Nombre y apellido: Enric Gil Gallen

Nombre y apellido: Victor Granados Segara

Tiempo en casa: 1:00

Tema 04. Conceptos Básicos de Concurrencia en Java

1 Se desea crear y arrancar dos hebras que se ejecuten concurrentemente. Cada hebra debe disponer de un identificador entero único, cuya secuencia comienza por cero. Además, cada hebra debe escribir en pantalla mil veces su identificador.

1.1) Crea las hebras a partir de una subclase de la clase `Thread`.
Escribe a continuación el código completo.

```
class MiHebra extends Thread {
 int miId;
 public MiHebra( int miId ) {
   this.miId = miId;
 }

 public void run() {
   int num_impresinos = 1000;
   for (int i = 0; i < num_impresinos; i++ ){
     System.out.println("Mi identificador de hebra es: "+ miId);
   }
 }
}

class EjemploCreacionThread {
 public static void main( String args[] ) {
   new MiHebra(0).start();
   new MiHebra(1).start();
 }
}
```

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**.
Escribe a continuación el código completo.

```
class MiRun implements Runnable {
 int miId;
 public MiRun( int miId ) {
   this.miId = miId;
 }

 public void run() {
   int num_impresinos = 1000;
   for (int i = 0; i < num_impresinos; i++ ){
     System.out.println("Mi identificador de hebra es: "+ miId);
   }
 }
}

class EjemploCreacionRunnable {
 public static void main( String args[] ) {
   new Thread(new MiRun(0)).start();
   new Thread(new MiRun(1)).start();
 }
}
```

1.3) Al probar los códigos, ¿las dos hebras se ejecutan concurrentemente? Razona tu respuesta.

- Si, ya que al usar el método **.start()** cuando una hebra está en ejecución la otra pasa a ejecutarse, esto se puede observar en el resultado donde se van intercalando las respuestas

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

- Se ejecutaría la hebra 0, deteniendo la ejecución de la hebra madre lo que impidió que se ejecutara la hebra 1, por lo tanto primero se mostrarán los primeros 1000 resultados con el Id siendo 0 y luego los otros 1000 Id siendo 1.

1.5) Realiza el cambio anterior, ejecuta el código y describe qué ocurre.

- Efectivamente lo descrito en el punto anterior: “primero se mostrarán los primeros 1000 resultados con el Id siendo 0 y luego los otros 1000 Id siendo 1 ”

2 Se desea crear y arrancar dos hebras que se ejecuten concurrentemente. Cada hebra debe disponer de un identificador entero único, cuya secuencia comienza por cero. Además, el código de cada hebra debe incluir un bucle para calcular la suma de los números comprendidos entre dos números que recibe en el constructor. Finalmente, las hebras deben imprimir un mensaje con su identificador y la suma calculada.

Además, el programa principal debe imprimir un mensaje al inicio del programa, arrancar las hebras para que realicen un millón de incrementos, y finalizar con la impresión de otro mensaje.

2.1) Escribe el código de la clase hebra

2.2) Escribe el código del programa principal

```
public class Ejercicio2 {
    public static void main( String args[] ){
        MiHebraEje2 h0 = new MiHebraEje2(0, 1,10);
        MiHebraEje2 h1 = new MiHebraEje2(1,1,2);
        h0.start();
        System.out.println("Empieza hebra 1");
        h1.start();
        System.out.println("Empieza hebra 2");
        try{
            h0.join();
            h1.join();
        }catch (InterruptedException ex){
            ex.printStackTrace();
        }
        System.out.println("Acaba hebra 1");
        System.out.println("Acaba hebra 2");

    }

}

class MiHebraEje2 extends Thread {
    int miId;
    int num1;
    int num2;
    int actual = 0;
    public MiHebraEje2(int miId, int num1, int num2) {
        this.miId = miId;
        this.num1 = num1;
        this.num2 = num2;
    }

    public void run() {
        for (int t = 0; t < 1000000; t++){
            for (int i = num1; i < num2; i++){
                actual += i;
            }
        }

        System.out.println("Mi hebra es: "+miId+" -- Suma: " + actual);
    }
}
```

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.

- Si, las hebras auxiliares puesto que tienen la función .join para esperar a que acaben, aunque si no estuvieran igualmente la hebra acaba la última pero los mensajes podrían no salir en el orden esperado.

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.

- Las hebras hijas ni empiezan su ejecución ya que en el momento que van a empezar la hebra principal ya ha acabado (Suponiendo que quitamos los .join)

2.5) Incluye las órdenes necesarias para que el programa principal espere la finalización de las dos hebras antes de realizar la impresión final. ¿Cómo se altera la ejecución? ¿El comportamiento varía de usar hebras de tipo "Daemon" o de tipo no "Daemon"? Razona tus respuestas.

- Usando un **.join,** para que se espere
- El comportamiento no variaría si fuera Daemon o no. Ya que igualmente las hebras hijas acabarían antes de que acabe la principal.

3 Se desea crear y arrancar un conjunto de hebras. Cada hebra debe disponer de un identificador entero único, cuya secuencia comienza por cero.

Cada hebra debe realizar un millón de incrementos sobre un objeto compartido recibido. Además, cada hebra debe imprimir 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 sobre el cual se realizarán los incrementos. 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 comprobarlos (su número y tipo) de forma similar.

```
// =============================================================
class CuentaIncrementos1a {
// =============================================================
  long contador = 0;

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

  // -----------------------------------------------------------
  long dameContador() {
    return( contador );
  }
}
```

```
// =============================================================
class EjemploIncrementos1a {
// =============================================================

  // -----------------------------------------------------------
  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 ] );
    } catch( NumberFormatException ex ) {
      numHebras = -1;
      System.out.println( "ERROR: Argumentos numericos incorrectos." );
      System.exit( -1 );
    }

    System.out.println( "numHebras: " + numHebras );
  }
}
```

```
class miHebra extends Thread{
  int miId;
  CuentaIncrementos1a contador;

  public miHebra(int miId, CuentaIncrementos1a contador){
    this.miId = miId;
    this.contador = contador;
  }

  public void run(){
    System.out.println("Empiezo hebra: " + miId);
    for(int i = 0; i<1000000; i++){
      contador.incrementaContador();
    }
    System.out.println("Termino hebra: " + miId);
  }
}
```

3.2) Escribe un programa principal que realice las siguientes tareas, en el siguiente orden:

1. El programa principal debe averiguar el número de hebras que debe crear. Este número debe ser pasado al programa en la línea de argumentos de la forma habitual. Por ejemplo, el comando `java EjemploIncrementos 4` deberá crear 4 hebras.
   Si el número de argumentos de la línea de argumentos no es correcto, el programa debe avisar y terminar. Asimismo, si algún argumento no es correcto (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.

5. El programa principal debe esperar a que todas las hebras terminen.
   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.

```
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 ] );
  } catch( NumberFormatException ex ) {
    numHebras = -1;
    System.out.println( "ERROR: Argumentos numericos incorrectos." );
    System.exit( -1 );
  }

  System.out.println( "numHebras: " + numHebras );

  CuentaIncrementos1a contador = new CuentaIncrementos1a();
  System.out.println("Valor inicial: "+contador.dameContador());
  miHebra[] vectorHebra = new miHebra[numHebras];
  for (int i = 0; i < numHebras; i++){
    vectorHebra[i] = new miHebra(i, contador);
    vectorHebra[i].start();
  }

  try{
    for (int i = 0; i < numHebras; i++){
      vectorHebra[i].join();
    }
  }catch (InterruptedException ex){
    ex.printStackTrace();
  }


  System.out.println("Valor final del contador: "+contador.dameContador());
 }
}
```

3.3) Comprueba si hay concurrencia entre las hebras. ¿Cómo puedes demostrarlo?

- Se observa porque en los mensajes de inicio y fin de cada hebra están alterados

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

- 4 * 1000000 = 4000000 pero se muestra el 2763960

3.5) ¿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.

- Es un problema de atomicidad, la variable no le da tiempo a actualizarse

4 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`.

4.1) Realiza las siguientes acciones de modo consecutivo: Pulsa varias veces el botón `Pulsa aquí`. A continuación teclea algún número primo grande (321534781, 433494437, 780291637, 1405695061, 2971215073, etc.) y pulsa el botón de `Comienza calculo`. Inmediatamente después de lanzar el cálculo, vuelve a pulsar varias veces el botón `Pulsa aquí`. ¿Qué ocurre? ¿Es interactiva la interfaz?

No, podemos apreciar que no es interactiva.

4.2) Modifica la aplicación para que cada número que se desee examinar sea evaluado por una nueva hebra. La validez del número debe realizarla el programa principal, mientras que las hebras únicamente deben evaluar si un número válido es primo e imprimir el resultado. Escribe a continuación la parte de tu código que realiza tal tarea: la definición de la nueva clase hebra y la modificación del gestor de evento correspondiente.

```
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 {
         long numero = Long.parseLong( txfNumero.getText().trim() );

         miHebraEje4 h = new miHebraEje4(numero);
         h.start();

       } catch( NumberFormatException ex ) {
         txfMensajes.setText( "No es un numero correcto." );
       }
     }
   }
 }
);

public class miHebraEje4 extends Thread {
   long numero;

   public miHebraEje4(long num) {
```

```
        this.numero = num;
    }

    public void run(){
        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." );
        }
    }
}
```

4.3) Con el nuevo código modificado repite el proceso inicial (pulsa 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?

Sí, se puede comprobar que es totalmente interactiva.

4.4) ¿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.

Depende de si quieres que se pueda cerrar el programa antes de saber si un número es primo o no.