```java
import java.awt.Color;
import java.awt.Graphics;
public class MovingBall
{
    private double x;
    private double y;
    private int radius;
    private double xStep;
    private double yStep;

    public MovingBall(double x, double y, int radius, double xStep, double yStep)
    {
        this.x = x;
        this.y = y;
        this.radius = radius;
        this.xStep = xStep;
        this.yStep = yStep;
    }

    public double getX()
    {
        return x;
    }

    public double getY()
    {
        return y;
    }

    public int getRadius()
    {
        return 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)
    {
        if (x + radius + xStep >= width)
        {
            x = width - radius - 1;
            xStep = -xStep;
        }
        else if (x - radius + xStep < 0)
        {
            x = radius;
            xStep = -xStep;
        }
        else
            x = x + xStep;

        if (y + radius + yStep >= height)
        {
            y = height - radius - 1;
            yStep = -yStep;
        }
        else if (y - radius + yStep < 0)
        {
            y = radius;
            yStep = -yStep;
        }
        else
            y += yStep;  // même chose que: y = y + yStep;
    }
}
```

```java
import java.awt.Color;
import java.awt.Graphics;
import java.util.ArrayList;
public class MovingBalls
{
    private ArrayList<MovingBall> alMovingBalls = new ArrayList<>();

    public void addBall(MovingBall movingBall)
    {
        alMovingBalls.add(movingBall);
    }

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

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

        // Dessiner les lignes rouges entre le centre des balles...
        for (int i = 0; i < alMovingBalls.size(); i++)
        {
            MovingBall fromBall = alMovingBalls.get(i);

            // Truc pour que la dernière balle soit reliée automatiquement à la première
            // un "if" est autorisé!
            MovingBall toBall = alMovingBalls.get((i + 1) % alMovingBalls.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 < alMovingBalls.size(); i++)
            alMovingBalls.get(i).doStep(width, height);
    }
}
```

```java
import java.awt.Color;
import java.awt.Graphics;
public class DrawPanel extends javax.swing.JPanel
{
    private MovingBalls movingBalls = null;

    public DrawPanel()
    {
        initComponents();
    }

    public void setMovingBalls(MovingBalls movingBalls)
    {
        this.movingBalls = movingBalls;
    }

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

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

```java
import javax.swing.Timer;
public class MainFrame extends javax.swing.JFrame
{
    private MovingBalls movingBalls = null;

    private int delay = 10;
    private Timer timer = null;

    public MainFrame()
    {
        initComponents();

        movingBalls = new MovingBalls();
        drawPanel.setMovingBalls(movingBalls);
        timer = new Timer(delay, stepButton.getActionListeners()[0]);
        stepButton.setVisible(false);
    }
// Skipped: ... initComponents { ... }
    private void startStopButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_startStopButtonActionPerformed
        if (timer.isRunning())
        {
            timer.stop();
            startStopButton.setText("Start");
        }
        else
        {
            movingBalls.clear();
            int r = 30;             // rayon: à modifier pour tester
            int nbrBalls = 30;      // nombre de balles: à modifier pour tester
            for (int i = 0; i < nbrBalls; i++)
            {
                int xStep = (int) (Math.random() * (5-(-5)+1)) -5;  // intervalle: [-5,+5]
                int yStep = (int) (Math.random() * (5-(-5)+1)) -5;  // intervalle: [-5,+5]

                int w = drawPanel.getWidth();
                int h = drawPanel.getHeight();

                int x = (int) (Math.random() * ((w - r - 1) - r + 1)) + r;  // intervalle: [r, w-r-1]
                int y = (int) (Math.random() * ((h - r - 1) - r + 1)) + r;  // intervalle: [r, h-r-1]

                MovingBall movingBall = new MovingBall(x, y, r, xStep, yStep);
                movingBalls.addBall(movingBall);
            }
            timer.start();
            startStopButton.setText("Stop");
        }
        repaint();
    }//GEN-LAST:event_startStopButtonActionPerformed

    private void stepButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_stepButtonActionPerformed
        movingBalls.doStep(drawPanel.getWidth(), drawPanel.getHeight());
        repaint();
    }//GEN-LAST:event_stepButtonActionPerformed
// Skipped: ... Look & Feel
    // Variables declaration - do not modify//GEN-BEGIN:variables
    private DrawPanel drawPanel;
    private javax.swing.JButton startStopButton;
    private javax.swing.JButton stepButton;
    // End of variables declaration//GEN-END:variables
}
```