

EI1024/MT1024 “Programación Concurrente y Paralela” 2025–26 Nombre y apellidos (1): Nombre y apellidos (2): Tiempo empleado para tareas en casa en formato <i>h:mm</i> (obligatorio):	Entregable para Laboratorio la01_g
---	---

Tema 03. Conceptos Básicos de Concurrencia en Java

1 Se desea crear y arrancar dos hebras que se ejecuten concurrentemente, como sigue:

- Cada hebra tiene un identificador entero único, cuya secuencia comienza en 0.
- Cada hebra escribe en pantalla mil veces su identificador.

1.1) Crea las hebras a partir de una subclase de la clase **Thread**, y utiliza el método **start** para arrancar las hebras.

Escribe a continuación el código completo partiendo del siguiente esquema.

```

class MiHebra ... Thread {
    ...
    public MiHebra( int miId ) {
        ...
    }
    public void run() {
        for( int i = 0; i < 1000; i++ ) {
            System.out.println( "Hebra: " + miId );
        }
    }
}
class EjemploCreacionThread {
    public static void main( String args[] ) {
        new ...
        new ...
    }
}

```

.....

.....

.....

.....

.....

.....

1.2) Crea las hebras a partir de un objeto de la clase **Thread** y un objeto de una clase que implemente la interfaz **Runnable**, utilizando el método **start** para arrancar las hebras.

Escribe a continuación el código completo partiendo del siguiente esquema.

```

class MiRun ... Runnable {
    ...
    public MiRun( int miId ) {
        ...
    }
}

```

```

    public void run() {
        for( int i = 0; i < 1000; i++ ) {
            System.out.println( "Hebra: " + miId );
        }
    }
}
class EjemploCreacionRunnable {
    public static void main( String args[] ) {
        new ...
        new ...
    }
}

```

.....

.....

.....

.....

.....

.....

1.3) Al probar los códigos, ¿cómo observas si las dos hebras se han ejecutado concurrentemente?

.....

.....

1.4) Explica, SIN PROBARLO, qué ocurriría si se sustituyese el método **start** por el método **run** en alguno de los códigos anteriores.

.....

.....

1.5) Sustituye el método **start** por el método **run**, ejecuta el código y describe qué ocurre.

.....

.....

2 Se desea crear y arrancar dos hebras que se ejecuten concurrentemente, donde:

- Cada hebra tiene un **identificador entero único**, cuya secuencia comienza en 0.
- Cada hebra también recibe **dos números en el constructor**.
- Cada hebra utiliza un bucle para calcular la suma de los números comprendidos entre estos números, ambos inclusive.

Por su parte, el programa principal:

- Imprime un mensaje al inicio del programa.
- Crea y rranca las hebras para que realicen un millón de sumas.
- Finaliza con la impresión de otro mensaje.

Escribe a continuación el código completo partiendo del siguiente esquema.

```

class MiHebra ... Thread { // (A)
    // ... (B)

    public MiHebra( int miId, int num1, int num2 ) {
        // ... (C)
    }

    public void run() {
        long suma = 0;

        System.out.println( "Hebra Auxiliar " + miId + " , inicia calculo" );
        for( int i = num1; i <= num2 ; i++ ) {
            suma += (long) i;
        }
        System.out.println( "Hebra Auxiliar " + miId + " , suma: " + suma);
    }
}

class EjemploDaemon {
    public static void main( String args[] ) {
        System.out.println( "Hebra Principal inicia" );
        // Crea y arranca hebra t0 sumando desde 1 hasta 1000000
        // Crea y arranca hebra t1 sumando desde 1 hasta 1000000
        // ... (D)
        // Espera la finalizacion de las hebras t0 y t1
        // ... (E)
        System.out.println( "Hebra Principal finaliza" );
    }
}

```

2.1) Escribe la declaración de variables y el constructor de la clase **MiHebra**.

Estas líneas se deben insertar a continuación de las líneas marcadas con (A), (B), y (C).

.....

.....

.....

.....

.....

2.2) Escribe el código del programa principal que realiza la creación y el arranque de las hebras.

Estas líneas se deben insertar a continuación de la línea marcada con (D).

.....

.....

.....

.....

2.3) Al probar el código, ¿se observa que las hebras se han ejecutado concurrentemente con el programa principal? ¿Qué hebra ha finalizado antes, el programa principal o las hebras auxiliares? Razona tus respuestas.

.....

.....

.....

-
- 2.4) Si antes de arrancar las hebras, éstas se definen como hebras de tipo “Daemon”, ¿cómo se altera la ejecución? Prueba el nuevo código y razona tu respuesta.

.....

.....

.....

.....

- 2.5) Escribe las órdenes necesarias para asegurar que el programa principal espere la finalización de las dos hebras antes de realizar la impresión final.

Estas líneas se deben insertar a continuación de la línea marcada con (E).

.....

.....

.....

.....

.....

- 2.6) ¿Cómo se ha alterado la ejecución con estos cambios? ¿Cambiaría su comportamiento si las hebras fuesen no “Daemon”? Razona tus respuestas.

.....

.....

.....

.....

- 3** Duplica el código obtenido tras obtener resolver el ejercicio 2.6, y sustituye TODAS las líneas insertadas después de la línea marcada por (D), para que ahora se manejen hebras virtuales.

- 3.1) Compila y ejecuta el código y describe su funcionamiento.

.....

.....

.....

.....

.....

- 3.2) Modifica el código eliminando TODAS las insertadas después de la línea marcada por (E). ¿Cómo se altera su funcionamiento?

.....

.....

.....

.....

.....

3.3) Así pues, ¿las hebras virtuales funcionan como “Daemon” o como no “Daemon”? ¿Por qué?

```
.....
.....
.....
.....
.....
```

4 Se desea crear y arrancar un conjunto de hebras, donde:

- Cada hebra tiene un **identificador entero único**, cuya secuencia comienza en 0.
- Cada hebra realiza un **millón de incrementos sobre un objeto compartido** recibido en el constructor.
- Cada hebra imprime un **mensaje justo antes** de comenzar dicha tarea y también **justo después** de terminar dicha tarea.

A continuación se muestra un código que se puede emplear como punto de partida. Este código contiene la definición de la clase del objeto (**CuentaIncrementos**) sobre el cual se realizarán los incrementos, y un esquema de la clase **MiHebra**. Además, el programa principal realiza la comprobación y extracción de los argumentos de entrada de la línea de comandos.

En el resto de las prácticas de la asignatura, siempre que se vayan a usar argumentos de la línea de comandos, habrá que comprobar su número y tipo de forma similar.

```
// =====
class CuentaIncrementos {
// =====

    long contador = 0;

    // =====
    void incrementaContador() {
        contador++;
    }

    // =====
    long dameContador() {
        return( contador );
    }
}

// =====
class MiHebra extends Thread {
// =====
    // Declaracion de variables
    // ...

    // =====
    // Definicion del constructor, si es necesario
    // ...

    // =====
    public void run() {
        System.out.println( "Hebra: " + miId + " Comenzando incrementos" );
        // Bucle de 1000000 incrementos del objeto compartido
        // ...
        System.out.println( "Hebra: " + miId + " Terminando incrementos" );
    }
}
```

```

    }
}

// =====
class EjemploIncrementos {
// =====

// -----
public static void main( String args[] ) {
    int    numHebras;

    // Comprobacion y extraccion de los argumentos de entrada.
    if( args.length != 1 ) {
        System.err.println( "Uso: java programa <numHebras>" );
        System.exit( -1 );
    }
    try {
        numHebras = Integer.parseInt( args[ 0 ] );
        if( numHebras <= 0 ) {
            System.err.println( "Uso: [ java programa <numHebras> ] donde numHebras > 0" );
            System.exit( -1 );
        }
    } catch( NumberFormatException ex ) {
        numHebras = -1;
        System.out.println( "ERROR: Argumentos numericos incorrectos." );
        System.exit( -1 );
    }
    System.out.println( "numHebras: " + numHebras );

    // -----  INCLUIR NUEVO CODIGO A CONTINUACION -----
    // ...
}
}

```

4.2) Escribe un programa principal que realice las siguientes tareas, en el orden especificado:

1. El programa principal debe averiguar el número de hebras que debe crear. Este número es recibido por el programa a través de la línea de argumentos. Así, por ejemplo, el comando `java EjemploIncrementos 4` creará 4 hebras.

Si el número de argumentos de la línea de argumentos no es correcto, el programa debe avisar y terminar. Además, si algún argumento no es válido (por ejemplo, se esperaba un argumento numérico y no lo es), el programa también debe avisar y terminar.

Este apartado aparece resuelto en el código anterior, y servirá de ejemplo para el resto de prácticas.

2. El programa principal debe crear e inicializar el objeto compartido.

Escribe a continuación la parte de tu código que realiza tal tarea.

.....

3. El programa principal debe imprimir el valor inicial del contador.

Escribe a continuación la parte de tu código que realiza tal tarea.

.....

4. El programa principal debe crear y arrancar las hebras, utilizando un vector de hebras.

Escribe a continuación la parte de tu código que realiza tal tarea.

.....

5. El programa principal debe esperar a que todas las hebras finalicen su ejecución.

Escribe a continuación la parte de tu código que realiza tal tarea.

.....

6. El programa principal debe imprimir el valor final del contador.

Escribe a continuación la parte de tu código que realiza tal tarea.

.....

- 4.3) Comprueba si hay concurrencia entre las hebras. ¿Cómo puedes demostrarlo? Razona tu respuesta.

.....

- 4.4) ¿Si se crean 4 hebras, qué valor debería imprimir el programa principal? ¿Cuál es el valor escrito por el ordenador?

.....

- 4.5) ¿Dónde crees que está el error?

Nota: Este apartado no se evaluará, dado que este problema y su solución se estudiarán a fondo en temas siguientes. Simplemente lo hemos añadido para que comencéis a reflexionar sobre este problema.

.....

- 5** Se dispone de una interfaz gráfica sencilla, cuyo código se muestra a continuación, que permite examinar si un número es primo o no. Este código también contabiliza el número de veces que se ha pulsado el botón **Pulsa aquí**.

El código es el siguiente:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;

// =====
public class GUIPrimoSencillo {
// =====

    // Declaration of variables.
    JFrame      container;
    JPanel      jpanel;
    JTextField  txfNumero, txfMensajes, txfSugerencias;
    JButton     btnPulsaAqui, btnComienzaCalculo;
    int         numVecesPulsado = 0;

    // =====
    public static void main( String args[] ) {
        GUIPrimoSencillo gui = new GUIPrimoSencillo();
        SwingUtilities.invokeLater(new Runnable(){
            public void run(){
                gui.go();
            }
        });
    }
}
```



```
// ===== INICIO CODIGO A MODIFICAR EN EJERCICIO 5.2 ===== */
public void go() {
    // Variables.
    JPanel      tempPanel;

    // Crea el JFrame principal.
    container = new JFrame( "GUI Primo Sencillo" );

    // Consigue el panel principal del Frame "container".
    jpanel = ( JPanel ) container.getContentPane();
    //// jpanel.setPreferredSize( new Dimension( maxWinX, maxWinY ) );
    jpanel.setLayout( new GridLayout( 4, 1 ) );

    // Crea y anyade la zona de entrada de datos.
    tempPanel = new JPanel();
    tempPanel.setLayout( new FlowLayout() );
    tempPanel.add( new JLabel( "Numero a estudiar:" ) );
    txfNumero = new JTextField( "", 20 );
    tempPanel.add( txfNumero );
    jpanel.add( tempPanel );

    // Crea y anyade la zona de control (botones).
    tempPanel = new JPanel();
    tempPanel.setLayout( new FlowLayout() );

    btnPulsaAqui = new JButton( "Pulsa aqui" );
    btnPulsaAqui.addActionListener( new ActionListener() {
        public void actionPerformed((ActionEvent e) {
            numVecesPulsado++;
            txfMensajes.setText( "Has pulsado " + numVecesPulsado +
                                " veces el boton 'Pulsa aqui'" );
        }
    } );
    tempPanel.add( btnPulsaAqui );

    btnComienzaCalculo = new JButton( "Comienza calculo" );
    btnComienzaCalculo.addActionListener( new ActionListener() {
        public void actionPerformed( ActionEvent e ) {
            if( txfNumero.getText().trim().length() == 0 ) {
                txfMensajes.setText( "Debes escribir un numero." );
            } else {
                try {
                    // Validacion del numero
                    long numero = Long.parseLong( txfNumero.getText().trim() );
                    // Calculo e impresion en el terminal
                    System.out.println( "Examinando numero: " + numero );
                    boolean primo = esPrimo( numero );
                    if( primo ) {
                        System.out.println( "El numero " + numero + " SI es primo." );
                    } else {
                        System.out.println( "El numero " + numero + " NO es primo." );
                    }
                } catch( NumberFormatException ex ) {
                    txfMensajes.setText( "No es un numero correcto." );
                }
            }
        }
    } );
}

```

```

        tempPanel.add( btnComienzaCalculo );
/* ===== FIN CODIGO A MODIFICAR EN EJERCICIO 5.2 ===== */

jpanel.add( tempPanel );

// Crea y anyade la zona de mensajes.
tempPanel = new JPanel();
tempPanel.setLayout( new FlowLayout() );
tempPanel.add( new JLabel( "Mensajes: " ) );
txfMensajes = new JTextField( "", 30 );
tempPanel.add( txfMensajes );
jpanel.add( tempPanel );

// Crea e inserta el cuadro de texto de sugerencias.
txfSugerencias = new JTextField( 40 );
txfSugerencias.setEditable( false );
txfSugerencias.setText( "321534781, 433494437, 780291637, 1405695061, 2971215073" );
tempPanel = new JPanel();
tempPanel.setLayout( new FlowLayout() );
tempPanel.add( new JLabel( "Sugerencias: " ) );
tempPanel.add( txfSugerencias );
jpanel.add( tempPanel );

// Fija características del container.
container.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
container.pack();
container.setResizable( false );
container.setVisible( true );

System.out.println( "% End of routine: go.\n" );
}

// =====
static boolean esPrimo( long num ) {
    boolean primo;
    if( num < 2 ) {
        primo = false;
    } else {
        primo = true;
        long i = 2;
        while( ( i < num ) && ( primo ) ) {
            primo = ( num % i != 0 );
            i++;
        }
    }
    return( primo );
}
}

```


- 5.3) Con el nuevo código modificado repite el proceso inicial (pulsas varias veces el botón **Pulsa aquí**, a continuación teclea algún número primo grande, e inmediatamente después vuelve a pulsar varias veces el botón **Pulsa aquí**). ¿Qué ocurre? ¿Es interactiva la interfaz ahora?

.....

- 5.4) ¿Las hebras auxiliares deberían ser definidas del tipo “Daemon”? ¿Cómo varía el comportamiento de la interfaz gráfica si se define o no a las hebras como de tipo “Daemon”? Razona tu respuesta.

.....

Las hebras indican si un número es primo o no realizando una llamada al método `System.out.println`. Sería más conveniente que las hebras indicaran si un número es o no primo mediante un mensaje en el cuadro de texto de la interfaz gráfica etiquetado con el texto **Mensajes:**.

Desafortunadamente, una hebra auxiliar creada por el programador no puede *modificar* la interfaz gráfica dado que los métodos de los objetos gráficos de AWT y Swing no son *thread-safe*. Por ello, en este caso la hebra auxiliar se limita a escribir el resultado en la salida estándar. Más adelante se estudiará una solución a este problema.

<p>Nota: Esta entrega forma parte de la evaluación de la asignatura. Debe ser guardado por el estudiantado junto con el resto de entregas en una carpeta. El profesorado podrá pedir al estudiantado que le entregue dicha carpeta en cualquier momento.</p>
--