

# Folder src

4 printable files

(file list disabled)

## src/DessinDisqueApp.java

```
/**
 * @author Lestiboudois Maxime & Parisod Nathan
 * @date 20/01/2025
 */

import javax.swing.*;

public class DessinDisqueApp {
    public static void main(String[] args) {
        SwingUtilities.invokeLater(DessinDisqueFrame::new);
    }
}
```

## src/DessinDisqueFrame.java

```
/**
 * @author Lestiboudois Maxime & Parisod Nathan
 * @date 20/01/2025
 */

import javax.swing.*;
import java.awt.*;

/**
 * Classe représentant la fenêtre principale de l'application.
 * Contient le panneau de dessin et les boutons "Clear" et "Quit".
 */
class DessinDisqueFrame extends JFrame {
    private final DessinPanel dessinPanel;

    /**
     * Constructeur de la fenêtre principale.
     * Initialise les composants graphiques et configure la disposition.
     */
    public DessinDisqueFrame() {
        setTitle("Application de Dessin de Disques");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        // Panel principal pour le dessin
        dessinPanel = new DessinPanel();
        add(dessinPanel, BorderLayout.CENTER);

        // Panel des boutons
        JPanel buttonPanel = new JPanel();
        JButton clearButton = new JButton("Clear");
        JButton quitButton = new JButton("Quit");
    }
}
```

```

        clearButton.addActionListener(e -> dessinPanel.clear());
        quitButton.addActionListener(e -> System.exit(0));

        buttonPanel.add(clearButton);
        buttonPanel.add(quitButton);
        add(buttonPanel, BorderLayout.SOUTH);

        pack();
        setVisible(true);
    }
}

```

## src/DessinPanel.java

```

/**
 * @author Lestiboudois Maxime & Parisod Nathan
 * @date 20/01/2025
 */

import javax.swing.*;
import java.awt.*;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.MouseMotionListener;
import java.util.LinkedList;
import java.util.ListIterator;

/**
 * Classe représentant le panneau de dessin.
 * Permet de dessiner, déplacer et supprimer des disques.
 */
class DessinPanel extends JPanel implements MouseListener, MouseMotionListener {
    private final LinkedList<Disque> disques = new LinkedList<>();
    private Disque disqueTemporaire = null;
    private Disque disqueDeplace = null; // Disque actuellement déplacé
    private Point lastMousePosition = null; // Dernière position de la souris pour le déplacement

    /**
     * Constructeur du panneau de dessin.
     * Configure les paramètres du panneau et enregistre les écouteurs de la souris.
     */
    public DessinPanel() {
        setBackground(Color.WHITE);
        setPreferredSize(new Dimension(800, 600));
        addMouseListener(this);
        addMouseMotionListener(this);
    }

    /**
     * Redéfinition de la méthode paintComponent pour dessiner les disques.
     *
     * @param g L'objet Graphics utilisé pour dessiner.
     */
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
    }
}

```

```

    // Dessiner tous les disques
    for (Disque disque : disques) {
        disque.dessiner(g);
    }

    // Dessiner le disque temporaire
    if (disqueTemporaire != null) {
        disqueTemporaire.dessiner(g);
    }
}

/**
 * Efface tous les disques du panneau de dessin.
 */
public void clear() {
    disques.clear();
    disqueTemporaire = null;
    repaint();
}

/**
 * Gère l'événement de clic de souris.
 *
 * @param e L'événement de la souris.
 */
@Override
public void mousePressed(MouseEvent e) {
    if (e.isShiftDown() && e.getButton() == MouseEvent.BUTTON1) { // SHIFT + clic gauche
        for (Disque disque : disques) {
            if (disque.contient(e.getPoint())) {
                disqueDeplace = disque;
                lastMousePosition = e.getPoint();
                break;
            }
        }
    } else if (e.getButton() == MouseEvent.BUTTON1) { // Bouton gauche pour créer un disque
        disqueTemporaire = new Disque(e.getPoint(), 0, disques.size());
    } else if (e.getButton() == MouseEvent.BUTTON3) { // Bouton droit pour effacer
        ListIterator<Disque> iterator = disques.listIterator(disques.size());
        while (iterator.hasPrevious()) {
            Disque disque = iterator.previous();
            if (disque.contient(e.getPoint())) {
                iterator.remove();
                repaint();
                break;
            }
        }
    }
}

/**
 * Gère l'événement de relâchement de souris.
 *
 * @param e L'événement de la souris.
 */
@Override
public void mouseReleased(MouseEvent e) {

```

```

        if (e.getButton() == MouseEvent.BUTTON1 && disqueTemporaire != null) { // Fin de création d'un disque
            disques.add(disqueTemporaire);
            disqueTemporaire = null;
            repaint();
        }
        disqueDeplace = null;
        lastMousePosition = null;
    }

    /**
     * Gère l'événement de glissement de souris.
     *
     * @param e L'événement de la souris.
     */
    @Override
    public void mouseDragged(MouseEvent e) {
        if (disqueTemporaire != null) { // Ajuster le rayon du disque temporaire
            int rayon = (int) disqueTemporaire.getCentre().distance(e.getPoint());
            disqueTemporaire.setRayon(rayon);
            repaint();
        } else if (disqueDeplace != null && lastMousePosition != null) { // Déplacer un disque sélectionné
            int dx = e.getX() - lastMousePosition.x;
            int dy = e.getY() - lastMousePosition.y;
            disqueDeplace.deplacer(dx, dy);
            lastMousePosition = e.getPoint();
            repaint();
        }
    }

    @Override
    public void mouseMoved(MouseEvent e) {}

    @Override
    public void mouseClicked(MouseEvent e) {}

    @Override
    public void mouseEntered(MouseEvent e) {}

    @Override
    public void mouseExited(MouseEvent e) {}
}

```

## src/Disque.java

```

/**
 * @author Lestiboudois Maxime & Parisod Nathan
 * @date 20/01/2025
 */

import java.awt.*;

/**
 * Classe représentant un disque.
 * Chaque disque est défini par son centre, son rayon et sa couleur.
 */
class Disque {
    private Point centre;

```

```

private int rayon;
private final int couleurIndex;
private static final Color[] COULEURS = {Color.RED, Color.GREEN, Color.BLUE, Color.YELLOW, Color.MAGENTA,
Color.PINK, Color.cyan, Color.DARK_GRAY};

/**
 * Constructeur pour créer un disque.
 *
 * @param centre      Le centre du disque.
 * @param rayon      Le rayon du disque.
 * @param couleurIndex L'index de la couleur dans le tableau prédéfini.
 */
public Disque(Point centre, int rayon, int couleurIndex) {
    this.centre = centre;
    this.rayon = rayon;
    this.couleurIndex = couleurIndex % COULEURS.length;
}

/**
 * Retourne le centre du disque.
 *
 * @return Le centre du disque.
 */
public Point getCentre() {
    return centre;
}

/**
 * Définit le rayon du disque.
 *
 * @param rayon Le nouveau rayon.
 */
public void setRayon(int rayon) {
    this.rayon = rayon;
}

/**
 * Déplace le disque en ajoutant un décalage.
 *
 * @param dx Décalage en X.
 * @param dy Décalage en Y.
 */
public void deplacer(int dx, int dy) {
    centre = new Point(centre.x + dx, centre.y + dy);
}

/**
 * Vérifie si un point est contenu dans le disque.
 *
 * @param p Le point à vérifier.
 * @return True si le point est dans le disque, sinon false.
 */
public boolean contient(Point p) {
    return centre.distance(p) <= rayon;
}

/**
 * Dessine le disque sur le panneau.
 *

```

```
* @param g L'objet Graphics pour dessiner.  
*/  
public void dessiner(Graphics g) {  
    g.setColor(COULEURS[couleurIndex]);  
    g.fillOval(centre.x - rayon, centre.y - rayon, rayon * 2, rayon * 2);  
}  
}
```