



---

# REPORTE – PRACTICA 2

---

Programación concurrente y paralela



14 DE FEBRERO DE 2025

ALUMNO: ENRIQUE HERNANDEZ LUNA  
DOCENTE: DRA CARMEN CERON GARCIA

## Introducción:

En esta práctica números 2 se realizaron los siguientes programas:

1. Realiza un programa de dos hilos para imprimir los números pares e impares usando los cambios de estado de los hilos, el hilo A imprime impares y el hilo B los pares y al final imprime la suma de los pares e impares.
2. Realizar de forma sincronizada y comunicada de forma alternada la cooperación de dos hilos para identificar que números son pares y que números son impares de una serie de 50 y escribe la suma de los pares e impares. Los números se generan de manera aleatoria.
3. Realizar de forma sincronizada y comunicada de forma alternada la cooperación de dos hilos para identificar que números son positivos y que números son negativos (forma aleatoria) de una serie de -100 a 100 y escribe la suma de cada hilo.

```

1. package Practica2;
2.
3. class Jardin {
4.     public synchronized void sembrar(String nombre) {
5.         for (int i = 1; i <= 10; i++) {
6.             System.out.println(nombre + " esta sembrando el arbol " + i);
7.
8.             try {
9.                 Thread.sleep(500);
10.            } catch (InterruptedException e) {
11.                e.printStackTrace();
12.            }
13.
14.            System.out.println(nombre + " ha terminado de sembrar el arbol " + i);
15.        }
16.    }
17. }
18.
19. class Hermano implements Runnable {
20.     private final Jardin jardin;
21.     private final String nombre;
22.
23.     public Hermano(Jardin jardin, String nombre) {
24.         this.jardin = jardin;
25.         this.nombre = nombre;
26.     }
27.
28.     @Override
29.     public void run() {
30.         jardin.sembrar(nombre);
31.     }
32. }
33.
34. public class HermanoV1Runnable {
35.     public static void main(String[] args) {
36.         Jardin jardin = new Jardin();
37.         Hermano hermano1 = new Hermano(jardin, "Hermano 1");
38.         Hermano hermano2 = new Hermano(jardin, "Hermano 2");
39.
40.         Thread t1 = new Thread(hermano1);
41.         Thread t2 = new Thread(hermano2);
42.
43.         t1.start();
44.         try {
45.             t1.join();
46.         } catch (InterruptedException e) {
47.             e.printStackTrace();
48.         }
49.
50.         t2.start();
51.         try {
52.             t2.join();
53.         } catch (InterruptedException e) {
54.             e.printStackTrace();
55.         }
56.
57.         System.out.println("Ambos hermanos han terminado de sembrar los árboles");
58.     }
59. }
60.

```

```

1. package Practica2;
2.
3. class Jardin {
4.     public synchronized void sembrar(String nombre) {
5.         for (int i = 1; i <= 10; i++) {
6.             System.out.println(nombre + " esta sembrando el arbol " + i);
7.
8.             try {
9.                 Thread.sleep(500);
10.            } catch (InterruptedException e) {
11.                e.printStackTrace();
12.            }
13.
14.            System.out.println(nombre + " ha terminado de sembrar el arbol " + i);
15.        }
16.    }
17. }
18.
19. class Hermano extends Thread {
20.     private final Jardin jardin;
21.
22.     public Hermano(Jardin jardin, String nombre) {
23.         super(nombre);
24.         this.jardin = jardin;
25.     }
26.
27.     @Override
28.     public void run() {
29.         jardin.sembrar(getName());
30.     }
31. }
32.
33. public class HermanoV1Thread {
34.     public static void main(String[] args) {
35.         Jardin jardin = new Jardin();
36.         Hermano hermano1 = new Hermano(jardin, "Hermano 1");
37.         Hermano hermano2 = new Hermano(jardin, "Hermano 2");
38.
39.         hermano1.start();
40.
41.         try {
42.             hermano1.join();
43.         } catch (InterruptedException e) {
44.             e.printStackTrace();
45.         }
46.
47.         hermano2.start();
48.
49.         try {
50.             hermano2.join();
51.         } catch (InterruptedException e) {
52.             e.printStackTrace();
53.         }
54.
55.         System.out.println("Ambos hermanos han terminado de sembrar los árboles");
56.     }
57. }
58.

```

```

1. package Practica2;
2.
3. import javax.swing.*;
4. import java.awt.*;
5.
6. class Jardin {
7.     private final JTextArea textArea;
8.
9.     public Jardin(JTextArea textArea) {
10.         this.textArea = textArea;
11.     }
12.
13.     public synchronized void sembrar(String nombre) {
14.         for (int i = 1; i <= 10; i++) {
15.             textArea.append(nombre + " está sembrando el árbol " + i + "\n");
16.
17.             try {
18.                 Thread.sleep(500);
19.             } catch (InterruptedException e) {
20.                 e.printStackTrace();
21.             }
22.
23.             textArea.append(nombre + " ha terminado de sembrar el árbol " + i + "\n");
24.         }
25.     }
26. }
27.
28. class Hermano extends Thread {
29.     private final Jardin jardin;
30.
31.     public Hermano(Jardin jardin, String nombre) {
32.         super(nombre);
33.         this.jardin = jardin;
34.     }
35.
36.     @Override
37.     public void run() {
38.         jardin.sembrar(getName());
39.     }
40. }
41.
42. public class HermanosV1ThreadGUI {
43.     public static void main(String[] args) {
44.         JFrame frame = new JFrame("Hermanos Sembrando Árboles");
45.         JTextArea textArea = new JTextArea(20, 40);
46.         textArea.setEditable(false);
47.         JScrollPane scrollPane = new JScrollPane(textArea);
48.         frame.add(scrollPane, BorderLayout.CENTER);
49.         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
50.         frame.pack();
51.         frame.setVisible(true);
52.
53.         Jardin jardin = new Jardin(textArea);
54.         Hermano hermano1 = new Hermano(jardin, "Hermano 1");
55.         Hermano hermano2 = new Hermano(jardin, "Hermano 2");
56.
57.         hermano1.start();
58.         try {
59.             hermano1.join();
60.         } catch (InterruptedException e) {
61.             e.printStackTrace();
62.         }
63.
64.         hermano2.start();
65.         try {

```

```

66.         hermano2.join();
67.     } catch (InterruptedException e) {
68.         e.printStackTrace();
69.     }
70.
71.     textArea.append("Ambos hermanos han terminado de sembrar los árboles\n");
72. }
73. }
74.

```

```

1. package Practica2;
2.
3. import javax.swing.*;
4. import java.awt.*;
5.
6. class Jardin {
7.     private boolean turnoHermano1 = true;
8.     private final JTextArea textArea;
9.
10.    public Jardin(JTextArea textArea) {
11.        this.textArea = textArea;
12.    }
13.
14.    public synchronized void sembrar(String nombre, boolean esHermano1) {
15.        for (int i = 1; i <= 10; i++) {
16.            synchronized (this) {
17.                while (turnoHermano1 != esHermano1) {
18.                    try {
19.                        wait();
20.                    } catch (InterruptedException e) {
21.                        e.printStackTrace();
22.                    }
23.                }
24.
25.                textArea.append(nombre + " está sembrando el árbol " + i + "\n");
26.
27.                try {
28.                    Thread.sleep(500);
29.                } catch (InterruptedException e) {
30.                    e.printStackTrace();
31.                }
32.
33.                textArea.append(nombre + " ha terminado de sembrar el árbol " + i + "\n");
34.
35.                turnoHermano1 = !esHermano1;
36.                notifyAll();
37.            }
38.        }
39.    }
40. }
41.
42. class Hermano extends Thread {
43.     private final Jardin jardin;
44.     private final boolean esHermano1;
45.
46.     public Hermano(Jardin jardin, String nombre, boolean esHermano1) {
47.         super(nombre);
48.         this.jardin = jardin;
49.         this.esHermano1 = esHermano1;
50.     }
51.
52.     @Override
53.     public void run() {

```

```

54.     jardin.sembrar(getName(), esHermano1);
55. }
56. }
57.
58. public class HermanosV2RunnableGUI {
59.     public static void main(String[] args) {
60.         JFrame frame = new JFrame("Hermanos Sembrando Árboles");
61.         JTextArea textArea = new JTextArea(20, 40);
62.         textArea.setEditable(false);
63.         JScrollPane scrollPane = new JScrollPane(textArea);
64.         frame.add(scrollPane, BorderLayout.CENTER);
65.         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
66.         frame.pack();
67.         frame.setVisible(true);
68.
69.         Jardin jardin = new Jardin(textArea);
70.         Hermano hermano1 = new Hermano(jardin, "Hermano 1", true);
71.         Hermano hermano2 = new Hermano(jardin, "Hermano 2", false);
72.
73.         hermano1.start();
74.         hermano2.start();
75.
76.         try {
77.             hermano1.join();
78.             hermano2.join();
79.         } catch (InterruptedException e) {
80.             e.printStackTrace();
81.         }
82.
83.         textArea.append("Ambos hermanos han terminado de sembrar los árboles\n");
84.     }
85. }
86.

```

```

1. package Practica2;
2.
3. class Jardin {
4.     private boolean turnoHermano1 = true;
5.
6.     public synchronized void sembrar(String nombre, boolean esHermano1) {
7.         for (int i = 1; i <= 10; i++) {
8.             synchronized (this) {
9.                 while (turnoHermano1 != esHermano1) {
10.                    try {
11.                        wait();
12.                    } catch (InterruptedException e) {
13.                        e.printStackTrace();
14.                    }
15.                }
16.
17.                System.out.println(nombre + " esta sembrando el arbol " + i);
18.
19.                try {
20.                    Thread.sleep(500);
21.                } catch (InterruptedException e) {
22.                    e.printStackTrace();
23.                }
24.
25.                System.out.println(nombre + " ha terminado de sembrar el arbol " + i);
26.
27.                turnoHermano1 = !esHermano1;
28.                notifyAll();
29.            }

```

```

30.     }
31. }
32. }
33.
34. class Hermano implements Runnable {
35.     private final Jardin jardin;
36.     private final String nombre;
37.     private final boolean esHermano1;
38.
39.     public Hermano(Jardin jardin, String nombre, boolean esHermano1) {
40.         this.jardin = jardin;
41.         this.nombre = nombre;
42.         this.esHermano1 = esHermano1;
43.     }
44.
45.     @Override
46.     public void run() {
47.         jardin.sembrar(nombre, esHermano1);
48.     }
49. }
50.
51. public class HermanosV2Runnbale {
52.     public static void main(String[] args) {
53.         Jardin jardin = new Jardin();
54.         Hermano hermano1 = new Hermano(jardin, "Hermano 1", true);
55.         Hermano hermano2 = new Hermano(jardin, "Hermano 2", false);
56.
57.         Thread t1 = new Thread(hermano1);
58.         Thread t2 = new Thread(hermano2);
59.
60.         t1.start();
61.         t2.start();
62.
63.         try {
64.             t1.join();
65.             t2.join();
66.         } catch (InterruptedException e) {
67.             e.printStackTrace();
68.         }
69.
70.         System.out.println("Ambos hermanos han terminado de sembrar los árboles");
71.     }
72. }
73.

```

```

1. package Practica2;
2.
3. class Jardin {
4.     private boolean turnoHermano1 = true;
5.
6.     public synchronized void sembrar(String nombre, boolean esHermano1) {
7.         for (int i = 1; i <= 10; i++) {
8.             synchronized (this) {
9.                 while (turnoHermano1 != esHermano1) {
10.                    try {
11.                        wait();
12.                    } catch (InterruptedException e) {
13.                        e.printStackTrace();
14.                    }
15.                }
16.
17.                System.out.println(nombre + " esta sembrando el arbol " + i);
18.

```



```

19.         try {
20.             Thread.sleep(500);
21.         } catch (InterruptedException e) {
22.             e.printStackTrace();
23.         }
24.
25.         System.out.println(nombre + " ha terminado de sembrar el arbol " + i);
26.
27.         turnoHermano1 = !esHermano1;
28.         notifyAll();
29.     }
30. }
31. }
32. }
33.
34. class Hermano extends Thread {
35.     private final Jardin jardin;
36.     private final boolean esHermano1;
37.
38.     public Hermano(Jardin jardin, String nombre, boolean esHermano1) {
39.         super(nombre);
40.         this.jardin = jardin;
41.         this.esHermano1 = esHermano1;
42.     }
43.
44.     @Override
45.     public void run() {
46.         jardin.sembrar(getName(), esHermano1);
47.     }
48. }
49.
50. public class HermanosV2Thread {
51.     public static void main(String[] args) {
52.         Jardin jardin = new Jardin();
53.         Hermano hermano1 = new Hermano(jardin, "Hermano 1", true);
54.         Hermano hermano2 = new Hermano(jardin, "Hermano 2", false);
55.
56.         hermano1.start();
57.         hermano2.start();
58.
59.         try {
60.             hermano1.join();
61.             hermano2.join();
62.         } catch (InterruptedException e) {
63.             e.printStackTrace();
64.         }
65.
66.         System.out.println("Ambos hermanos han terminado de sembrar los arboles");
67.     }
68. }
69.

```

```

1. package Practica2;
2.
3. import java.awt.*;
4. import javax.swing.*;
5. import java.awt.event.*;
6.
7. class Pelota {
8.     private int x = 7, xCambio = 7;
9.     private int y = 0, yCambio = 2;
10.    private int diametro = 10;
11.    private int rectIzqX = 0, rectDerX = 100;

```

```

12.     private int rectSupY = 0, rectInfY = 100;
13.
14.     public void mover() {
15.         // Verifica colisiones y cambia la dirección
16.         if (x + xCambio <= rectIzqX || x + xCambio >= rectDerX) {
17.             xCambio = -xCambio;
18.         }
19.         if (y + yCambio <= rectSupY || y + yCambio >= rectInfY) {
20.             yCambio = -yCambio;
21.         }
22.
23.         // Actualiza posición
24.         x += xCambio;
25.         y += yCambio;
26.     }
27.
28.     public void dibujar(Graphics g) {
29.         g.setColor(Color.red);
30.         g.fillOval(x, y, diametro, diametro);
31.     }
32. }
33.
34. public class NoHiloPelota extends JPanel implements ActionListener {
35.     private JButton iniciar;
36.     private Timer timer; // Usaremos un Timer en lugar de un Thread
37.     private Pelota pelota;
38.
39.     public NoHiloPelota() {
40.         iniciar = new JButton("Iniciar");
41.         add(iniciar);
42.         iniciar.addActionListener(this);
43.         pelota = new Pelota();
44.         timer = new Timer(50, e -> {
45.             pelota.mover();
46.             repaint(); // Redibuja la pelota con la nueva posición
47.         });
48.     }
49.
50.     @Override
51.     protected void paintComponent(Graphics g) {
52.         super.paintComponent(g);
53.         g.drawRect(0, 0, 110, 110); // Dibuja el área de rebote
54.         pelota.dibujar(g); // Dibuja la pelota
55.     }
56.
57.     public void actionPerformed(ActionEvent event) {
58.         if (!timer.isRunning()) {
59.             timer.start(); // Iniciar el Timer cuando se presiona el botón
60.         }
61.     }
62.
63.     public static void main(String[] args) {
64.         JFrame frame = new JFrame("Hilo Pelota");
65.         NoHiloPelota panel = new NoHiloPelota();
66.         frame.add(panel);
67.         frame.setSize(200, 200);
68.         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
69.         frame.setVisible(true);
70.     }
71. }
72.

```

```

1. package Practica2;

```

```

2.
3. class Contador {
4.     private int sumaPares = 0;
5.     private int sumaImpares = 0;
6.
7.     public synchronized void sumarNumero(int num, boolean esPar) {
8.         if (esPar) {
9.             sumaPares += num;
10.        } else {
11.            sumaImpares += num;
12.        }
13.    }
14.
15.    public void imprimirSumaTotal() {
16.        System.out.println("Suma total de pares: " + sumaPares);
17.        System.out.println("Suma total de impares: " + sumaImpares);
18.        System.out.println("Suma total de pares e impares: " + (sumaPares + sumaImpares));
19.    }
20. }
21.
22. class HiloNumeros implements Runnable {
23.     private final Contador contador;
24.     private final boolean esPar;
25.     private final String nombreHilo;
26.
27.     public HiloNumeros(Contador contador, boolean esPar, String nombreHilo) {
28.         this.contador = contador;
29.         this.esPar = esPar;
30.         this.nombreHilo = nombreHilo;
31.     }
32.
33.     @Override
34.     public void run() {
35.         int inicio = esPar ? 2 : 1;
36.         for (int i = inicio; i <= 10; i += 2) {
37.             System.out.println(nombreHilo + " - " + (esPar ? "Par" : "Impar") + ": " + i);
38.             contador.sumarNumero(i, esPar);
39.             try {
40.                 Thread.sleep(500); // Simula procesamiento
41.             } catch (InterruptedException e) {
42.                 Thread.currentThread().interrupt();
43.             }
44.         }
45.     }
46. }
47.
48. public class NumerosEstadosRunnable {
49.     public static void main(String[] args) {
50.         Contador contador = new Contador();
51.
52.         // Crear dos hilos con la misma clase, pero con configuraciones diferentes
53.         Thread hiloA = new Thread(new HiloNumeros(contador, false, "Hilo A")); // Imprime
54.         Thread hiloB = new Thread(new HiloNumeros(contador, true, "Hilo B")); // Imprime
55.
56.         // Iniciar hilos
57.         hiloA.start();
58.         hiloB.start();
59.
60.         // Esperar a que los hilos terminen
61.         try {
62.             hiloA.join();
63.             hiloB.join();
64.         } catch (InterruptedException e) {

```

```

65.     System.out.println("Error en la sincronización de los hilos.");
66. }
67.
68.     // Imprimir la suma total
69.     contador.imprimirSumaTotal();
70. }
71. }
72.

```

```

1. package Practica2;
2.
3. class Contador {
4.     private int sumaPares = 0;
5.     private int sumaImpares = 0;
6.
7.     public synchronized void sumarNumero(int num, boolean esPar) {
8.         if (esPar) {
9.             sumaPares += num;
10.        } else {
11.            sumaImpares += num;
12.        }
13.    }
14.
15.    public void imprimirSumaTotal() {
16.        System.out.println("Suma total de pares: " + sumaPares);
17.        System.out.println("Suma total de impares: " + sumaImpares);
18.        System.out.println("Suma total de pares e impares: " + (sumaPares + sumaImpares));
19.    }
20. }
21.
22. class HiloNumeros extends Thread {
23.     private final Contador contador;
24.     private final boolean esPar;
25.
26.     public HiloNumeros(Contador contador, boolean esPar, String nombreHilo) {
27.         super(nombreHilo);
28.         this.contador = contador;
29.         this.esPar = esPar;
30.     }
31.
32.     @Override
33.     public void run() {
34.         int inicio = esPar ? 2 : 1;
35.         for (int i = inicio; i <= 10; i += 2) {
36.             System.out.println(getName() + " - " + (esPar ? "Par" : "Impar") + ": " + i);
37.             contador.sumarNumero(i, esPar);
38.             try {
39.                 Thread.sleep(500); // Simula procesamiento
40.             } catch (InterruptedException e) {
41.                 Thread.currentThread().interrupt();
42.             }
43.         }
44.     }
45. }
46.
47. public class NumerosEstadosThread {
48.     public static void main(String[] args) {
49.         Contador contador = new Contador();
50.
51.         // Crear dos hilos con la misma clase, pero con configuraciones diferentes
52.         HiloNumeros hiloA = new HiloNumeros(contador, false, "Hilo A"); // Imprime impares
53.         HiloNumeros hiloB = new HiloNumeros(contador, true, "Hilo B"); // Imprime pares
54.

```

```

55.         // Iniciar hilos
56.         hiloA.start();
57.         hiloB.start();
58.
59.         // Esperar a que los hilos terminen
60.         try {
61.             hiloA.join();
62.             hiloB.join();
63.         } catch (InterruptedException e) {
64.             System.out.println("Error en la sincronización de los hilos.");
65.         }
66.
67.         // Imprimir la suma total
68.         contador.imprimirSumaTotal();
69.     }
70. }
71.

```

```

1. package Practica2;
2.
3. import java.util.Random;
4.
5. class Contador {
6.     private int sumaPares = 0;
7.     private int sumaImpares = 0;
8.     private final Object lock = new Object();
9.     private boolean turnoPar = true;
10.
11.     public void sumarNumero(int num, boolean esPar, String nombreHilo) {
12.         synchronized (lock) {
13.             while (turnoPar != esPar) {
14.                 try {
15.                     lock.wait();
16.                 } catch (InterruptedException e) {
17.                     Thread.currentThread().interrupt();
18.                 }
19.             }
20.
21.             System.out.println(nombreHilo + " - " + (esPar ? "Par" : "Impar") + ": " +
num);
22.
23.             if (esPar) {
24.                 sumaPares += num;
25.             } else {
26.                 sumaImpares += num;
27.             }
28.
29.             turnoPar = !turnoPar;
30.             lock.notifyAll();
31.         }
32.
33.     public void imprimirSumaTotal() {
34.         System.out.println("Suma total de pares: " + sumaPares);
35.         System.out.println("Suma total de impares: " + sumaImpares);
36.         System.out.println("Suma total de pares e impares: " + (sumaPares + sumaImpares));
37.     }
38. }
39.
40. class HiloNumeros implements Runnable {
41.     private final Contador contador;
42.     private final boolean esPar;
43.     private final String nombreHilo;
44.     private final Random random = new Random();

```

```

45.
46.     public HiloNumeros(Contador contador, boolean esPar, String nombreHilo) {
47.         this.contador = contador;
48.         this.esPar = esPar;
49.         this.nombreHilo = nombreHilo;
50.     }
51.
52.     public boolean esPar(int num) {
53.         return num % 2 == 0;
54.     }
55.
56.     @Override
57.     public void run() {
58.         for (int i = 0; i < 25; i++) {
59.             int num = random.nextInt(100) + 1;
60.             contador.sumarNumero(num, esPar, nombreHilo);
61.             try {
62.                 Thread.sleep(500); // Simula procesamiento
63.             } catch (InterruptedException e) {
64.                 Thread.currentThread().interrupt();
65.             }
66.         }
67.     }
68. }
69.
70. public class NumerosSincronizadosRunnable {
71.     public static void main(String[] args) {
72.         Contador contador = new Contador();
73.
74.         // Crear dos hilos con la misma clase, pero con configuraciones diferentes
75.         Thread hiloA = new Thread(new HiloNumeros(contador, false, "Hilo A")); // Imprime
impares
76.         Thread hiloB = new Thread(new HiloNumeros(contador, true, "Hilo B")); // Imprime
pares
77.
78.         // Iniciar hilos
79.         hiloA.start();
80.         hiloB.start();
81.
82.         // Esperar a que los hilos terminen
83.         try {
84.             hiloA.join();
85.             hiloB.join();
86.         } catch (InterruptedException e) {
87.             System.out.println("Error en la sincronización de los hilos.");
88.         }
89.
90.         // Imprimir la suma total
91.         contador.imprimirSumaTotal();
92.     }
93. }
94.

```

```

1. package Practica2;
2.
3. import java.util.Random;
4.
5. class Contador {
6.     private int sumaPares = 0;
7.     private int sumaImpares = 0;
8.     private final Object lock = new Object();
9.     private boolean turnoPar = true;
10.

```

```

11.     public void sumarNumero(int num, boolean esPar, String nombreHilo) {
12.         synchronized (lock) {
13.             while (turnoPar != esPar) {
14.                 try {
15.                     lock.wait();
16.                 } catch (InterruptedException e) {
17.                     Thread.currentThread().interrupt();
18.                 }
19.             }
20.
21.             System.out.println(nombreHilo + " - " + (esPar ? "Par" : "Impar") + ": " +
num);
22.             if (esPar) {
23.                 sumaPares += num;
24.             } else {
25.                 sumaImpares += num;
26.             }
27.
28.             turnoPar = !turnoPar;
29.             lock.notifyAll();
30.         }
31.     }
32.
33.     public void imprimirSumaTotal() {
34.         System.out.println("Suma total de pares: " + sumaPares);
35.         System.out.println("Suma total de impares: " + sumaImpares);
36.         System.out.println("Suma total de pares e impares: " + (sumaPares + sumaImpares));
37.     }
38. }
39.
40. class HiloNumeros extends Thread {
41.     private final Contador contador;
42.     private final boolean esPar;
43.     private final Random random = new Random();
44.
45.     public HiloNumeros(Contador contador, boolean esPar, String nombreHilo) {
46.         super(nombreHilo);
47.         this.contador = contador;
48.         this.esPar = esPar;
49.     }
50.
51.     public boolean esPar(int num) {
52.         return num % 2 == 0;
53.     }
54.
55.     @Override
56.     public void run() {
57.         for (int i = 0; i < 25; i++) {
58.             int num = random.nextInt(100) + 1;
59.             contador.sumarNumero(num, esPar, getName());
60.             try {
61.                 Thread.sleep(500); // Simula procesamiento
62.             } catch (InterruptedException e) {
63.                 Thread.currentThread().interrupt();
64.             }
65.         }
66.     }
67. }
68.
69. public class NumerosSincronizadosThread {
70.     public static void main(String[] args) {
71.         Contador contador = new Contador();
72.
73.         // Crear dos hilos con la misma clase, pero con configuraciones diferentes
74.         HiloNumeros hiloA = new HiloNumeros(contador, false, "Hilo A"); // Imprime impares

```

```

75.     HiloNumeros hiloB = new HiloNumeros(contador, true, "Hilo B"); // Imprime pares
76.
77.     // Iniciar hilos
78.     hiloA.start();
79.     hiloB.start();
80.
81.     // Esperar a que los hilos terminen
82.     try {
83.         hiloA.join();
84.         hiloB.join();
85.     } catch (InterruptedException e) {
86.         System.out.println("Error en la sincronización de los hilos.");
87.     }
88.
89.     // Imprimir la suma total
90.     contador.imprimirSumaTotal();
91. }
92. }
93.

```

```

1. package Practica2;
2.
3. import java.util.ArrayList;
4. import java.util.Collections;
5. import java.util.List;
6. import java.util.Random;
7.
8. class Controlador {
9.     private boolean turnoPositivos = true; // Comienza con positivos
10.    private int sumaPositivos = 0;
11.    private int sumaNegativos = 0;
12.    private final List<Integer> numeros; // Lista de numeros generados aleatoriamente
13.    private int index = 0; // Indice para recorrer la lista de numeros
14.
15.    public Controlador() {
16.        // Generar numeros aleatorios entre -100 y 100
17.        numeros = new ArrayList<>();
18.        Random random = new Random();
19.        for (int i = 0; i < 201; i++) {
20.            numeros.add(random.nextInt(201) - 100);
21.        }
22.        Collections.shuffle(numeros); // Mezclar los numeros aleatoriamente
23.    }
24.
25.    public synchronized void procesarNumero(String nombre, boolean esPositivo) {
26.        while (index < numeros.size()) {
27.            while (turnoPositivos != esPositivo) { // Espera su turno
28.                try {
29.                    wait(); // Hilo espera hasta que sea su turno
30.                } catch (InterruptedException e) {
31.                    Thread.currentThread().interrupt();
32.                }
33.            }
34.
35.            if (index >= numeros.size()) {
36.                break;
37.            }
38.
39.            int num = numeros.get(index);
40.            index++;
41.
42.            // Verifica si el numero es del tipo que debe procesar el hilo
43.            if ((num >= 0 && esPositivo) || (num < 0 && !esPositivo)) {

```



```

44.         System.out.println(nombre + " identfico: " + num);
45.         if (esPositivo) {
46.             sumaPositivos += num;
47.         } else {
48.             sumaNegativos += num;
49.         }
50.
51.         // Cambia el turno y notifica a los otros hilos
52.         turnoPositivos = !turnoPositivos;
53.         notifyAll(); // Notifica al otro hilo
54.     }
55. }
56. notifyAll(); // Asegura que el otro hilo no se quede esperando
57. }
58.
59. public void imprimirSumaTotal() {
60.     System.out.println("Suma de numeros positivos: " + sumaPositivos);
61.     System.out.println("Suma de numeros negativos: " + sumaNegativos);
62. }
63. }
64.
65. class HiloNumeros implements Runnable {
66.     private final Controlador controlador;
67.     private final boolean esPositivo;
68.     private final String nombreHilo;
69.
70.     public HiloNumeros(Controlador controlador, boolean esPositivo, String nombreHilo) {
71.         this.controlador = controlador;
72.         this.esPositivo = esPositivo;
73.         this.nombreHilo = nombreHilo;
74.     }
75.
76.     @Override
77.     public void run() {
78.         controlador.procesarNumero(nombreHilo, esPositivo);
79.     }
80. }
81.
82. public class PosNegRunnable {
83.     public static void main(String[] args) {
84.         Controlador controlador = new Controlador();
85.
86.         // Crear dos hilos con la misma clase, pero con configuraciones diferentes
87.         Thread hiloPositivos = new Thread(new HiloNumeros(controlador, true, "Hilo
Positivos"));
88.         Thread hiloNegativos = new Thread(new HiloNumeros(controlador, false, "Hilo
Negativos"));
89.
90.         // Iniciar hilos
91.         hiloPositivos.start();
92.         hiloNegativos.start();
93.
94.         // Esperar a que los hilos terminen
95.         try {
96.             hiloPositivos.join();
97.             hiloNegativos.join();
98.         } catch (InterruptedException e) {
99.             System.out.println("Error en la sincronizacion de los hilos.");
100.        }
101.
102.        // Imprimir la suma total
103.        controlador.imprimirSumaTotal();
104.    }
105. }
106.

```

```

1. package Practica2;
2.
3. import java.util.ArrayList;
4. import java.util.Collections;
5. import java.util.List;
6. import java.util.Random;
7.
8. class Controlador {
9.     private boolean turnoPositivos = true; // Comienza con positivos
10.    private int sumaPositivos = 0;
11.    private int sumaNegativos = 0;
12.    private final List<Integer> numeros; // Lista de numeros generados aleatoriamente
13.    private int index = 0; // Indice para recorrer la lista de numeros
14.
15.    public Controlador() {
16.        // Generar numeros aleatorios entre -100 y 100
17.        numeros = new ArrayList<>();
18.        Random random = new Random();
19.        for (int i = 0; i < 201; i++) {
20.            numeros.add(random.nextInt(201) - 100);
21.        }
22.        Collections.shuffle(numeros); // Mezclar los numeros aleatoriamente
23.    }
24.
25.    public synchronized void procesarNumero(String nombre, boolean esPositivo) {
26.        while (index < numeros.size()) {
27.            while (turnoPositivos != esPositivo) { // Espera su turno
28.                try {
29.                    wait(); // Hilo espera hasta que sea su turno
30.                } catch (InterruptedException e) {
31.                    Thread.currentThread().interrupt();
32.                }
33.            }
34.
35.            if (index >= numeros.size()) {
36.                break;
37.            }
38.
39.            int num = numeros.get(index);
40.            index++;
41.
42.            // Verifica si el numero es del tipo que debe procesar el hilo
43.            if ((num >= 0 && esPositivo) || (num < 0 && !esPositivo)) {
44.                System.out.println(nombre + " identifico: " + num);
45.                if (esPositivo) {
46.                    sumaPositivos += num;
47.                } else {
48.                    sumaNegativos += num;
49.                }
50.
51.                // Cambia el turno y notifica a los otros hilos
52.                turnoPositivos = !turnoPositivos;
53.                notifyAll(); // Notifica al otro hilo
54.            }
55.        }
56.        notifyAll(); // Asegura que el otro hilo no se quede esperando
57.    }
58.
59.    public void imprimirSumaTotal() {
60.        System.out.println("Suma de numeros positivos: " + sumaPositivos);
61.        System.out.println("Suma de numeros negativos: " + sumaNegativos);
62.    }
63. }
64.
65. class HiloNumeros implements Runnable {

```

```

66.     private final Controlador controlador;
67.     private final boolean esPositivo;
68.     private final String nombreHilo;
69.
70.     public HiloNumeros(Controlador controlador, boolean esPositivo, String nombreHilo) {
71.         this.controlador = controlador;
72.         this.esPositivo = esPositivo;
73.         this.nombreHilo = nombreHilo;
74.     }
75.
76.     @Override
77.     public void run() {
78.         controlador.procesarNumero(nombreHilo, esPositivo);
79.     }
80. }
81.
82. public class PosNegThread {
83.     public static void main(String[] args) {
84.         Controlador controlador = new Controlador();
85.
86.         // Crear dos hilos con la misma clase, pero con configuraciones diferentes
87.         Thread hiloPositivos = new Thread(new HiloNumeros(controlador, true, "Hilo
Positivos"));
88.         Thread hiloNegativos = new Thread(new HiloNumeros(controlador, false, "Hilo
Negativos"));
89.
90.         // Iniciar hilos
91.         hiloPositivos.start();
92.         hiloNegativos.start();
93.
94.         // Esperar a que los hilos terminen
95.         try {
96.             hiloPositivos.join();
97.             hiloNegativos.join();
98.         } catch (InterruptedException e) {
99.             System.out.println("Error en la sincronizacion de los hilos.");
100.        }
101.
102.        // Imprimir la suma total
103.        controlador.imprimirSumaTotal();
104.    }
105. }
106.

```

```

1. package Practica2;
2.
3. import java.awt.*;
4. import javax.swing.*;
5. import java.awt.event.*;
6.
7. class Pelota implements Runnable { // Implementar Runnable para usar hilos
8.     private Graphics g;
9.     private int x = 7, xCambio = 7;
10.    private int y = 0, yCambio = 2;
11.    private int diametro = 10;
12.    private int rectIzqX = 0, rectDerX = 100;
13.    private int rectSupY = 0, rectInfY = 100;
14.    private boolean running = true; // Control para detener el hilo
15.
16.    public Pelota(Graphics graficos) {
17.        g = graficos;
18.    }
19.

```

```

20.     public void detener() {
21.         running = false; // Método para detener la pelota si es necesario
22.     }
23.
24.     @Override
25.     public void run() {
26.         g.drawRect(rectIzqX, rectSupY, rectDerX - rectIzqX + 10, rectInfY - rectSupY +
27.         10);
28.         for (int n = 1; n < 1000 && running; n++) { // Se ejecuta mientras "running" sea
29.         true
30.             g.setColor(Color.white);
31.             g.fillOval(x, y, diametro, diametro);
32.             if (x + xCambio <= rectIzqX || x + xCambio >= rectDerX) {
33.                 xCambio = -xCambio;
34.             }
35.             if (y + yCambio <= rectSupY || y + yCambio >= rectInfY) {
36.                 yCambio = -yCambio;
37.             }
38.
39.             x += xCambio;
40.             y += yCambio;
41.             g.setColor(Color.red);
42.             g.fillOval(x, y, diametro, diametro);
43.
44.             try {
45.                 Thread.sleep(50); // Pequeña pausa para ver el movimiento
46.             } catch (InterruptedException e) {
47.                 System.err.println("Excepcion de inactividad");
48.                 Thread.currentThread().interrupt();
49.             }
50.         }
51.     }
52. }
53.
54. public class hiloPelota extends JPanel implements ActionListener {
55.     private JButton iniciar;
56.     private Thread hiloPelota; // Hilo para manejar la animación de la pelota
57.
58.     public hiloPelota() {
59.         iniciar = new JButton("Iniciar");
60.         add(iniciar);
61.         iniciar.addActionListener(this);
62.     }
63.
64.     public void actionPerformed(ActionEvent event) {
65.         if (event.getSource() == iniciar) {
66.             Graphics g = getGraphics();
67.             Pelota pelota = new Pelota(g);
68.             hiloPelota = new Thread(pelota); // Crear hilo con la pelota
69.             hiloPelota.start(); // Iniciar hilo
70.         }
71.     }
72.
73.     public static void main(String[] args) {
74.         JFrame frame = new JFrame("Hilo Pelota");
75.         hiloPelota panel = new hiloPelota();
76.         frame.add(panel);
77.         frame.setSize(200, 200);
78.         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
79.         frame.setVisible(true);
80.     }
81. }

```