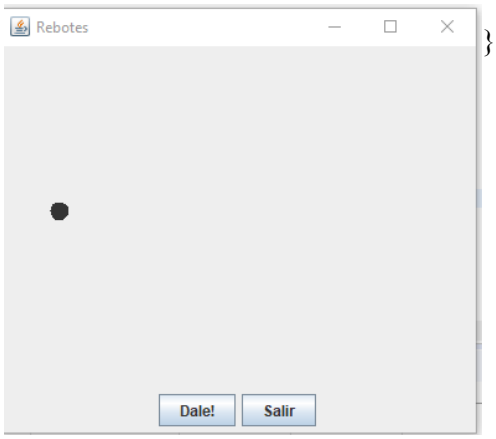
Ejercicio de Rebote de pelota.

```
package usoHilo;

import java.awt.geom.*;

import javax.swing.*;

import java.util.*;
import java.awt.*;
import java.awt.event.*;


public class UsoHilo {

	public static void main (String[] args) {

		JFrame marco=new MarcoRebote();

		marco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		marco.setVisible(true);
	}

}

//Movimiento de la pelota ====================================================

class Pelota{

	private static final int TAMX=15;
	private static final int TAMY=15;

	private double x=0;
	private double y=0;
	private double dx=1;
	private double dy=1;

	//Mueve la pelota invirtiendo posición si choca con los limites
	public void mueve_pelota(Rectangle2D limites) {

		x+=dx;

		y+=dy;

		if(x<limites.getMinX()) {
			dx=-dx;
		}

		if(x+TAMX >= limites.getMaxX()) {
			x=limites.getMaxX() - TAMX;
			dx=-dx;
```

```
                }
                if(y<limites.getMinY()) {
                        y=limites.getMinY();
                        dy=-dy;
                }

                if(y + TAMY >= limites.getMaxY()) {
                        y=limites.getMaxY() - TAMY;
                        dy=-dy;
                }
        }

        //Forma de la pelota en su posición inicial

        public Ellipse2D getShape() {
                return new Ellipse2D.Double(x,y,TAMX,TAMY);
        }
}

//Lámina que dibuja las pelotas

class LaminaPelota extends JPanel{

        private ArrayList<Pelota> pelotas=new ArrayList<Pelota>();

        //Añadimos pelota a la lámina

        public void add(Pelota b) {
                pelotas.add(b);
        }

        public void paintComponent(Graphics g) {

                super.paintComponent(g);
                Graphics2D g2=(Graphics2D)g;

                for(Pelota b: pelotas) {
                        g2.fill(b.getShape());
                }
        }


}
//Marco con lámina y botones=============================================0000

class MarcoRebote extends JFrame{

        private LaminaPelota lamina;

        public MarcoRebote() {
                setBounds(600,300,400,350);
                setTitle ("Rebotes");
```

```
		lamina=new LaminaPelota();
		add(lamina, BorderLayout.CENTER);
		JPanel laminaBotones=new JPanel();
		ponerBoton(laminaBotones, "Dale!", new ActionListener(){
			public void actionPerformed(ActionEvent evento) {
				comienza_el_juego();
			}
		});

		ponerBoton(laminaBotones, "Salir", new ActionListener() {
			public void actionPerformed(ActionEvent evento) {
				System.exit(0);

			}
		});

		add(laminaBotones, BorderLayout.SOUTH);
}

//Ponemos botones

public void ponerBoton (Container c, String titulo, ActionListener oyente) {
		JButton boton=new JButton(titulo);
		c.add(boton);
		boton.addActionListener(oyente);
}

//Añade pelota y la bota 1000 veces

public void comienza_el_juego () {

		Pelota pelota=new Pelota();
		lamina.add(pelota);

		for(int i=1; i<=3000; i++) {
			pelota.mueve_pelota(lamina.getBounds());
			lamina.paint(lamina.getGraphics());
		}
}
```



}

Ejercicio 1

La pelota no se ve en movimiento por la velocidad con que se ejecuta el bucle for

```
for(int i=1; i<=3000; i++) {
                    pelota.mueve_pelota(lamina.getBounds());
                    lamina.paint(lamina.getGraphics());
            }
```

Introducir una pausa en la ejecución del hilo. Este programa en definitiva es un hilo.

Para hacer una pausa en la ejecución de un programa usamos el métod “sleep()”

Ejercicio 2

Este programa solo ejecuta un hilo al mismo tiempo, si mientra se esta ejecutando la aplicación le damos nuevamente al botón de inicio, la segunda pelota no saldrá hasta que la primera no pare, es entonces cuando se ejecuta el siguiente hilo.

Si mientras la pelota está en movimiento le damos al botón de salir, no nos deja, porque eso sería una nueva tarea, es decir , un nuevo hilo de ejecución. Recordemos que este programa solo permite la ejecución de un hilo al mismo tiempo.

Convertir esta aplicación en multitarea, es decir, cada vez que se pulse en el botón de inicio salga una nueva pelota (nuevo hilo de ejecución) y que en cualquier momento que pulsemos sobre salir, el programa finalice (otro nuevo hilo).

Para convertir esta aplicación en multitarea hay que seguir los siguientes pasos:

1. Crear clase que implemente la interfaz Runnable (método run())
2. Escribir el código de la tarea dentro del método run
3. Instanciar la clase creada y almacenar la instancia en variable de tipo Runnable
4. Crear instancia de la clase Thread pasando como parámetro al constructor de Thread el objeto Runnable anterior.
5. Poner en marcha el hilo de ejecución con el método start() de la clase Thread.
6. Instanciar la clase creada y almacenar la instancia en variable de tipo Runnable
7. Crear instancia de la clase Thread pasando como parámetro al constructor de Thread el objeto Runnable anterior.
8. Poner en marcha el hilo de ejecución con el método start() de la clase Thread.