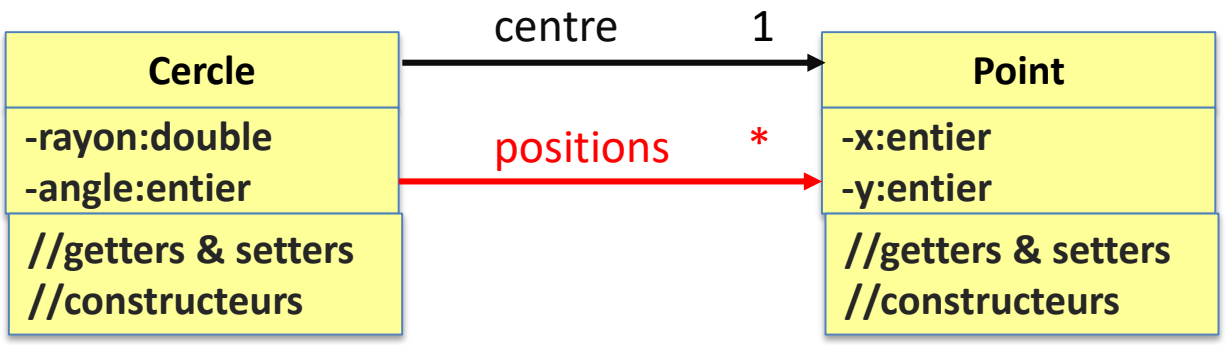


Diagramme de classes d'analyse

- **Le diagramme de classes d'analyse (V2) ne contient que les données à gérer**
- **La classe Cercle a deux associations avec la classe Point**
  - **Une association pour définir le centre du cercle**
  - **Une association pour définir les positions d'un cercle**
- **Comparer les deux modèles V1 et V2?**
- **Justifier que le modèle V2 est meilleur que V1**

Code source: Point & Cercle

```
package revision2.com.jeu;
//créer le type de données Point
public class Point {
//structure & getters & setters & constructeurs
private int x;
private int y;
//getters & setters

public Point(int x, int y) {
this.x = x;
this.y = y;
}
@Override
public String toString() {
//(x,y)
return ("("+x+","+y+")");
}
}
```

```
package revision2.com.jeu;

public class Cercle {
private Point centre;
LinkedList<Point> positions=new LinkedList<>();
private int angle=180;
private int rayon;
//getters & setters

public Cercle(Point centre, int rayon) {
this.centre = centre;
this.rayon = rayon;
}
@Override
public String toString() {
return ("["+centre.toString()+ ", rayon="+rayon);
}
}
```

- La classe Cercle contient aussi un ensemble de points pour mémoriser les positions
- Implémenter ce deuxième modèle selon l'architecture (2)