REPORTE – PRACTICA 4

Programación concurrente y paralela

29 DE MARZO DE 2025

ALUMNO: ENRIQUE HERNANDEZ LUNA
DOCENTE: DRA CARMEN CERON GARCIA

En esta práctica se realizaron 2 programas los cuales fueron:

1. El programa de los hermanos sembradores, pero en esta implementación se usarán monitores.
2. El programa anterior pero ahora se debe agregar un hermano más para que entre los 3 siembran 120 árboles.

```java
package Practica4; // Define el paquete donde se encuentra el programa

// Clase principal del programa
public class ProblemaHermanos {
  public static void main(String[] args) {
    // Cantidad total de árboles a sembrar
    int totalArboles = 10;

    // Se crea una instancia del monitor compartido, pasándole el total de árboles
    Monitor_Parcela parcela = new Monitor_Parcela(totalArboles);

    // Se crean dos hilos, uno para cada hermano
    Hermano1 h1 = new Hermano1(parcela);
    Hermano2 h2 = new Hermano2(parcela);

    // Se asignan nombres a los hilos para identificar su salida
    h1.setName("H1");
    h2.setName("H2");

    // Se inician ambos hilos
    h1.start();
    h2.start();
  }
}
```

```java
package Practica4; // Paquete del monitor

// Clase que actúa como monitor para coordinar el acceso a la parcela
public class Monitor_Parcela {
  private int arbolesSembrados = 0; // Contador de árboles sembrados
  private final int totalArboles;   // Total de árboles a sembrar (límite)
  private boolean turnoHermano1 = true; // Variable para alternar turnos

  // Constructor que recibe la cantidad total de árboles a sembrar
  public Monitor_Parcela(int totalArboles) {
    this.totalArboles = totalArboles;
  }

  // Método sincronizado para consultar si aún deben sembrarse árboles
  public synchronized boolean debeContinuar() {
    return arbolesSembrados < totalArboles;
  }

  // Método sincronizado que controla la siembra del Hermano 1
  public synchronized void SiembraHermano_1() {
    while (!turnoHermano1 && debeContinuar()) {
      try {
        wait(); // Espera si no es su turno
      } catch (InterruptedException ex) {
        Thread.currentThread().interrupt();
      }
    }

    if (!debeContinuar()) return; // Sale si ya no hay más árboles por sembrar

    arbolesSembrados++; // Incrementa el contador de árboles sembrados
    System.out.println(Thread.currentThread().getName() + ": Hermano 1 siembra el árbol: "
+ arbolesSembrados);
    try {
      Thread.sleep(500); // Simula el tiempo de siembra
    } catch (InterruptedException ex) {
      Thread.currentThread().interrupt();
    }
```

```
38.
39.     turnoHermano1 = false; // Cambia el turno al Hermano 2
40.     notifyAll(); // Notifica a los hilos en espera
41.   }
42.
43.   // Método sincronizado que controla la siembra del Hermano 2
44.   public synchronized void SiembraHermano_2() {
45.     while (turnoHermano1 && debeContinuar()) {
46.       try {
47.         wait(); // Espera si no es su turno
48.       } catch (InterruptedException ex) {
49.         Thread.currentThread().interrupt();
50.       }
51.     }
52.
53.     if (!debeContinuar()) return; // Sale si ya no hay más árboles por sembrar
54.
55.     arbolesSembrados++; // Incrementa el contador de árboles sembrados
56.     System.out.println(Thread.currentThread().getName() + ": Hermano 2 siembra el árbol: "
+ arbolesSembrados);
57.     try {
58.       Thread.sleep(500); // Simula el tiempo de siembra
59.     } catch (InterruptedException ex) {
60.       Thread.currentThread().interrupt();
61.     }
62.
63.     turnoHermano1 = true; // Cambia el turno al Hermano 1
64.     notifyAll(); // Notifica a los hilos en espera
65.   }
66. }
```

```
 1. package Practica4; // Paquete de la clase
 2.
 3. // Clase que representa el hilo del Hermano 2
 4. public class Hermano2 extends Thread {
 5.   private final Monitor_Parcela parcela; // Referencia al monitor compartido
 6.
 7.   // Constructor que recibe el monitor
 8.   public Hermano2(Monitor_Parcela parcela) {
 9.     this.parcela = parcela;
10.   }
11.
12.   // Método que se ejecuta cuando el hilo comienza
13.   public void run() {
14.     while (parcela.debeContinuar()) {
15.       parcela.SiembraHermano_2(); // Llama al método de siembra del Hermano 2
16.     }
17.   }
18. }
```

```
 1. package Practica4; // Paquete de la clase
 2.
 3. // Clase que representa el hilo del Hermano 1
 4. public class Hermano1 extends Thread {
 5.   private final Monitor_Parcela parcela; // Referencia al monitor compartido
 6.
 7.   // Constructor que recibe el monitor
 8.   public Hermano1(Monitor_Parcela parcela) {
 9.     this.parcela = parcela;
10.   }
11.
```

```
12.   // Método que se ejecuta cuando el hilo comienza
13.   public void run() {
14.     while (parcela.debeContinuar()) {
15.       parcela.SiembraHermano_1(); // Llama al método de siembra del Hermano 1
16.     }
17.   }
18. }
```

```
@enrique on c-y-p  3.11.1 on  master 117ms  java Practica4/ProblemaHermanos
  H1: Hermano 1 siembra el Ã¡rbol: 1
  H2: Hermano 2 siembra el Ã¡rbol: 2
  H1: Hermano 1 siembra el Ã¡rbol: 3
  H2: Hermano 2 siembra el Ã¡rbol: 4
  H1: Hermano 1 siembra el Ã¡rbol: 5
  H2: Hermano 2 siembra el Ã¡rbol: 6
  H1: Hermano 1 siembra el Ã¡rbol: 7
  H2: Hermano 2 siembra el Ã¡rbol: 8
  H1: Hermano 1 siembra el Ã¡rbol: 9
  H2: Hermano 2 siembra el Ã¡rbol: 10
@enrique on c-y-p  3.11.1 on  master 5.349s
```

```
 1. package Practica4.TresHermanos; // Declaramos el paquete del proyecto
 2.
 3. // Clase principal donde se inicia la ejecución del programa
 4. public class Problema3Hermanos {
 5.     public static void main(String[] args) {
 6.         int totalArboles = 15; // Definimos el número total de árboles que se desean
sembrar (120)
 7.
 8.         // Creamos una instancia del monitor compartido que regula la siembra
 9.         Monitor_Parcela parcela = new Monitor_Parcela(totalArboles);
10.
11.         // Creamos los tres hilos, uno por cada hermano
12.         Hermano1 h1 = new Hermano1(parcela);
13.         Hermano2 h2 = new Hermano2(parcela);
14.         Hermano3 h3 = new Hermano3(parcela);
15.
16.         // Asignamos nombres para diferenciarlos en la salida
17.         h1.setName("H1");
18.         h2.setName("H2");
19.         h3.setName("H3");
20.
21.         // Iniciamos los tres hilos
22.         h1.start();
23.         h2.start();
24.         h3.start();
25.     }
26. }
```

```
 1. package Practica4.TresHermanos; // Paquete del monitor
 2.
 3. // Monitor que controla el acceso sincronizado a la parcela
 4. public class Monitor_Parcela {
 5.     private int arbolesSembrados = 0; // Contador de árboles sembrados
 6.     private final int totalArboles;   // Total de árboles a sembrar
```

```
 7.      private int turnoActual = 1;      // Turno de siembra: 1, 2 o 3
 8.
 9.      // Constructor que recibe cuántos árboles se sembrarán en total
10.      public Monitor_Parcela(int totalArboles) {
11.          this.totalArboles = totalArboles;
12.      }
13.
14.      // Indica si aún hay árboles pendientes por sembrar
15.      public synchronized boolean debeContinuar() {
16.          return arbolesSembrados < totalArboles;
17.      }
18.
19.      // Método sincronizado para el Hermano 1
20.      public synchronized void SiembraHermano_1() {
21.          // Espera mientras no sea su turno y aún falten árboles
22.          while (turnoActual != 1 && debeContinuar()) {
23.              try {
24.                  wait(); // Libera el monitor y espera
25.              } catch (InterruptedException ex) {
26.                  Thread.currentThread().interrupt(); // Restablece el estado de
interrupción
27.              }
28.          }
29.
30.          if (!debeContinuar()) return; // Verificación final por seguridad
31.
32.          arbolesSembrados++; // Aumenta el contador
33.          System.out.println(Thread.currentThread().getName() + ": Hermano 1 siembra el
arbol: " + arbolesSembrados);
34.          dormir(); // Simula el tiempo de siembra
35.
36.          turnoActual = 2; // Cambia el turno al Hermano 2
37.          notifyAll(); // Despierta a los demás hilos
38.      }
39.
40.      // Método sincronizado para el Hermano 2
41.      public synchronized void SiembraHermano_2() {
42.          while (turnoActual != 2 && debeContinuar()) {
43.              try {
44.                  wait();
45.              } catch (InterruptedException ex) {
46.                  Thread.currentThread().interrupt();
47.              }
48.          }
49.
50.          if (!debeContinuar()) return;
51.
52.          arbolesSembrados++;
53.          System.out.println(Thread.currentThread().getName() + ": Hermano 2 siembra el
arbol: " + arbolesSembrados);
54.          dormir();
55.
56.          turnoActual = 3; // Turno para el Hermano 3
57.          notifyAll();
58.      }
59.
60.      // Método sincronizado para el Hermano 3
61.      public synchronized void SiembraHermano_3() {
62.          while (turnoActual != 3 && debeContinuar()) {
63.              try {
64.                  wait();
65.              } catch (InterruptedException ex) {
66.                  Thread.currentThread().interrupt();
67.              }
68.          }
```

```
69.
70.         if (!debeContinuar()) return;
71.
72.         arbolesSembrados++;
73.         System.out.println(Thread.currentThread().getName() + ": Hermano 3 siembra el
arbol: " + arbolesSembrados);
74.         dormir();
75.
76.         turnoActual = 1; // Regresa el turno al Hermano 1
77.         notifyAll();
78.     }
79.
80.     // Método auxiliar para simular el tiempo de siembra
81.     private void dormir() {
82.         try {
83.             Thread.sleep(500); // 500 ms por árbol
84.         } catch (InterruptedException ex) {
85.             Thread.currentThread().interrupt();
86.         }
87.     }
88. }
```

```
 1. package Practica4.TresHermanos;
 2.
 3. // Hilo que representa al Hermano 3
 4. public class Hermano3 extends Thread {
 5.     private final Monitor_Parcela parcela;
 6.
 7.     public Hermano3(Monitor_Parcela parcela) {
 8.         this.parcela = parcela;
 9.     }
10.
11.     public void run() {
12.         while (parcela.debeContinuar()) {
13.             parcela.SiembraHermano_3();
14.         }
15.     }
16. }
```

```
 1. package Practica4.TresHermanos;
 2.
 3. // Hilo que representa al Hermano 2
 4. public class Hermano2 extends Thread {
 5.     private final Monitor_Parcela parcela;
 6.
 7.     public Hermano2(Monitor_Parcela parcela) {
 8.         this.parcela = parcela;
 9.     }
10.
11.     public void run() {
12.         while (parcela.debeContinuar()) {
13.             parcela.SiembraHermano_2();
14.         }
15.     }
16. }
```

```
 1. package Practica4.TresHermanos; // Declaramos el paquete
 2.
 3. // Hilo que representa al Hermano 1
```

```java
public class Hermano1 extends Thread {
    private final Monitor_Parcela parcela; // Referencia al monitor

    // Constructor que recibe el monitor compartido
    public Hermano1(Monitor_Parcela parcela) {
        this.parcela = parcela;
    }

    // Método principal del hilo
    public void run() {
        while (parcela.debeContinuar()) { // Se repite mientras haya árboles
            parcela.SiembraHermano_1();   // Intenta sembrar su árbol
        }
    }
}
```

```
@enrique on c-y-p 3.11.1 on master java Practica4/TresHermanos/Problema3Hermanos
H1: Hermano 1 siembra el arbol: 1
H2: Hermano 2 siembra el arbol: 2
H3: Hermano 3 siembra el arbol: 3
H1: Hermano 1 siembra el arbol: 4
H2: Hermano 2 siembra el arbol: 5
H3: Hermano 3 siembra el arbol: 6
H1: Hermano 1 siembra el arbol: 7
H2: Hermano 2 siembra el arbol: 8
H3: Hermano 3 siembra el arbol: 9
H1: Hermano 1 siembra el arbol: 10
H2: Hermano 2 siembra el arbol: 11
H3: Hermano 3 siembra el arbol: 12
H1: Hermano 1 siembra el arbol: 13
H2: Hermano 2 siembra el arbol: 14
H3: Hermano 3 siembra el arbol: 15
@enrique on c-y-p 3.11.1 on master 7.789s
```