EI1024/MT1024 “Programación Concurrente y Paralela” 2022–23

Nombre y apellidos (1): ....

Nombre y apellidos (2): ....

Tiempo empleado para tareas en casa en formato *h:mm* (obligatorio): ....

Entregable
para
Laboratorio

la03_g

Tema 05. El Problema de la Visibilidad en Java

Tema 06. El Problema de la Atomicidad en Java

1 Estudia el siguiente código y responde a las siguientes preguntas.

```
// ============================================================================
class CuentaIncrementos {
// ============================================================================
  int numIncrementos = 0;

  // --------------------------------------------------------------------------
  void incrementaNumIncrementos() {
    numIncrementos++;
  }

  // --------------------------------------------------------------------------
  int dameNumIncrementos() {
    return( numIncrementos );
  }
}


// ============================================================================
class MiHebra extends Thread {
// ============================================================================
  int                  tope;
  CuentaIncrementos  c;

  // --------------------------------------------------------------------------
  public MiHebra( int tope, CuentaIncrementos c ) {
    this.tope  = tope;
    this.c     = c;
  }

  // --------------------------------------------------------------------------
  public void run() {
    for( int i = 0; i < tope; i++ ) {
      c.incrementaNumIncrementos();
    }
  }
}

// ============================================================================
class EjemploCuentaIncrementos {
// ============================================================================

  // --------------------------------------------------------------------------
```

```
  public static void main( String args[] ) {
    long     t1, t2;
    double   tt;
    int      numHebras, tope;

    // Comprobacion y extraccion de los argumentos de entrada.
    if( args.length != 2 ) {
      System.err.println( "Uso: java programa <numHebras> <tope>" );
      System.exit( -1 );
    }
    try {
      numHebras = Integer.parseInt( args[ 0 ] );
      tope      = Integer.parseInt( args[ 1 ] );
    } catch( NumberFormatException ex ) {
      numHebras = -1;
      tope      = -1;
      System.out.println( "ERROR: Argumentos numericos incorrectos." );
      System.exit( -1 );
    }

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

    System.out.println( "Creando y arrancando " + numHebras + " hebras." );
    t1 = System.nanoTime();
    MiHebra v[] = new MiHebra[ numHebras ];
    CuentaIncrementos c = new CuentaIncrementos();
    for( int i = 0; i < numHebras; i++ ) {
      v[ i ] = new MiHebra( tope, c );
      v[ i ].start();
    }
    for( int i = 0; i < numHebras; i++ ) {
      try {
        v[ i ].join();
      } catch( InterruptedException ex ) {
        ex.printStackTrace();
      }
    }
    t2 = System.nanoTime();
    tt = ( ( double ) ( t2 - t1 ) ) / 1.0e9;
    System.out.println( "Total de incrementos: " + c.dameNumIncrementos() );
    System.out.println( "Tiempo transcurrido en segs.: " + tt );
  }
}
```

1.1) ¿Qué realiza el código? ¿Qué debería mostrar en pantalla si se ejecutase con los parámetros hebras 4 y tope 1 000 000?

. . . .

. . . .

. . . .

1.2) Compila y ejecuta el código con dichos valores en tu ordenador local. ¿Qué muestra realmente en pantalla si se ejecuta con los parámetros hebras 4 y tope 1 000 000?

. . . .

. . . .

. . . .

1.3) ¿Es un código *thread-safe*? Justifica tu respuesta.

....

....

....

....

....

1.4) Crea una **copia del código original** e inserta en la copia el modificador `volatile` en la variable `numIncrementos` de la clase `CuentaIncrementos`.

A continuación, compila y prueba el nuevo código.

¿Resuelve el problema el modificador `volatile`? ¿Por qué?

....

....

....

....

....

1.5) ¿Se podría resolver con el modificador `synchronized`?

Para ello, crea una **copia del código original** y aplica el modificador `synchronized` sobre cada una de las rutinas de la clase `CuentaIncrementos`.

Después compila y prueba el código, antes de contestar a la pregunta anterior.

Escribe a continuación los cambios realizados en la clase `CuentaIncrementos`.

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

1.6) ¿Se puede arreglar empleando clases y operadores atómicos?

Para ello, crea otra **copia del código original**, **ELIMINA COMPLETAMENTE** la clase `CuentaIncrementos` y utiliza en su lugar una **clase atómica y sus métodos**.

Después compila y prueba el código, antes de contestar la pregunta.

Escribe a continuación los cambios realizados en el código.

1.7) Completa la siguiente tabla con datos de todas las versiones anteriores en tu ordenador, utilizando hebras 4 y un tope de 1 000 000. Comenta los resultados.

Código	Total incrementos
Código original	
Código con `volatile`	
Código con `synchronized`	
Código con clases atómicas	

2 Se desea imprimir en pantalla aquellos números primos contenidos en un vector.

El código completo es el siguiente:

```
// ===========================================================================
public class EjemploMuestraPrimosEnVector {
// ===========================================================================

  // -------------------------------------------------------------------------
  public static void main( String args[] ) {
    int     numHebras, vectOpt;
    boolean option = true;
    long    t1, t2;
    double  ts, tc, tb, td;

    // Comprobacion y extraccion de los argumentos de entrada.
    if( args.length != 2 ) {
      System.err.println( "Uso: java programa <numHebras> <vectorOption>" );
      System.exit( -1 );
    }
    try {
      numHebras = Integer.parseInt( args[ 0 ] );
      vectOpt   = Integer.parseInt( args[ 1 ] );
      if ( ( vectOpt != 0 ) && ( vectOpt != 1 ) ) {
        System.out.println( "ERROR: vectorOption should be 0 or 1.");
        System.exit( -1 );
      } else {
        option = (vectOpt == 0);
      }
    } catch( NumberFormatException ex ) {
      numHebras = -1;
      System.out.println( "ERROR: Argumentos numericos incorrectos." );
      System.exit( -1 );
    }

    //
    // Eleccion del vector de trabajo
    //
    VectorNumeros vn = new VectorNumeros (option);
    long vectorNumeros[] = vn.vector;

    //
    // Implementacion secuencial.
    //
    System.out.println( "" );
    System.out.println( "Implementacion secuencial." );
    t1 = System.nanoTime();
    for( int i = 0; i < vectorNumeros.length; i++ ) {
      if( esPrimo( vectorNumeros[ i ] ) ) {
        System.out.println( "  Encontrado primo: " + vectorNumeros[ i ] );
      }
    }
    t2 = System.nanoTime();
    ts = ( ( double ) ( t2 - t1 ) ) / 1.0e9;
    System.out.println( "Tiempo secuencial (seg.):                    " + ts );
/*
    //
    // Implementacion paralela ciclica.
    //
    System.out.println( "" );
    System.out.println( "Implementacion paralela ciclica." );
    t1 = System.nanoTime();
```

```
    // Gestion de hebras para la implementacion paralela ciclica
    // ....
    t2 = System.nanoTime();
    tc = ( ( double ) ( t2 - t1 ) ) / 1.0e9;
    System.out.println( "Tiempo paralela ciclica (seg.):              " + tc );
    System.out.println( "Incremento paralela ciclica:                 " + ... );
    //
    // Implementacion paralela por bloques.
    //
    // ....
    //
    // Implementacion paralela dinamica.
    //
    // ....
*/
  }

  // --------------------------------------------------------------------------
  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 );
  }
}

// ============================================================================
class VectorNumeros {
// ============================================================================
  long    vector[];

  // --------------------------------------------------------------------------
  public VectorNumeros (boolean caso) {
    if (caso) {
      vector = new long [] {
      200000033L, 200000039L, 200000051L, 200000069L,
      200000081L, 200000083L, 200000089L, 200000093L,
      200000107L, 200000117L, 200000123L, 200000131L,
      200000161L, 200000183L, 200000201L, 200000209L,
      4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
```

```
      4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L
                        };
    } else {
      vector = new long [] {
      200000033L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      200000039L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      200000051L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      200000069L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      200000081L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      200000083L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      200000089L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      200000093L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      200000107L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      200000117L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      200000123L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      200000131L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      200000161L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      200000183L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      200000201L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      200000209L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L
      };
    }
  }
}
```

2.1) Compila y ejecuta el programa anterior, utilizando un 0 como segundo parámetro. En este caso se trabajo con el siguiente vector de números:

```
long vectorNumeros[] = {
      200000033L, 200000039L, 200000051L, 200000069L,
      200000081L, 200000083L, 200000089L, 200000093L,
      200000107L, 200000117L, 200000123L, 200000131L,
      200000161L, 200000183L, 200000201L, 200000209L,
      4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L
   };
```

¿Cuáles son los números primos contenidos en el vector?

....

....

....

....

....

2.2) Realiza una implementación paralela con distribución cíclica, en la que cada hebra procese un conjunto de elementos del vector. Para cada elemento del vector procesado, se mostrará su valor SOLO si es primo.

Incluye la gestión de hebras de esta versión a continuación de la implementación secuencial.

Comprueba que los números primos mostrados en la versión paralela coinciden con los de la versión secuencial.

Escribe, a continuación, la parte de tu código que realiza tal tarea: la definición de la clase `MiHebraPrimoDistCiclica` y el código a incluir en el programa principal que permite gestionar los objetos de esta clase.

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

ATENCIÓN: Los ejercicios anteriores deben realizarse en casa. Los siguientes, en el aula.

2.3) Realiza una implementación paralela con distribución por bloques, en la que cada hebra procese un conjunto de elementos del vector. Para cada elemento del vector procesado, se mostrará su valor SOLO si es primo.

Incluye la gestión de hebras de esta versión a continuación de la implementación cíclica.

Comprueba que los números primos mostrados en la versión paralela coinciden con los de la versión secuencial.

Escribe, a continuación, la parte de tu código que realiza tal tarea: la definición de la clase `MiHebraPrimoDistPorBloques` y el código a incluir en el programa principal que permite gestionar los objetos de esta clase.

2.4) Realiza una implementación paralela con distribución dinámica, utilizando un número entero atómico (`AtomicInteger`), que apunta a una posición en el vector. Las hebras recibe un único objeto de este tipo, que siempre contiene la primera posición del vector sin procesar. Para ello, las hebras deben **realizar de modo atómico, la lectura del valor actual y su incremento**. Las hebras finalizarán cuando el índice sobrepase la dimensión del vector.

Incluye la gestión de hebras de esta versión a continuación de la implementación por bloques.

Comprueba que los números primos mostrados en la versión paralela coinciden con los de la versión secuencial.

Escribe, a continuación, la parte de tu código que realiza tal tarea: la definición de la clase `MiHebraPrimoDistDinamica` y el código a incluir en el programa principal que permite gestionar los objetos de esta clase.

2.5) Completa la siguiente tabla, obteniendo los resultados para 4 hebras en el ordenador del aula y los resultados para 16 hebras en patan. Redondea los tiempos dejando sólo tres decimales y redondea los incrementos dejando dos decimales.

	4 hebras		16 hebras	
	Tiempo	Incremento	Tiempo	Incremento
Secuencial		—		—
Paralela con distribución cíclica				
Paralela con distribución por bloques				
Paralela con distribución dinámica				

....

2.6) Justifica los resultados de la tabla anterior.

....
....
....
....

2.7) Evalúa y compara las tres versiones (secuencial, paralela cíclica y paralela por bloques), pero en este caso utilizando 1 como segundo parámetro, es decir, manejando el vector:

```
long vectorNumeros[] = {
      200000033L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      200000039L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      200000051L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      200000069L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      200000081L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      200000083L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      200000089L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      200000093L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      200000107L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      200000117L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      200000123L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      200000131L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      200000161L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      200000183L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      200000201L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
      200000209L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L
   };
```

Completa la siguiente tabla, obteniendo los resultados para 4 hebras en el ordenador del aula y los resultados para 16 hebras en patan. Redondea los tiempos dejando sólo tres decimales y redondea los incrementos dejando dos decimales.

	4 hebras		16 hebras	
	Tiempo	Incremento	Tiempo	Incremento
Secuencial		—		—
Paralela con distribución cíclica				
Paralela con distribución por bloques				
Paralela con distribución dinámica				

....

2.8) Justifica los resultados de la tabla anterior.

. . . .

. . . .

. . . .

. . . .

. . . .

2.9) ¿Cuál es la mejor distribución con ambos vectores? Justifica tu respuesta.

. . . .

. . . .

. . . .

. . . .

. . . .

3 Empleando el ordenador del aula, completa la siguiente tabla con datos de todas las versiones desarrolladas en el ejercicio 1, utilizando hebras 4 y un tope de 10 000 000. Redondea los tiempos dejando sólo tres decimales y comenta los resultados.

Código	Total incrementos	Tiempo transcurrido (seg.)
Código original		
Código con `volatile`		
Código con `synchronized`		
Código con clases atómicas		

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .