

```
public class MovingBall
{
    private double x;
    private double y;
    private int radius;
    private double xStep;
    private double yStep;

    public MovingBall(double pX, double pY, int pRadius, double pXStep, double pYStep)
    {
        x = pX;
        y = pY;
        radius = pRadius;
        xStep = pXStep;
        yStep = pYStep;
    }

    public double getX()
    {
        return x;
    }

    public double getY()
    {
        return y;
    }

    public int getRadius()
    {
        return radius;
    }

    public void setRadius(int radius)
    {
        this.radius = radius;
    }

    public double getXStep()
    {
        return xStep;
    }

    public double getYStep()
    {
        return yStep;
    }

    public void draw(Graphics g)
    {
        g.setColor(Color.BLUE);
        g.drawOval((int) x - radius, (int) y - radius, 2 * radius, 2 * radius);
    }

    public void doStep(int width, int height)
    {
        // Vérifier les limites puis déplacer
        if ((x + xStep + radius >= width) || (x + xStep - radius < 0))
            xStep = -xStep;
        if ((y + yStep + radius >= height) || (y + yStep - radius < 0))
            yStep = -yStep;

        x = x + xStep;
        y = y + yStep;
    }
}
```

```
public class MovingBalls
{
    private ArrayList<MovingBall> alBalls = new ArrayList<>();

    public void addBall(MovingBall pBall)
    {
        alBalls.add(pBall);
    }

    public void clear()
    {
        alBalls.clear();
    }

    public void changeRadius(int newRadius)
    {
        for (int i = 0; i < alBalls.size(); i++)
            alBalls.get(i).setRadius(newRadius);
    }

    public void draw(Graphics g)
    {
        for (int i = 0; i < alBalls.size(); i++)
            alBalls.get(i).draw(g);

        // Dessiner les lignes rouges entre les balles...
        for (int i = 0; i < alBalls.size(); i++)
        {
            MovingBall fromBall = alBalls.get(i);
            // Truc pour que la dernière balle soit reliée à la première
            MovingBall toBall = alBalls.get((i + 1) % alBalls.size());
            g.setColor(Color.RED);
            g.drawLine((int) fromBall.getX(), (int) fromBall.getY(), (int) toBall.getX(), (int) toBall.getY());
        }
    }

    public void doStep(int width, int height)
    {
        for (int i = 0; i < alBalls.size(); i++)
            alBalls.get(i).doStep(width, height);
    }
}
```

```
public class DrawPanel extends javax.swing.JPanel
{
    private MovingBalls balls = null;

    public DrawPanel()
    {
        initComponents();
    }

    public void setBalls(MovingBalls pBalls)
    {
        balls = pBalls;
    }

    @Override
    public void paintComponent(Graphics g)
    {
        // clean the background
        g.setColor(Color.WHITE);
        g.fillRect(0, 0, getWidth(), getHeight());

        // draw the balls
        if (balls != null)
            balls.draw(g);
    }
}
// Skipped: ... initComponents { ... }
// Variables declaration - do not modify//GEN-BEGIN:variables
// End of variables declaration//GEN-END:variables
}
```

```

public class MainFrame extends javax.swing.JFrame
{
    private MovingBalls balls = new MovingBalls();
    private Timer timer;

    public MainFrame()
    {
        initComponents();
        drawPanel.setBalls(balls);
        stepButton.setVisible(false);
        numberSlider.setEnabled(true);
        timer = new Timer(delaySlider.getValue(), stepButton.getActionListeners()[0]);
    }
    // Skipped: ... initComponents { ... }
    public double random(double min, double max)
    {
        return (Math.random() * (max - min)) + min;
    }

    private void startStopButtonActionPerformed(java.awt.event.ActionEvent evt) { //GEN-FIRST:event_startStopButtonActionPerformed
        if (!timer.isRunning())
        {
            balls.clear();
            for (int i = 0; i < numberSlider.getValue(); i++)
            {
                int radius = radiusSlider.getValue();
                double x = random(radius, drawPanel.getWidth() - radius);
                double y = random(radius, drawPanel.getHeight() - radius);
                double xStep = random(-5, 5);
                double yStep = random(-5, 5);

                MovingBall ball = new MovingBall(x, y, radius, xStep, yStep);
                balls.addBall(ball);
            }
            numberSlider.setEnabled(false);
            timer.setDelay(delaySlider.getValue());
            timer.start();
            startStopButton.setText("Stop");
        }
        else
        {
            numberSlider.setEnabled(true);
            timer.stop();
            startStopButton.setText("Start");
        }
        repaint();
    } //GEN-LAST:event_startStopButtonActionPerformed

    private void stepButtonActionPerformed(java.awt.event.ActionEvent evt) { //GEN-FIRST:event_stepButtonActionPerformed
        balls.doStep(drawPanel.getWidth(), drawPanel.getHeight());
        repaint();
    } //GEN-LAST:event_stepButtonActionPerformed

    private void radiusSliderStateChanged(javax.swing.event.ChangeEvent evt) //GEN-FIRST:event_radiusSliderStateChanged
    { //GEN-HEADEREND:event_radiusSliderStateChanged
        balls.changeRadius(radiusSlider.getValue());
        repaint();
    } //GEN-LAST:event_radiusSliderStateChanged

    private void delaySliderStateChanged(javax.swing.event.ChangeEvent evt) //GEN-FIRST:event_delaySliderStateChanged
    { //GEN-HEADEREND:event_delaySliderStateChanged
        timer.setDelay(delaySlider.getValue());
        repaint();
    } //GEN-LAST:event_delaySliderStateChanged
    // Skipped: ... Look & Feel
    // Variables declaration - do not modify //GEN-BEGIN:variables
    private javax.swing.JSlider delaySlider;
    private DrawPanel drawPanel;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JSlider numberSlider;
    private javax.swing.JSlider radiusSlider;
    private javax.swing.JButton startStopButton;
    private javax.swing.JButton stepButton;
    // End of variables declaration //GEN-END:variables
}

```