Class PrizePanel

```java
/**
 * Class PrizePanel draws a ball and polkadot counting how many times they overlap
 *
 * @author  Nathan Chen
 * @author  Benjamin Tu
 * @period  2
 * @version 10-23-18
 * @teacher Coglianese
 */
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.image.*;
public class PrizePanel extends JPanel
{
    private static final int FRAME = 400;
    private static final Color BACKGROUND = new Color(204, 204, 204);
    private BufferedImage myImage;
    private Graphics myBuffer;
    private Ball myBall;
    private Polkadot myPDot;
    private Timer myTimer;
    private int hits = 0;
    /**
     * Constructor with no arguments that draws Ball and Polkadot and refreshes them
     */
    public PrizePanel()
    {
        myImage =  new BufferedImage(FRAME, FRAME, BufferedImage.TYPE_INT_RGB);
        myBuffer = myImage.getGraphics();
        myBuffer.setColor(BACKGROUND);
        myBuffer.fillRect(0, 0, FRAME,FRAME);
        int xPos = (int)(Math.random()*(FRAME-100) + 50);
        int yPos = (int)(Math.random()*(FRAME-100)+ 50);
        myBall = new Ball(xPos, yPos, 50, Color.BLACK);
        myPDot = new Polkadot(xPos, yPos, 25, Color.RED);
        myPDot.jump(FRAME,FRAME);
        myTimer = new Timer(5, new Listener());
        myTimer.start();
    }

    /**
     * paintComponent used draw image
     *
     * @param   g   graphics panel where thisn are drawn
     */
    public void paintComponent(Graphics g)
    {
        g.drawImage(myImage, 0, 0, getWidth(), getHeight(), null);
    }
    /**
     * Class Listener redraws things
     */
    private class Listener implements ActionListener
    {
        /**
         * actionPerformed is what is done every refresh
         *
         * @params  e   ActionEvent not used
         */
        public void actionPerformed(ActionEvent e)
        {
            myBuffer.setColor(BACKGROUND);
            myBuffer.fillRect(0,0,FRAME,FRAME);

            myBall.move(FRAME, FRAME);
            collide(myBall, myPDot);

            myBall.draw(myBuffer);
            myPDot.draw(myBuffer);

            myBuffer.setColor(Color.BLACK);
            myBuffer.setFont(new Font("Monospaced", Font.BOLD, 24));
            myBuffer.drawString("Count: " + hits, FRAME - 150, 25);
            repaint();
        }
    }
    /**
     * collide adds to hits if distance between balls is less than 75
     *
     * @param   ballIn  Ball class being checked
     * @param   pDot    Polkadot being checked
     */
    private void collide(Ball ballIn, Polkadot pDot)
    {
        double d = distance(ballIn.getX(),ballIn.getY(),pDot.getX(),pDot.
getY());
        if (d<=75)
            hits++;
    }

    /**
     * Distance calculator between "object" 1 and 2 given x and y coordin
ates of each
     *
     * @param   x1  X coordinate of "object" 1
     * @param   y1  Y coordinate of "object" 1
     * @param   x2  X coordinate of "object" 2
     * @param   y2  Y coordinate of "object" 2
     */
    private double distance(double x1, double y1, double x2, double y2)
    {
        return Math.sqrt(Math.pow(x2-x1,2) + Math.pow(y2-y1,2));
    }
}
```