

```
public class MovingBall
{
    /*
     * Version 2:
     * Solution en ajoutant la classe Emoticon qui hérite de MovingBall et qui dessine le partie commune à tous les emoticons,
     * les 4 emoticons héritent de la classe Emoticon
     */

    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 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 Emoticon extends MovingBall
{
    public Emoticon(double pX, double pY, int pRadius, double pXStep, double pYStep)
    {
        super(pX, pY, pRadius, pXStep, pYStep);
    }

    public void draw(Graphics g)
    {
        // Accès rapide
        double x = getX();
        double y = getY();
        int r = getRadius();

        // Dessine le disque jaune et les yeux
        g.setColor(Color.YELLOW);
        g.fillOval((int) x - r, (int) y - r, 2 * r, 2 * r);

        // Oeil gauche / droit - partie blanche
        g.setColor(Color.WHITE);
        g.fillOval((int) (x - r * 5 / 8), (int) (y - r * 4 / 8), r / 2, r / 2);
        g.fillOval((int) (x + r * 1 / 8), (int) (y - r * 4 / 8), r / 2, r / 2);

        // Oeil gauche / droit - contour gris
        g.setColor(Color.LIGHT_GRAY);
        g.drawOval((int) (x - r * 5 / 8), (int) (y - r * 4 / 8), r / 2, r / 2);
        g.drawOval((int) (x + r * 1 / 8), (int) (y - r * 4 / 8), r / 2, r / 2);

        // Oeil gauche / droit - iris en bleu
        g.setColor(Color.BLUE);
        g.fillOval((int) (x - r * 10 / 16 + r * 2 / 16), (int) (y - r * 8 / 16 + r * 3 / 16), (int) (r / 4), (int) (r / 4));
        g.fillOval((int) (x + r * 2 / 16 + r * 2 / 16), (int) (y - r * 8 / 16 + r * 3 / 16), (int) (r / 4), (int) (r / 4));
    }
}
```

```
public class EmoBigSmile extends Emoticon
{
    public EmoBigSmile(double pX, double pY, int pRadius, double pXStep, double pYStep)
    {
        super(pX, pY, pRadius, pXStep, pYStep);
    }

    public void draw(Graphics g)
    {
        // Dessine le cercle et les yeux
        super.draw(g);

        // Accès rapide
        double x = getX();
        double y = getY();
        int r = getRadius();

        // Dessiner la bouche
        g.setColor(Color.BLACK);
        g.fillArc((int) (x - r * 0.5), (int) (y - r * 0.25), r, r, 0, -180);
    }
}
```

```
public class EmoSad extends Emoticon
{
    public EmoSad(double pX, double pY, int pRadius, double pXStep, double pYStep)
    {
        super(pX, pY, pRadius, pXStep, pYStep);
    }

    @Override
    public void draw(Graphics g)
    {
        // Dessine le cercle et les yeux
        super.draw(g);

        // Accès rapide
        double x = getX();
        double y = getY();
        int r = getRadius();

        // Dessiner la bouche
        g.setColor(Color.BLACK);
        g.drawArc((int) (x - r * 0.5), (int) (y + r * 0.25), r, r, 0, 180);
    }
}
```

```
public class EmoSmile extends Emoticon
{
    public EmoSmile(double pX, double pY, int pRadius, double pXStep, double pYStep)
    {
        super(pX, pY, pRadius, pXStep, pYStep);
    }

    public void draw(Graphics g)
    {
        // Dessine le cercle et les yeux
        super.draw(g);

        // Accès rapide
        double x = getX();
        double y = getY();
        int r = getRadius();

        // Dessiner la bouche
        g.setColor(Color.BLACK);
        g.drawArc((int) (x - r * 0.5), (int) (y - r * 0.25), r, r, 0, -180);
    }
}
```

```
public class EmoSurprised extends Emoticon
{
    public EmoSurprised(double pX, double pY, int pRadius, double pXStep, double pYStep)
    {
        super(pX, pY, pRadius, pXStep, pYStep);
    }

    public void draw(Graphics g)
    {
        // Dessine le cercle et les yeux
        super.draw(g);

        // Accès rapide
        double x = getX();
        double y = getY();
        int r = getRadius();

        // Dessiner la bouche
        g.setColor(Color.BLACK);
        g.fillOval((int) (x - r * 0.25), (int) (y + r * 0.25), (int) (r * 0.5), (int) (r * 0.5));
    }
}
```

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

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

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

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

    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 int delay = 10;
    private Timer timer;

    public MainFrame()
    {
        initComponents();
        drawPanel.setBalls(balls);
        timer = new Timer(delay, stepButton.getActionListeners()[0]);
        stepButton.setVisible(false);
    }
    // Skipped: ... initComponents { ... }
    public double andom(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 < 30; i++)
            {
                int radius = 30;
                double x = andom(radius, drawPanel.getWidth() - radius);
                double y = andom(radius, drawPanel.getHeight() - radius);
                double xStep = andom(-5, 5);
                double yStep = andom(-5, 5);

                int kind = (int) andom(0, 5); // valeurs: 0, 1, 2, 3, 4
                MovingBall ball = null;

                if (kind == 0)
                    ball = new EmoSmile(x, y, radius, xStep, yStep);
                else if (kind == 1)
                    ball = new EmoBigSmile(x, y, radius, xStep, yStep);
                else if (kind == 2)
                    ball = new EmoSad(x, y, radius, xStep, yStep);
                else if (kind == 3)
                    ball = new EmoSurprised(x, y, radius, xStep, yStep);
                else
                    ball = new MovingBall(x, y, radius, xStep, yStep);

                balls.addBall(ball);
            }
            timer.start();
            startStopButton.setText("Stop");
        }
        else
        {
            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
    // 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
}

```