El1024/MT1024 "Programación Concurrente y Paralela"	2025–26	Entregable
Nombre y apellidos (1):		para Laboratorio
Nombre y apellidos (2):		
Tiempo empleado para tareas en casa en formato h:mm (obligatorio):		la03 ₋ g

Tema 04. El Problema de la Visibilidad en Java

Tema 05. 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
                       numIters;
    CuentaIncrementos
    public MiHebra( int numIters, CuentaIncrementos c ) {
        {f this}.numIters = numIters;
        this.c
                     = c;
    }
    public void run() {
        for ( int i = 0; i < numIters; i++ ) {
            c.incrementaNumIncrementos();
    }
}
class EjemploCuentaIncrementos {
   // ==
```

```
public static void main( String args[] ) {
        long
                t1, t2;
        double
                tt;
        int
                numHebras, numIters;
        // Comprobacion y extraccion de los argumentos de entrada.
        \mathbf{if} (args.length!= 2) {
            System.err.println("Uso: java programa <numHebras> <numIters>");
            System. exit (-1);
        try {
            numHebras = Integer.parseInt( args[ 0 ] );
numIters = Integer.parseInt( args[ 1 ] );
            if( (numHebras <= 0) || (numIters <= 0) ) {}
                System.err.print("Uso: [ java programa <numHebras> <n> ] " );
                System.err.println("donde ( numHebras > 0 ) y ( numIters > 0 )");
                System.exit(-1);
        } catch( NumberFormatException ex ) {
            numHebras = -1;
            numIters = -1;
            System.out.println("ERROR: Argumentos numericos incorrectos.");
            System. exit (-1);
        }
        System.out.println("numHebras: " + numHebras );
        System.out.println("numIters : " + numIters );
        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( numIters, 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 numIters 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 num Iters 1 000 000?
1.3)	¿Es un código thread-safe? Justifica tu respuesta.
1 4)	
1.4)	Crea una copia del código original (EjemploCuentaIncrementosVolatile.java) e inserta 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?
	¿Resuerve el problema el modificador volacife: ¿i or que:
1.5)	¿Se podría resolver con el modificador synchronized? Para comprobarlo, crea una copia del código original (EjemploCuentaIncrementosSyn-
1.5)	¿Se podría resolver con el modificador synchronized? Para comprobarlo, crea una copia del código original (EjemploCuentaIncrementosSynchronized.java) 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.
1.5)	¿Se podría resolver con el modificador synchronized? Para comprobarlo, crea una copia del código original (EjemploCuentaIncrementosSynchronized.java) 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.
1.5)	¿Se podría resolver con el modificador synchronized? Para comprobarlo, crea una copia del código original (EjemploCuentaIncrementosSynchronized.java) 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.
1.5)	¿Se podría resolver con el modificador synchronized? Para comprobarlo, crea una copia del código original (EjemploCuentaIncrementosSynchronized.java) 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.
11.5)	¿Se podría resolver con el modificador synchronized? Para comprobarlo, crea una copia del código original (EjemploCuentaIncrementosSynchronized.java) 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.
11.5)	¿Se podría resolver con el modificador synchronized? Para comprobarlo, crea una copia del código original (EjemploCuentaIncrementosSynchronized.java) 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.
1.5)	¿Se podría resolver con el modificador synchronized? Para comprobarlo, crea una copia del código original (EjemploCuentaIncrementosSynchronized.java) 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.
11.5)	¿Se podría resolver con el modificador synchronized? Para comprobarlo, crea una copia del código original (EjemploCuentaIncrementosSynchronized.java) 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.
11.5)	¿Se podría resolver con el modificador synchronized? Para comprobarlo, crea una copia del código original (EjemploCuentaIncrementosSynchronized.java) 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.
11.5)	¿Se podría resolver con el modificador synchronized? Para comprobarlo, crea una copia del código original (EjemploCuentaIncrementosSynchronized.java) 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.
11.5)	¿Se podría resolver con el modificador synchronized? Para comprobarlo, crea una copia del código original (EjemploCuentaIncrementosSynchronized.java) 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.
11.5)	¿Se podría resolver con el modificador synchronized? Para comprobarlo, crea una copia del código original (EjemploCuentaIncrementosSynchronized.java) 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.
1.5)	¿Se podría resolver con el modificador synchronized? Para comprobarlo, crea una copia del código original (EjemploCuentaIncrementosSynchronized.java) 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.
1.5)	¿Se podría resolver con el modificador synchronized? Para comprobarlo, crea una copia del código original (EjemploCuentaIncrementosSynchronized.java) 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.
1.5)	¿Se podría resolver con el modificador synchronized? Para comprobarlo, crea una copia del código original (EjemploCuentaIncrementosSynchronized.java) 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.
1.5)	¿Se podría resolver con el modificador synchronized? Para comprobarlo, crea una copia del código original (EjemploCuentaIncrementosSynchronized.java) 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.
1.5)	¿Se podría resolver con el modificador synchronized? Para comprobarlo, crea una copia del código original (EjemploCuentaIncrementosSynchronized.java) 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.
1.5)	¿Se podría resolver con el modificador synchronized? Para comprobarlo, crea una copia del código original (EjemploCuentaIncrementosSynchronized.java) 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.
1.5)	¿Se podría resolver con el modificador synchronized? Para comprobarlo, crea una copia del código original (EjemploCuentaIncrementosSynchronized.java) 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)	Para comprobarlo, c NA la clase Cuenta: Después, compila y	arreglar empleando clases y o crea otra copia del código o Incrementos y utiliza en su lu prueba el código, antes de cor ón los cambios realizados en e	priginal (EjemploAt gar una clase atóm ntestar la pregunta.	omic.java), ELIMI -
			• • • • • • • • • • • • • • • • • • • •	
1.7)		te tabla con datos de todas l y un numIters de 1 000 000. C		
		Código	Total incrementos	
		Código original		
		Código con volatile		
		Código con synchronized		
		Código con clases atómicas		
				1
			• • • • • • • • • • • • • • • • • • • •	
			• • • • • • • • • • • • • • • • • • • •	
			• • • • • • • • • • • • • • • • • • • •	
			• • • • • • • • • • • • • • • • • • • •	

2 Se desea imprimir en pantalla los números primos que aparecen en un vector.

El código completo es el siguiente:

```
public class EjemploMuestraPrimosEnVector {
// =
  //
  public static void main( String args[] ) {
            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> <vectOpt>");
      System. exit (-1);
    }
    try {
      numHebras = Integer.parseInt( args[ 0 ] );
      vectOpt = Integer.parseInt( args[ 1 ] );
      if( (numHebras <= 0 ) || ( (vectOpt != 0 ) && (vectOpt != 1 ) ) ){}
           System.err.print("Uso: [ java programa <numHebras> <vecOpt> ] " );
           System.err.println("donde (numHebras > 0) y (vectOpt es 0 o 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 vectorTrabajo[] = vn.vector;
    // Implementacion secuencial.
    System.out.println("");
    System.out.println("Implementacion secuencial.");
    t1 = System.nanoTime();
    \label{eq:for_state} \textbf{for} \left( \begin{array}{ccc} \textbf{int} & \textbf{i} &= & 0; & \textbf{i} &< & \text{vectorTrabajo.length}; & \textbf{i++} \end{array} \right) \; \left\{ \right.
      if( esPrimo( vectorTrabajo[ i ] ) ) {
        System.out.println(" Encontrado primo: " + vectorTrabajo[ 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
  // (A) \dots
  t2 = System.nanoTime();
  tc = ((double)(t2 - t1)) / 1.0e9;
  System.out.println("Tiempo paralela ciclica (seg.):
                                               " + tc);
                                               " + ... ); // (B)
  System.out.println("Incremento paralela ciclica:
    Implementacion paralela por bloques.
  // (C) \dots
    Implementacion paralela dinamica.
  // (D) ....
 static boolean esPrimo( long num ) {
  boolean cond;
  if (num < 2)
   cond = false;
  } else {
   cond = true;
   long i = 2;
    \mathbf{while}(\ (\ i\ <\ \mathrm{num}\ )\&\&(\ \mathrm{cond}\ )\ )\ \{
     \mathrm{cond} \; = \; (\;\; \mathrm{num} \;\; \% \;\; i \;\; != \;\; 0 \;\;\;) \, ; \label{eq:cond}
     i++;
  return( cond );
// Definicion de las Clases Hebras
// (E) \dots
class VectorNumeros
      vector[];
 long
 public VectorNumeros (boolean caso) {
  if (caso) {
    vector = new long [] {
    200000081L,\ 200000083L,\ 200000089L,\ 200000093L,
```

```
else {
vector = new long [] {
};
}
}
```

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

```
long vectorTrabajo[] = {
 200000081L, 200000083L, 200000089L, 200000093L,
 200000107L, 200000117L, 200000123L, 200000131L,
 200000161L,\ 200000183L,\ 200000201L,\ 200000209L,
 200000221L, 200000237L, 200000239L, 200000243L,
 };
```

¿Cuáles son los números primos contenidos en el vector?

200000081, 200000083, 200000089, 200000093, 200000107, 200000117, 200000123, 200000131, 200000161, 200000183, 200000201, 200000209, 200000221, 200000237, 200000239, 200000243,

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, SOLO se mostrará su valor si el número es primo.
	Descomenta el código situado debajo de "Implementacion secuencial". Incluye la gestión de hebras que paraleliza el bucle comprendido entre la lectura de t1 y t2 en la versión secuencial, y la expresión que permite calcular el incremento de velocidad. 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 (E) y el código a incluir en el programa principal que permite gestionar los objetos de esta clase (A-B).

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, SOLO se mostrará su valor si el número es primo.
	Crea en (E) una clase nueva hebra para este caso. Replica en (C) el código del programa principal de la "Implementación paralela cíclica", para que se ejecute tras las otras versiones.
	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 (E) y el código a incluir en el programa principal que permite gestionar los objetos de esta clase (A-B).

2.4)	Realiza una implementación paralela con distribución dinámica, que utilice un número entero atómico (AtomicInteger), para apuntar a una posición del vector. Las hebras recibirár un objeto de este tipo, que siempre contendrá 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 Crea en (E) una clase nueva hebra para este caso. Replica en (D) el código del programa principal de la "Implementación paralela cíclica", para que se ejecute tras las otras versiones 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 (E) y el código a incluir en el programa principal que permite gestionar los objetos de esta clase (A-B).

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

	4 hebra	as (aula)	16 hebra	as (karen)
	Tiempo	Incremento	Tiempo	Incremento
Secuencial				
Paralela con distribución cíclica				
Paralela con distribución por bloques				
Paralela con distribución dinámica				

.....

2.6)	Justi	fica los resultad	los de	e la t	abla	ante	erior.										
2.7)		ía y compara l en este caso ut				,			. –			-	_		_	-	
		vectorTrabajo 200000081L, 200000083L, 200000089L, 200000093L, 200000107L, 200000117L, 200000131L, 200000161L, 200000183L, 200000201L, 200000201L, 200000237L, 200000239L, 200000243L, };	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 karen. Redondea los tiempos dejando sólo tres decimales y redondea los incrementos dejando dos decimales.

	4 hebra	as (aula)	16 hebra	as (karen)
	Tiempo	Incremento	Tiempo	Incremento
Secuencial				
Paralela con distribución cíclica				
Paralela con distribución por bloques				
Paralela con distribución dinámica				

• • •				
2.9) ¿C	uál es la mejor distribución co	on ambos vectores? J	ustifica tu respuesta.	
se Er	alicen un trabajo similar. Esta di encillos y terminan antes, sigan t n casos donde los datos pueden ueda caerle todo el trabajo a una	rabajando ayudando a ser muy diversos la d	ı las hebras que aún están calcu	ılando.
desarrol	ndo el ordenador del aula, con ladas en el ejercicio 1, utiliza dejando sólo tres decimales y	ndo hebras 4 y un n	umIters de 10 000 000. Redo	
desarrol	ladas en el ejercicio 1, utiliza dejando sólo tres decimales y	ndo hebras 4 y un n comenta los resultados	umIters de 10 000 000. Redo: dos.	
desarrol	ladas en el ejercicio 1, utiliza dejando sólo tres decimales y Código	ndo hebras 4 y un n	umIters de 10 000 000. Redo	
desarrol	ladas en el ejercicio 1, utiliza dejando sólo tres decimales y	ndo hebras 4 y un n comenta los resultados	umIters de 10 000 000. Redo: dos.	
desarrol	ladas en el ejercicio 1, utilizado dejando sólo tres decimales y Código Código original	ndo hebras 4 y un n comenta los resultados	umIters de 10 000 000. Redo: dos.	