PRACTICA 1

programación concurrente y paralela

20 DE ENERO DE 2025

ALUMNO: ENRIQUE HERNANDEZ LUNA
DOCENTE: DRA. CARMEN CERON GARCIA

Ejemplo 1:

```
package practica1;


// Definir la clase ThreadUsingExtends que extiende de Thread
class ThreadUsingExtends extends Thread {
 // Atributos
 String name;
 int[] numbers;

 // Constructor de la clase ThreadUsingExtends
 ThreadUsingExtends(String name, int[] numbers) {
  this.name = name;
  this.numbers = numbers;
 }

 // Método run que se ejecuta al llamar al método start
 @Override
 public void run() {
  for (int number : numbers) {
   System.out.println(name + ": " + number);
  }
 }

 // Método para imprimir la información del hilo
 public void printInfo() {
  System.out.println("Thread name: " + this.getName());
```

```
    System.out.println("ID: " + this.getId());
    System.out.println("Priority: " + this.getPriority());
    System.out.println("Alive: " + this.isAlive());
    System.out.println("State: " + this.getState());
  }
}

public class Program1 {
  // Método para imprimir la información de un hilo
  public static void printThreadInfo(ThreadUsingExtends thread) {
    thread.printInfo();
  }

  public static void main(String[] args) {
    // Crear dos arrays de 10 números
    int[] numbers1 = {1, 2, 3, 4, 5};
    int[] numbers2 = { 6, 7, 8, 9, 10 };
    int[] numbers3 = { 11, 12, 13, 14, 15 };

    // Crear dos instancias de la clase ThreadUsingExtends
    Thread t1 = new ThreadUsingExtends("T1", numbers1);
    Thread t2 = new ThreadUsingExtends("T2", numbers2);
    Thread t3 = new ThreadUsingExtends("T3", numbers3);

    // Iniciar los hilos
    t1.start();
```

```
    t2.start();
    t3.start();

    // esperar a que los hilos terminen
    try {
      t1.join();
      t2.join();
      t3.join();
    } catch (InterruptedException e) {
      System.out.println(e);
    }

    // Imprimir informacion de los hilos
    printThreadInfo((ThreadUsingExtends) t1);
    printThreadInfo((ThreadUsingExtends) t2);
    printThreadInfo((ThreadUsingExtends) t3);
  }
}
```

Ejemplo 2:

```
package practica1;


// Definir la clase ThreadUsingRunnable que implementa Runnable
class ThreadUsingRunnable implements Runnable {
 // Atributos
 String name;
 int[] numbers;

 // Constructor de la clase ThreadUsingRunnable
 ThreadUsingRunnable(String name, int[] numbers) {
  this.name = name;
  this.numbers = numbers;
 }

 // Método run que se ejecuta al llamar al método start
 @Override
 public void run() {
  for (int number : numbers) {
   System.out.println(name + ": " + number);
  }
 }

 // Método para imprimir la información del hilo
 public void printInfo(Thread thread) {
  System.out.println("Thread name: " + thread.getName());
```

```
    System.out.println("ID: " + thread.getId());
    System.out.println("Priority: " + thread.getPriority());
    System.out.println("Alive: " + thread.isAlive());
    System.out.println("State: " + thread.getState());
  }
}

public class Program2 {
  // Método para imprimir la información de un hilo
  public static void printThreadInfo(ThreadUsingRunnable runnable, Thread thread) {
    runnable.printInfo(thread);
  }

  public static void main(String[] args) {
    // Crear dos arrays de 10 números
    int[] numbers1 = {1, 2, 3, 4, 5};
    int[] numbers2 = {6, 7, 8, 9, 10};
    int[] numbers3 = {11, 12, 13, 14, 15};

    // Crear instancias de la clase ThreadUsingRunnable
    ThreadUsingRunnable runnable1 = new ThreadUsingRunnable("T1", numbers1);
    ThreadUsingRunnable runnable2 = new ThreadUsingRunnable("T2", numbers2);
    ThreadUsingRunnable runnable3 = new ThreadUsingRunnable("T3", numbers3);

    // Crear hilos
    Thread t1 = new Thread(runnable1);
```

```
    Thread t2 = new Thread(runnable2);
    Thread t3 = new Thread(runnable3);

    // Iniciar los hilos
    t1.start();
    t2.start();
    t3.start();

    // Esperar a que los hilos terminen
    try {
      t1.join();
      t2.join();
      t3.join();
    } catch (InterruptedException e) {
      System.out.println(e);
    }

    // Imprimir informacion de los hilos
    printThreadInfo(runnable1, t1);
    printThreadInfo(runnable2, t2);
    printThreadInfo(runnable3, t3);
  }
}
```

```java
package Practica1;

// Definir la clase ThreadUsingExtends que extiende de Thread
class ThreadUsingExtends extends Thread {
  // Atributos
  String name;

  // Constructor de la clase ThreadUsingExtends
  ThreadUsingExtends(String name) {
    this.name = name;
  }

  // Método run que se ejecuta al llamar al método start
  @Override
  public void run() {
    System.out.println("Hilo " + name + " - Estado: " + getState());
    try {
      Thread.sleep(2000); // Simula trabajo
    } catch (InterruptedException e) {
      e.printStackTrace();
    }
    System.out.println("Hilo " + name + " - Estado: " + getState());
  }

  // Método para imprimir la información del hilo
  public void printInfo() {
    System.out.println("Thread name: " + this.getName());
    System.out.println("ID: " + this.getId());
    System.out.println("Priority: " + this.getPriority());
    System.out.println("Alive: " + this.isAlive());
    System.out.println("State: " + this.getState());
  }
}

public class Program1thread {
    // Método para imprimir la información de un hilo
    public static void printThreadInfo(ThreadUsingExtends thread) {
      thread.printInfo();
    }

    public static void main(String[] args) {

      // Crear dos instancias de la clase ThreadUsingExtends
      Thread t1 = new ThreadUsingExtends("T1");
      Thread t2 = new ThreadUsingExtends("T2");
      Thread t3 = new ThreadUsingExtends("T3");
      Thread t4 = new ThreadUsingExtends("T4");

      // Imprimir estados antes de iniciar los hilos
      System.out.println("Estado de t1 antes de start(): " + t1.getState());
      System.out.println("Estado de t2 antes de start(): " + t2.getState());
      System.out.println("Estado de t3 antes de start(): " + t3.getState());
      System.out.println("Estado de t4 antes de start(): " + t4.getState());

      // Iniciar los hilos
      t1.start();
      t2.start();
      t3.start();
      t4.start();

      // Imprimir estados después de iniciar los hilos
      System.out.println("Estado de t1 después de start(): " + t1.getState());
      System.out.println("Estado de t2 después de start(): " + t2.getState());
      System.out.println("Estado de t3 después de start(): " + t3.getState());
      System.out.println("Estado de t4 después de start(): " + t4.getState());
```

```
66.
67.         // esperar a que los hilos terminen
68.         try {
69.             t1.join();
70.             t2.join();
71.             t3.join();
72.             t4.join();
73.         } catch (InterruptedException e) {
74.             System.out.println(e);
75.         }
76.
77.         // Imprimir estados después de que los hilos terminen
78.         System.out.println("Estado de t1 después de join(): " + t1.getState());
79.         System.out.println("Estado de t2 después de join(): " + t2.getState());
80.         System.out.println("Estado de t3 después de join(): " + t3.getState());
81.         System.out.println("Estado de t4 después de join(): " + t4.getState());
82.
83.         // Imprimir informacion de los hilos
84.         printThreadInfo((ThreadUsingExtends) t1);
85.         printThreadInfo((ThreadUsingExtends) t2);
86.         printThreadInfo((ThreadUsingExtends) t3);
87.         printThreadInfo((ThreadUsingExtends) t4);
88.     }
89. }
90.
```

```
 1. package Practica1;
 2.
 3. // Definir la clase ThreadUsingRunnable que implementa Runnable
 4. class ThreadUsingRunnable implements Runnable {
 5.     // Atributos
 6.     String name;
 7.
 8.     // Constructor de la clase ThreadUsingRunnable
 9.     ThreadUsingRunnable(String name) {
10.         this.name = name;
11.     }
12.
13.     // Método run que se ejecuta al llamar al método start
14.     @Override
15.     public void run() {
16.         System.out.println("Hilo " + name + " - Estado: " +
Thread.currentThread().getState());
17.         try {
18.             Thread.sleep(2000); // Simula trabajo
19.         } catch (InterruptedException e) {
20.             e.printStackTrace();
21.         }
22.         System.out.println("Hilo " + name + " - Estado: " +
Thread.currentThread().getState());
23.     }
24.
25.     // Método para imprimir la información del hilo
26.     public void printInfo(Thread thread) {
27.         System.out.println("Thread name: " + thread.getName());
28.         System.out.println("ID: " + thread.getId());
29.         System.out.println("Priority: " + thread.getPriority());
30.         System.out.println("Alive: " + thread.isAlive());
31.         System.out.println("State: " + thread.getState());
32.     }
33. }
34.
35. public class Program1runnable {
```

```
36.     // Método para imprimir la información de un hilo
37.     public static void printThreadInfo(ThreadUsingRunnable runnable, Thread thread) {
38.         runnable.printInfo(thread);
39.     }
40.
41.     public static void main(String[] args) {
42.
43.         // Crear dos instancias de la clase ThreadUsingRunnable
44.         ThreadUsingRunnable runnable1 = new ThreadUsingRunnable("T1");
45.         ThreadUsingRunnable runnable2 = new ThreadUsingRunnable("T2");
46.         ThreadUsingRunnable runnable3 = new ThreadUsingRunnable("T3");
47.         ThreadUsingRunnable runnable4 = new ThreadUsingRunnable("T4");
48.
49.         Thread t1 = new Thread(runnable1);
50.         Thread t2 = new Thread(runnable2);
51.         Thread t3 = new Thread(runnable3);
52.         Thread t4 = new Thread(runnable4);
53.
54.         // Imprimir estados antes de iniciar los hilos
55.         System.out.println("Estado de t1 antes de start(): " + t1.getState());
56.         System.out.println("Estado de t2 antes de start(): " + t2.getState());
57.         System.out.println("Estado de t3 antes de start(): " + t3.getState());
58.         System.out.println("Estado de t4 antes de start(): " + t4.getState());
59.
60.         // Iniciar los hilos
61.         t1.start();
62.         t2.start();
63.         t3.start();
64.         t4.start();
65.
66.         // Imprimir estados después de iniciar los hilos
67.         System.out.println("Estado de t1 después de start(): " + t1.getState());
68.         System.out.println("Estado de t2 después de start(): " + t2.getState());
69.         System.out.println("Estado de t3 después de start(): " + t3.getState());
70.         System.out.println("Estado de t4 después de start(): " + t4.getState());
71.
72.         // esperar a que los hilos terminen
73.         try {
74.             t1.join();
75.             t2.join();
76.             t3.join();
77.             t4.join();
78.         } catch (InterruptedException e) {
79.             System.out.println(e);
80.         }
81.
82.         // Imprimir estados después de que los hilos terminen
83.         System.out.println("Estado de t1 después de join(): " + t1.getState());
84.         System.out.println("Estado de t2 después de join(): " + t2.getState());
85.         System.out.println("Estado de t3 después de join(): " + t3.getState());
86.         System.out.println("Estado de t4 después de join(): " + t4.getState());
87.
88.         // Imprimir informacion de los hilos
89.         printThreadInfo(runnable1, t1);
90.         printThreadInfo(runnable2, t2);
91.         printThreadInfo(runnable3, t3);
92.         printThreadInfo(runnable4, t4);
93.     }
94. }
95.
```

```
1. package Practica1;
2.
```

```java
class TaskThread extends Thread {
  // Constructor que asigna un nombre al hilo
  public TaskThread(String name) {
    super(name);
  }

  @Override
  public void run() {
    // Imprimir el nombre del hilo en ejecución
    System.out.println(getName() + " en ejecución");
    try {
      // Simular trabajo durmiendo el hilo por 2000 milisegundos
      Thread.sleep(2000);
    } catch (InterruptedException e) {
      // Imprimir un mensaje si el hilo es interrumpido
      System.out.println(getName() + " interrumpido");
    }
    // Imprimir un mensaje indicando que el hilo ha finalizado
    System.out.println(getName() + " finalizado");
  }
}

class MonitorThread extends Thread {
  private final Thread threadToMonitor;

  // Constructor que asigna el hilo a monitorear y un nombre al monitor
  public MonitorThread(Thread threadToMonitor, String name) {
    super(name);
    this.threadToMonitor = threadToMonitor;
  }

  @Override
  public void run() {
    // Obtener el estado inicial del hilo
    Thread.State previousState = threadToMonitor.getState();
    // Imprimir el estado inicial del hilo
    System.out.println(threadToMonitor.getName() + " estado inicial: " + previousState);
    // Mientras el hilo no haya terminado
    while (threadToMonitor.getState() != Thread.State.TERMINATED) {
      // Obtener el estado actual del hilo
      Thread.State currentState = threadToMonitor.getState();
      // Si el estado actual es diferente al estado anterior
      if (currentState != previousState) {
        // Imprimir el cambio de estado
        System.out.println(threadToMonitor.getName() + " cambió a estado: " +
currentState);
        // Actualizar el estado anterior
        previousState = currentState;
      }
    }
    // Imprimir el estado final del hilo
    System.out.println(threadToMonitor.getName() + " estado final: " +
threadToMonitor.getState());
  }
}

public class Program2thread {
  public static void main(String[] args) {
    // Crear el hilo con la tarea definida y asignarle un nombre
    TaskThread thread = new TaskThread("Hilo-1");

    // Crear un monitor para observar el estado del hilo
    MonitorThread monitor = new MonitorThread(thread, "Monitor-Hilo");

    // Iniciar el monitor y el hilo
```

```
66.     monitor.start();
67.     thread.start();
68.
69.     // Esperar la finalización del hilo con join
70.     try {
71.       thread.join();
72.       monitor.join();
73.     } catch (InterruptedException e) {
74.       // Imprimir un mensaje si hay un error al esperar los hilos
75.       System.out.println("Error al esperar los hilos");
76.     }
77.   }
78. }
79.
```

```
 1. package Practica1;
 2.
 3. public class Program2runnable {
 4.   public static void main(String[] args) {
 5.     // Crear un hilo usando la interfaz Runnable
 6.     Runnable task = () -> {
 7.       // Imprimir el nombre del hilo en ejecución
 8.       System.out.println(Thread.currentThread().getName() + " en ejecución");
 9.       try {
10.         // Simular trabajo durmiendo el hilo por 2000 milisegundos
11.         Thread.sleep(2000);
12.       } catch (InterruptedException e) {
13.         // Imprimir un mensaje si el hilo es interrumpido
14.         System.out.println(Thread.currentThread().getName() + " interrumpido");
15.       }
16.       // Imprimir un mensaje indicando que el hilo ha finalizado
17.       System.out.println(Thread.currentThread().getName() + " finalizado");
18.     };
19.
20.     // Crear el hilo con la tarea definida y asignarle un nombre
21.     Thread thread = new Thread(task, "Hilo-1");
22.
23.     // Crear un monitor para observar el estado del hilo
24.     Thread monitor = new Thread(() -> {
25.       // Obtener el estado inicial del hilo
26.       Thread.State previousState = thread.getState();
27.       // Imprimir el estado inicial del hilo
28.       System.out.println(thread.getName() + " estado inicial: " + previousState);
29.       // Mientras el hilo no haya terminado
30.       while (thread.getState() != Thread.State.TERMINATED) {
31.         // Obtener el estado actual del hilo
32.         Thread.State currentState = thread.getState();
33.         // Si el estado actual es diferente al estado anterior
34.         if (currentState != previousState) {
35.           // Imprimir el cambio de estado
36.           System.out.println(thread.getName() + " cambió a estado: " + currentState);
37.           // Actualizar el estado anterior
38.           previousState = currentState;
39.         }
40.       }
41.       // Imprimir el estado final del hilo
42.       System.out.println(thread.getName() + " estado final: " + thread.getState());
43.     }, "Monitor-Hilo");
44.
45.     // Iniciar el monitor y el hilo
46.     monitor.start();
47.     thread.start();
48.
```

```
    // Esperar la finalización del hilo con join
    try {
      thread.join();
      monitor.join();
    } catch (InterruptedException e) {
      // Imprimir un mensaje si hay un error al esperar los hilos
      System.out.println("Error al esperar los hilos");
    }
  }
}

```

```
package Practica1;

// Definir la clase ThreadUsingRunnable que implementa Runnable
class ThreadUsingRunnable implements Runnable {
  private String name;

  // Constructor que asigna un nombre al hilo
  public ThreadUsingRunnable(String name) {
    this.name = name;
  }

  // Método para verificar si un número es par
  public String esPar(int i) {
    return (i % 2 == 0) ? "es par" : "es impar";
  }

  // Método para verificar si un número es primo
  public String esPrimo(int i) {
    if (i <= 1) return "no es primo";
    for (int j = 2; j < i; j++) {
      if (i % j == 0) return "no es primo";
    }
    return "es primo";
  }

  // Método run que se ejecuta al llamar al método start
  @Override
  public void run() {
    for (int i = 1; i <= 10; i += 1) {
      // Imprimir el nombre del hilo, el número, si es par o impar y si es primo o no
      System.out.println(name + " - " + i + " " + esPar(i) + " y " + esPrimo(i));
    }
  }
}

public class Program3runnable {
  public static void main(String[] args) {
    // Crear tres instancias de la clase ThreadUsingRunnable
    ThreadUsingRunnable runnable1 = new ThreadUsingRunnable("T1");
    ThreadUsingRunnable runnable2 = new ThreadUsingRunnable("T2");
    ThreadUsingRunnable runnable3 = new ThreadUsingRunnable("T3");

    Thread t1 = new Thread(runnable1);
    Thread t2 = new Thread(runnable2);
    Thread t3 = new Thread(runnable3);

    // Iniciar los hilos
    t1.start();
    t2.start();
    t3.start();
  }
```

```
52. }
53.
```

```
 1. package Practica1;
 2.
 3. // Definir la clase ThreadUsingExtends que extiende de Thread
 4. class ThreadUsingExtends extends Thread {
 5.   private String name;
 6.
 7.   // Constructor que asigna un nombre al hilo
 8.   public ThreadUsingExtends(String name) {
 9.     this.name = name;
10.   }
11.
12.   // Método para verificar si un número es par
13.   public String esPar(int i) {
14.     return (i % 2 == 0) ? "es par" : "es impar";
15.   }
16.
17.   // Método para verificar si un número es primo
18.   public String esPrimo(int i) {
19.     if (i <= 1) return "no es primo";
20.     for (int j = 2; j < i; j++) {
21.       if (i % j == 0) return "no es primo";
22.     }
23.     return "es primo";
24.   }
25.
26.   // Método run que se ejecuta al llamar al método start
27.   @Override
28.   public void run() {
29.     for (int i = 1; i <= 10; i += 1) {
30.       // Imprimir el nombre del hilo, el número, si es par o impar y si es primo o no
31.       System.out.println(name + " - " + i + " " + esPar(i) + " y " + esPrimo(i));
32.     }
33.   }
34. }
35.
36. public class Program3thread {
37.   public static void main(String[] args) {
38.     // Crear tres instancias de la clase ThreadUsingExtends
39.     ThreadUsingExtends t1 = new ThreadUsingExtends("T1");
40.     ThreadUsingExtends t2 = new ThreadUsingExtends("T2");
41.     ThreadUsingExtends t3 = new ThreadUsingExtends("T3");
42.
43.     // Iniciar los hilos
44.     t1.start();
45.     t2.start();
46.     t3.start();
47.   }
48. }
49.
```

```
@enrique on c-y-p 3.11.1 on master 1.018s java Practica1/Program1thread
Estado de t1 antes de start(): NEW
Estado de t2 antes de start(): NEW
Estado de t3 antes de start(): NEW
Estado de t4 antes de start(): NEW
Estado de t1 despuÃ©s de start(): RUNNABLE
Estado de t2 despuÃ©s de start(): RUNNABLE
Estado de t3 despuÃ©s de start(): RUNNABLE
Estado de t4 despuÃ©s de start(): RUNNABLE
Hilo T2 - Estado: RUNNABLE
Hilo T4 - Estado: RUNNABLE
Hilo T1 - Estado: RUNNABLE
Hilo T3 - Estado: RUNNABLE
Hilo T3 - Estado: RUNNABLE
Hilo T3 - Estado: RUNNABLE
Hilo T3 - Estado: RUNNABLE
Hilo T3 - Estado: RUNNABLE
Hilo T2 - Estado: RUNNABLE
Hilo T3 - Estado: RUNNABLE
Hilo T2 - Estado: RUNNABLE
Hilo T3 - Estado: RUNNABLE
Hilo T2 - Estado: RUNNABLE
Hilo T1 - Estado: RUNNABLE
Hilo T4 - Estado: RUNNABLE
Estado de t1 despuÃ©s de join(): TERMINATED
Estado de t2 despuÃ©s de join(): TERMINATED
Estado de t3 despuÃ©s de join(): TERMINATED
Estado de t4 despuÃ©s de join(): TERMINATED
Thread name: Thread-0
ID: 14
Priority: 5
Alive: false
State: TERMINATED
Thread name: Thread-1
ID: 15
Priority: 5
Alive: false
State: TERMINATED
Thread name: Thread-2
ID: 16
Priority: 5
Alive: false
State: TERMINATED
Thread name: Thread-3
ID: 17
Priority: 5
Alive: false
State: TERMINATED
@enrique on c-y-p 3.11.1 on master 2.213s
```

```
@enrique on c-y-p 3.11.1 on master 1.258s java Practica1/Program1runnable
Estado de t1 antes de start(): NEW
Estado de t2 antes de start(): NEW
Estado de t3 antes de start(): NEW
Estado de t4 antes de start(): NEW
Estado de t1 despuÃ©s de start(): RUNNABLE
Estado de t2 despuÃ©s de start(): RUNNABLE
Hilo T1 - Estado: RUNNABLE
Hilo T3 - Estado: RUNNABLE
Hilo T4 - Estado: RUNNABLE
Hilo T2 - Estado: RUNNABLE
Estado de t3 despuÃ©s de start(): BLOCKED
Estado de t4 despuÃ©s de start(): TIMED_WAITING
Hilo T1 - Estado: RUNNABLE
Hilo T2 - Estado: RUNNABLE
Hilo T4 - Estado: RUNNABLE
Hilo T3 - Estado: RUNNABLE
Estado de t1 despuÃ©s de join(): TERMINATED
Estado de t1 despuÃ©s de join(): TERMINATED
Estado de t1 despuÃ©s de join(): TERMINATED
Estado de t2 despuÃ©s de join(): TERMINATED
Estado de t3 despuÃ©s de join(): TERMINATED
Estado de t4 despuÃ©s de join(): TERMINATED
Thread name: Thread-0
ID: 14
Priority: 5
Alive: false
State: TERMINATED
Thread name: Thread-1
ID: 15
Priority: 5
Alive: false
State: TERMINATED
Thread name: Thread-2
ID: 16
Priority: 5
Alive: false
State: TERMINATED
Thread name: Thread-3
ID: 17
Priority: 5
Alive: false
State: TERMINATED
@enrique on c-y-p 3.11.1 on master 2.225s
```

```
@enrique on c-y-p 3.11.1 on master 982ms java Practica1/Program2thread
Hilo-1 en ejecuciÃ³n
Hilo-1 estado inicial: RUNNABLE
Hilo-1 cambiÃ³ a estado: TIMED_WAITING
Hilo-1 cambiÃ³ a estado: RUNNABLE
Hilo-1 finalizado
Hilo-1 estado final: TERMINATED
@enrique on c-y-p 3.11.1 on master 2.178s
```

```
@enrique on c-y-p 3.11.1 on master 1.038s java Practica1/Program2runnable
Hilo-1 en ejecuciÃ³n
Hilo-1 estado inicial: RUNNABLE
Hilo-1 cambiÃ³ a estado: TIMED_WAITING
Hilo-1 cambiÃ³ a estado: RUNNABLE
Hilo-1 finalizado
Hilo-1 estado final: TERMINATED
@enrique on c-y-p 3.11.1 on master 2.164s
```

```
@enrique on c-y-p 3.11.1 on master 954ms java Practica1/Program3runnable
T3 - 1 es impar y no es primo
T3 - 2 es par y es primo
T3 - 3 es impar y es primo
T3 - 4 es par y no es primo
T1 - 1 es impar y no es primo
T2 - 1 es impar y no es primo
T1 - 2 es par y es primo
T3 - 5 es impar y es primo
T1 - 3 es impar y es primo
T2 - 2 es par y es primo
T1 - 4 es par y no es primo
T3 - 6 es par y no es primo
T1 - 5 es impar y es primo
T2 - 3 es impar y es primo
T1 - 6 es par y no es primo
T3 - 7 es impar y es primo
T1 - 7 es impar y es primo
T2 - 4 es par y no es primo
T1 - 8 es par y no es primo
T3 - 8 es par y no es primo
T1 - 9 es impar y no es primo
T2 - 5 es impar y es primo
T1 - 10 es par y no es primo
T3 - 9 es impar y no es primo
T2 - 6 es par y no es primo
T3 - 10 es par y no es primo
T2 - 7 es impar y es primo
T2 - 8 es par y no es primo
T2 - 9 es impar y no es primo
T2 - 10 es par y no es primo
@enrique on c-y-p 3.11.1 on master 203ms
```

```
• @enrique on c-y-p  3.11.1 on  master 961ms  java Practica1/Program3thread
  T3 - 1 es impar y no es primo
  T3 - 2 es par y es primo
  T1 - 1 es impar y no es primo
  T2 - 1 es impar y no es primo
  T1 - 2 es par y es primo
  T3 - 3 es impar y es primo
  T1 - 3 es impar y es primo
  T2 - 2 es par y es primo
  T1 - 4 es par y no es primo
  T3 - 4 es par y no es primo
  T1 - 5 es impar y es primo
  T2 - 3 es impar y es primo
  T1 - 6 es par y no es primo
  T3 - 5 es impar y es primo
  T1 - 7 es impar y es primo
  T2 - 4 es par y no es primo
  T1 - 8 es par y no es primo
  T3 - 6 es par y no es primo
  T1 - 9 es impar y no es primo
  T2 - 5 es impar y es primo
  T1 - 10 es par y no es primo
  T3 - 7 es impar y es primo
  T2 - 6 es par y no es primo
  T3 - 8 es par y no es primo
  T2 - 7 es impar y es primo
  T3 - 9 es impar y no es primo
  T2 - 8 es par y no es primo
  T2 - 9 es impar y no es primo
  T3 - 10 es par y no es primo
  T2 - 10 es par y no es primo
 @enrique on c-y-p  3.11.1 on  master 170ms
```