



Programmation Orientée Objet

Présentation orale du projet

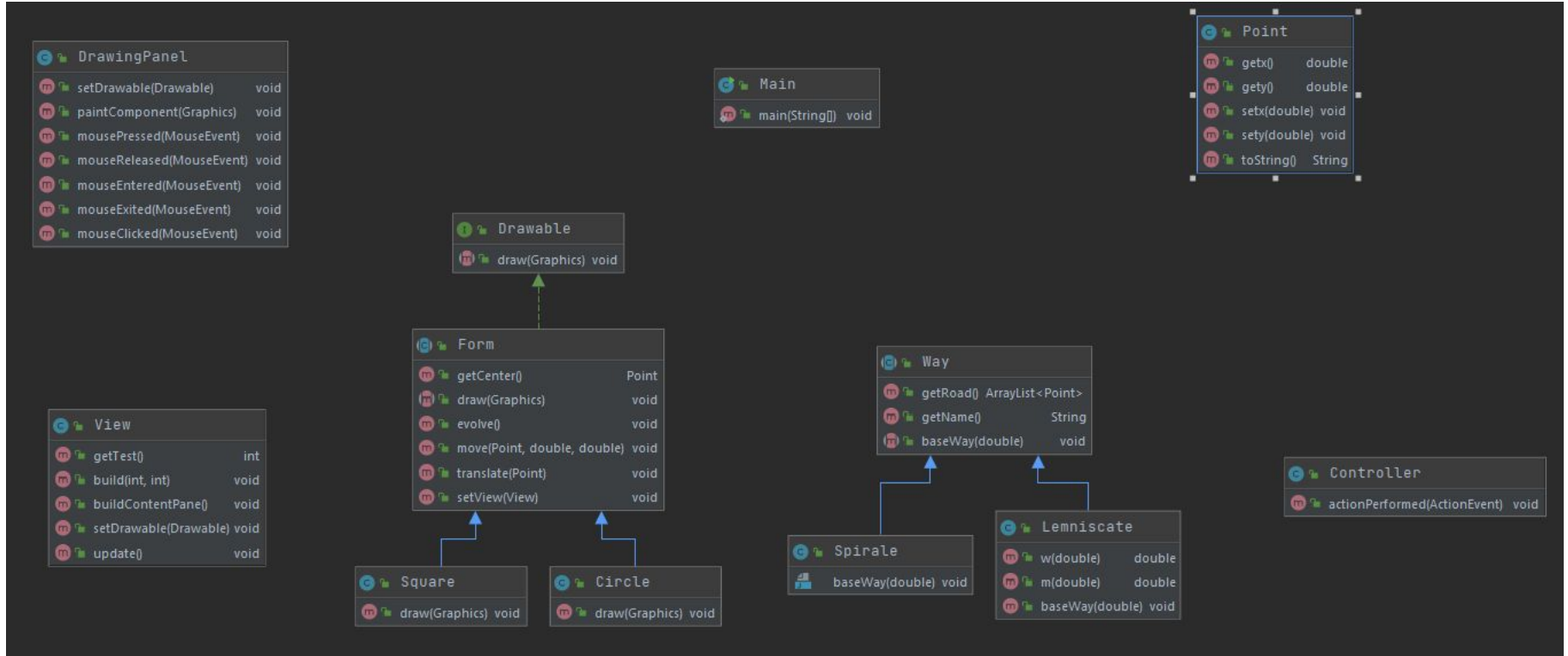
BILLAUD Maël
GATARD Maxime



Sommaire :

1. Introduction
2. Architecture des classes de forme et de chemin
3. Déplacement de la forme sur le chemin
4. Architecture graphique et interactions
5. État actuel du développement
6. Ouvertures

Introduction






Architecture des classes de forme et de chemin

Architecture des classes de forme et de chemin

```
public abstract class Form implements Drawable {  
  
    //Attribut  
  
    private Point center;  
    private double baseX, baseY;  
    private View view;  
  
    //Constructeur  
  
    public Form (Point c){  
        center = c;  
        baseX = center.getX();  
        baseY = center.getY();  
    }  
}
```

```
public class Square extends Form{  
  
    //Attributs :  
    private Point A, B, C, D; //Les quatre coins du carré où A est le côté en haut au gauche du carré  
    private float cote; //La longueur d'un côté  
  
    //Constructeur :  
    public Square(Point center, float cote){  
        super(center);  
        this.cote = cote;  
        A = new Point((int) center.getX()-(cote/2), (int) center.getY()+(cote/2));  
        B = new Point((int) center.getX()+(cote/2), (int) center.getY()+(cote/2));  
        C = new Point((int) center.getX()+(cote/2), (int) center.getY()-(cote/2));  
        D = new Point((int) center.getX()-(cote/2), (int) center.getY()-(cote/2));  
    }  
  
    //Methodes :  
  
    public void draw(Graphics g){  
        for (int i = 0; i < getCentersForm().size(); i+= 10) {  
            int x = (int) super.getCentersForm().get(i).getX(),  
                y = (int) super.getCentersForm().get(i).getY(),  
                c = (int) cote;  
            g.setColor(Color.black);  
            g.drawRect(x, y, c, c);  
        }  
    }  
}  
  
public class Circle extends Form{  
  
    //Attributs  
    private double rad; // rayon d'un cercle  
  
    // constructeur  
    public Circle(Point center, double r) {  
        super(center);  
        rad = r;  
    }  
  
    //Methodes  
  
    public void draw(Graphics g) {  
        for (int i = 0; i < getCentersForm().size(); i+= 10) {  
            int d = (int) (rad * 2),  
                x = (int) (super.getCentersForm().get(i).getX() - rad),  
                y = (int) (super.getCentersForm().get(i).getY() - rad);  
            g.setColor(Color.BLACK);  
            g.drawOval(x, y, d, d);  
        }  
    }  
}
```

Architecture des classes de forme et de chemin



```
public abstract class Way {  
    //Attributs  
    private String name;  
    private ArrayList<Point> road = new ArrayList<>();  
  
    //Constructeur  
    public Way(String name){  
        this.name = name;  
    }  
  
    //Methodes  
    public ArrayList<Point> getRoad(){  
        return road;  
    }  
  
    public String getName(){  
        return name;  
    }  
  
    public abstract void baseWay(double pas);  
}
```

Architecture des classes de forme et de chemin

```
public class Lemniscate extends Way{
    //Attributs
    private final double c;//Constante qui, une fois divisée par 2, correspond à la valeur maximale que peut prendre y
    private final double xMax;//Constante qui correspond à la valeur maximale que peut prendre x

    //Constructeur
    public Lemniscate(String name, double xMax){
        super(name);
        this.xMax = xMax;
        c = xMax/Math.sqrt(2);
    }

    //Methodes
    //Fonction qui s'occupe de la partie inferieur du Lemniscate (w)
    public double w(double x){
        return Math.sqrt( (-Math.pow(x, 2)-Math.pow(c, 2)) + c*Math.sqrt(4 * Math.pow(x, 2) + Math.pow(c, 2)));
    }

    //Fonction qui s'occupe de la partie superieur du Lemniscate (m)
    public double m(double x){
        return -Math.sqrt( (-Math.pow(x, 2)-Math.pow(c, 2)) + c*Math.sqrt(4 * Math.pow(x, 2) + Math.pow(c, 2)));
    }
}
```



Déplacement de la forme sur le chemin

Déplacement de la forme sur le chemin

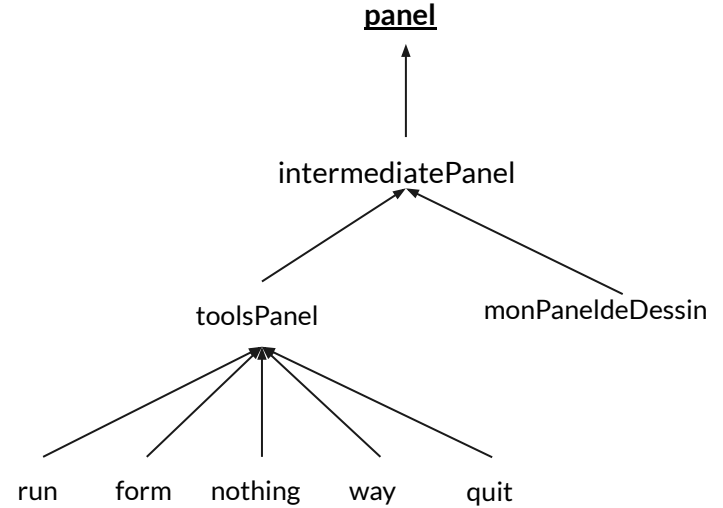
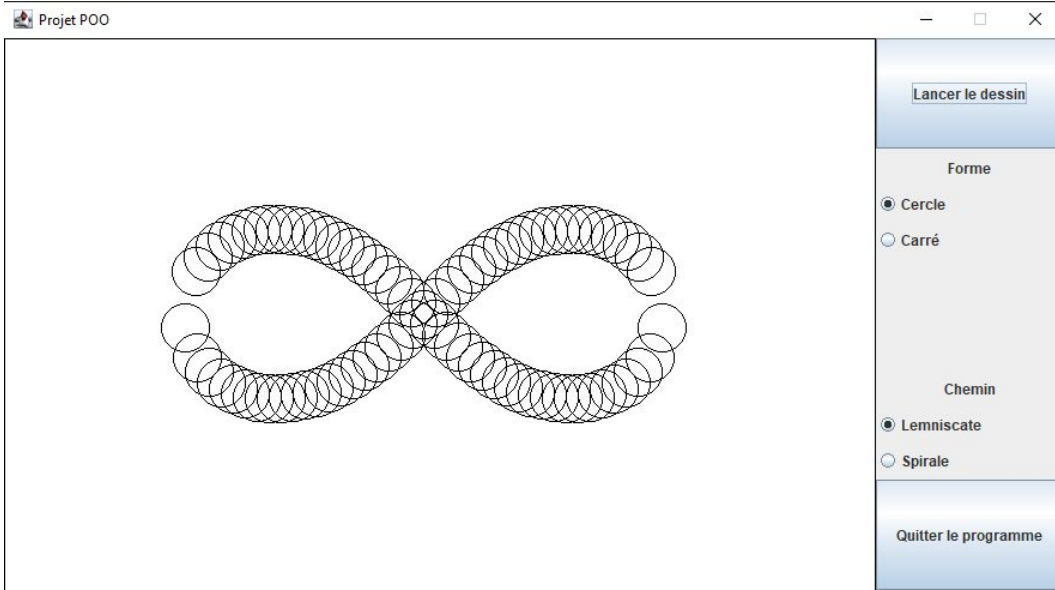
```
public void evolve(){
    Way test = new Lemniscate( name: "Lemniscate", xMax: 200);
    ((Lemniscate) test).baseWay( pas: 1);
    for (int i = 0; i < test.getRoad().size(); i++){
        translate(test.getRoad().get(i));
        try {
            Thread.sleep( millis: 5);
        } catch (Exception e){}
    }
}
```

```
public void baseWay(double pas){
    for (double i = 0; i < xMax; i += pas){
        getRoad().add(new Point(i, this.w(i)));
    }
    for (double i = (xMax-pas); i >= 0; i -= pas){
        getRoad().add(new Point(i, this.m(i)));
    }
    for (double i = 0; i > -xMax; i -= pas){
        getRoad().add(new Point(i, this.w(i)));
    }
    for (double i = (-xMax+pas); i <= 0; i += pas){
        getRoad().add(new Point(i, this.m(i)));
    }
}
```



Architecture graphique et interactions

Architecture graphique et interactions



```
private JPanel panel;  
private DrawingPanel monPaneldeDessin;  
private int test;
```

```
JPanel toolsPanel, form, way, nothing, intermediatePanel;  
JRadioButton square, circle, way1, way2;  
JButton run, quit;  
JLabel tool1, tool2;
```



Etat actuel du développement



Ouvertures

Ouvertures

```
public class Spirale extends Way{
    //Attributs
    private double x;
    private double y;

    //Constructeur
    public Spirale(String name, double x, double y){
        super(name);
        this.x = x;
        this.y = y;
    }

    public void baseWay(double pas){
    }
}
```

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class Controller implements ActionListener {
    //Attributs
    private Form forme;
    private Way chemin;

    //Constructeur
    public Controller(Form forme, Way chemin){
        this.forme = forme;
        this.chemin = chemin;
    }

    1 related problem
    public void mouseClicked(int x, int y){
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        forme.setColorNumber();
    }
}
```