

INTRODUCCIÓN AL PARALELISMO – HILOS

AUTORES:

YORKS GOMEZ – CESAR VÁSQUEZ

PROFESOR:

JAVIER IVAN TOQUICA

ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO

ARQUITECTURAS DE SOTFWARE – GRUPO # 1

INGENIERÍA DE SISTEMAS

BOGOTÁ D.C.

2023

INTRODUCCIÓN

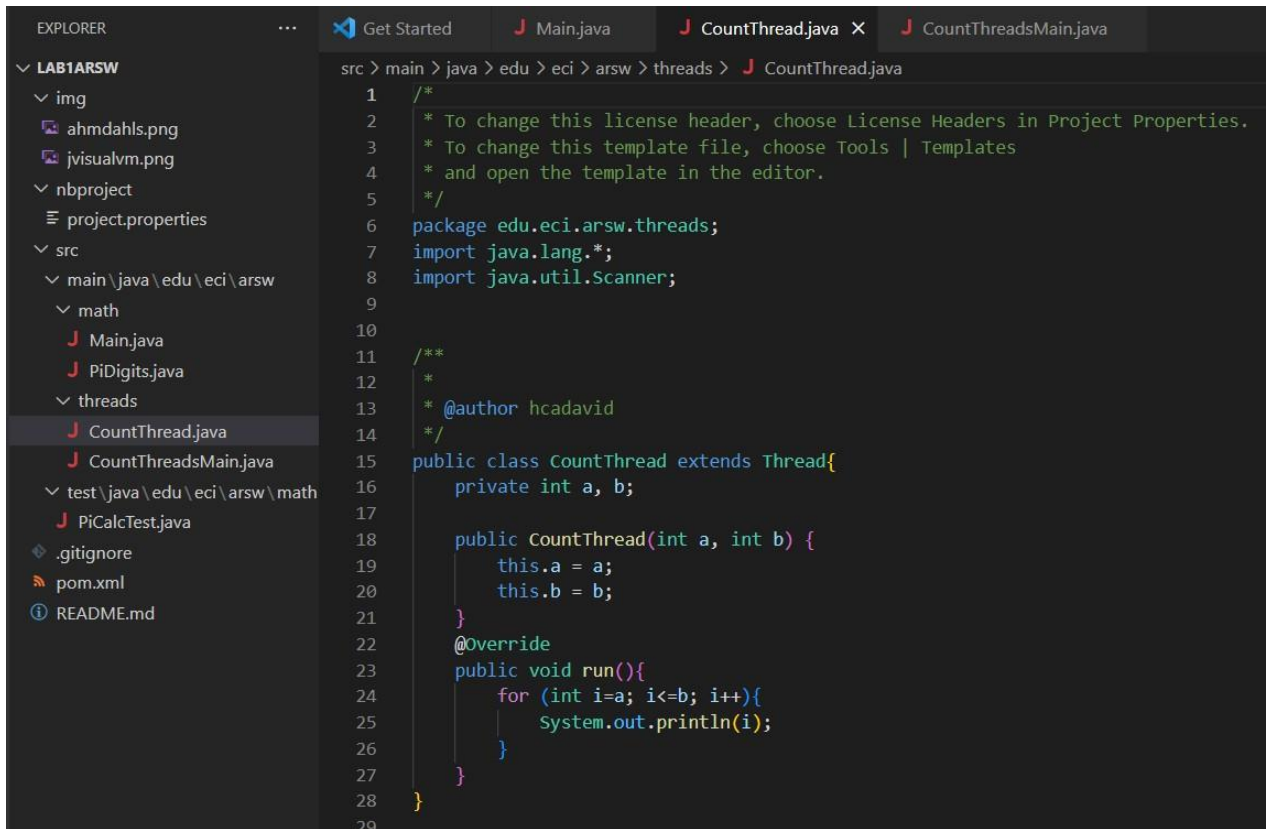
En este informe de laboratorio podremos profundizar y entender de lleno el concepto de paralelismo y ligado al concepto concurrencia, para qué nos sirve, cómo se maneja, de qué consta, etc. Lo podremos realizar a través de las diferentes formas de uso orientadas a **hilos** en java, aprenderemos a usarlas adecuadamente entendiendo el funcionamiento que tiene cada una de ellas y aplicándose cuando sea necesario.

Para ello utilizaremos correspondientemente todas las herramientas dadas, como lo son lecturas, videos, conocimientos, equipos, entre otras.

DESARROLLO

Parte I

- De acuerdo con lo revisado en las lecturas, complete las clases CountThread, para que las mismas definan el ciclo de vida de un hilo que imprima por pantalla los números entre A y B.

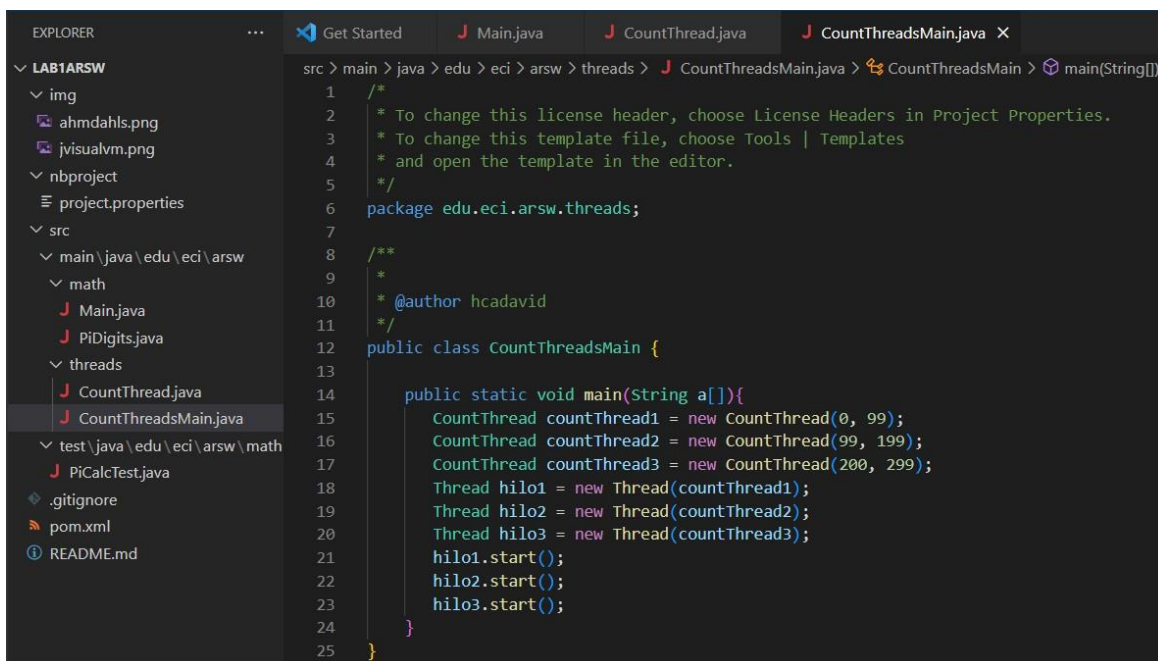


```

1  /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6  package edu.eci.arsw.threads;
7  import java.lang.*;
8  import java.util.Scanner;
9
10
11  /**
12   *
13   * @author hcadavid
14   */
15  public class CountThread extends Thread{
16      private int a, b;
17
18      public CountThread(int a, int b) {
19          this.a = a;
20          this.b = b;
21      }
22
23      @Override
24      public void run(){
25          for (int i=a; i<=b; i++){
26              System.out.println(i);
27          }
28      }
29  }

```

- Complete el método main de la clase CountMainThreads para que:
 - Cree 3 hilos de tipo CountThread, asignándole al primero el intervalo [0..99], al segundo [99..199], y al tercero [200..299].
 - Inicie los tres hilos con 'start()'.

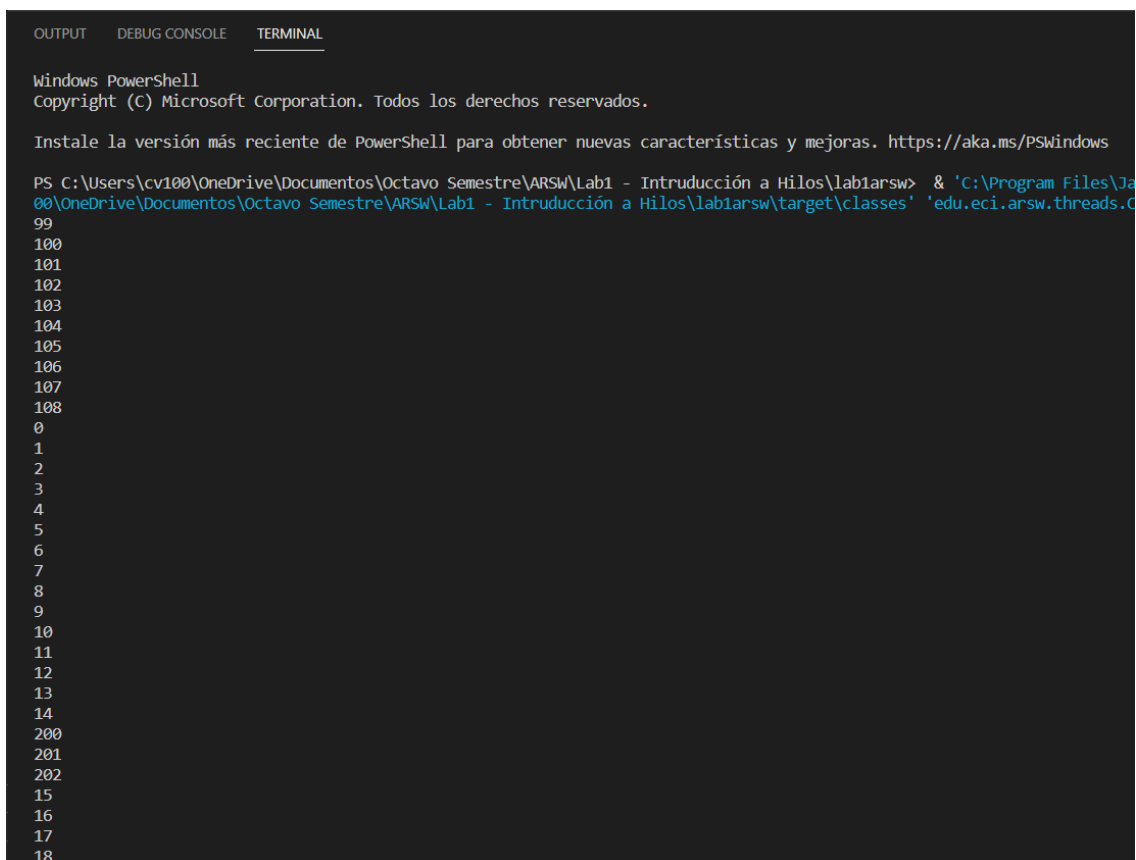


```

1  /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6  package edu.eci.arsw.threads;
7
8  /**
9   *
10   * @author hcadavid
11   */
12  public class CountThreadsMain {
13
14      public static void main(String a[]){
15          CountThread countThread1 = new CountThread(0, 99);
16          CountThread countThread2 = new CountThread(99, 199);
17          CountThread countThread3 = new CountThread(200, 299);
18          Thread hilo1 = new Thread(countThread1);
19          Thread hilo2 = new Thread(countThread2);
20          Thread hilo3 = new Thread(countThread3);
21          hilo1.start();
22          hilo2.start();
23          hilo3.start();
24      }
25  }

```

- Ejecute y revise la salida por pantalla.



```

Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

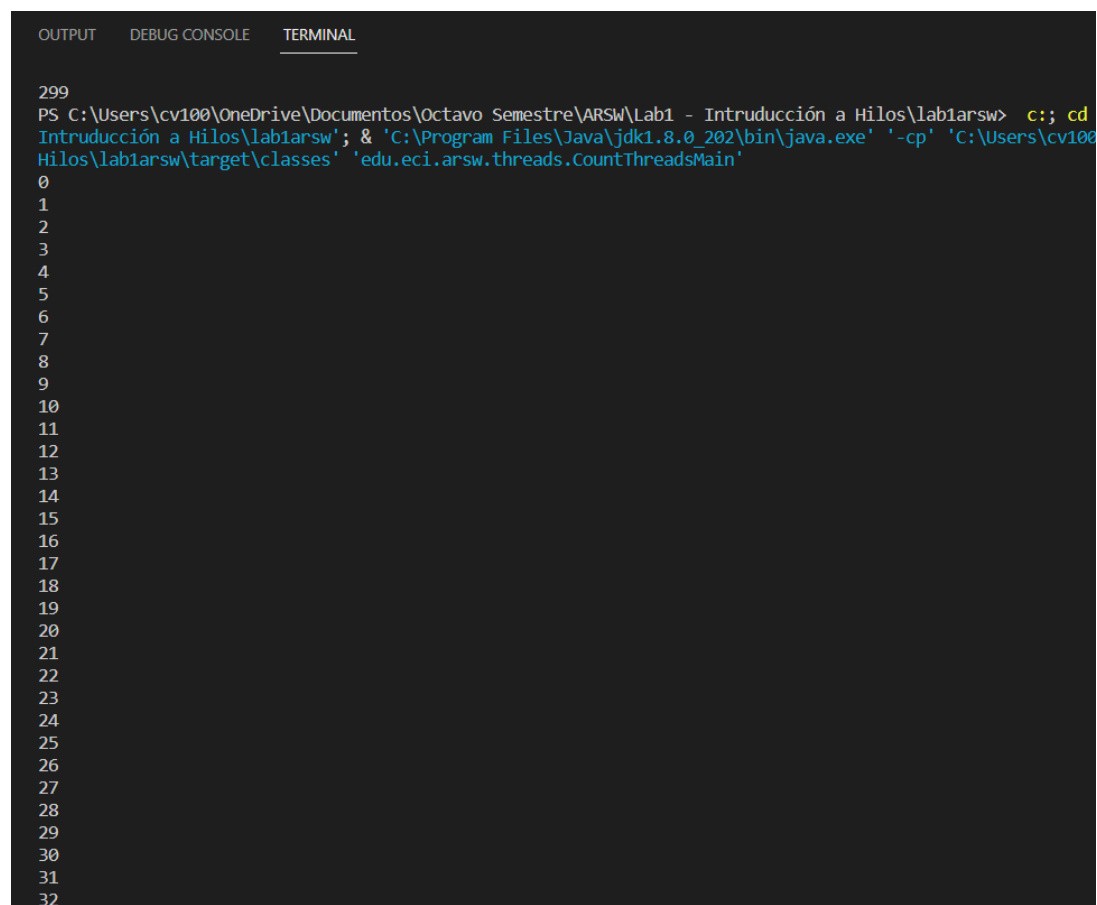
PS C:\Users\cv100\OneDrive\Documentos\Octavo Semestre\ARSW\Lab1 - Intrucción a Hilos\labiarsw> & 'C:\Program Files\Java\jdk-10.0.2\bin\java.exe' -cp 'C:\Program Files\Java\jdk-10.0.2\bin\java.exe' 'edu.eci.arsw.threads.CountThreadsMain'
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
200
201
202
15
16
17
18

```

- Cambie el inicio con 'start()' por 'run()'. ¿Cómo cambia la salida?, por qué?

Cuando utilizamos el método start() para el objeto CountThread lo que está haciendo es crear el hilo y cambiar el estado a Runnable, cuando le decimos al objeto de CountThread start() cierto número de ocasiones, está cambiando el estado de Runnable a Runnable, esto hace que imprima en desorden, pues cada hilo está interrumpiendo a los demás utilizando el medio cierto momento. El método run() lo que hace es ejecutar el hilo sin tener que cambiarle el estado más de una vez, por lo que hace el llamado a un método de una clase esto hace que no haga otra instrucción hasta que acabe el método, luego continuaran los hilos restantes hasta finalizar, por eso se puede ver en orden la ejecución.

Así se ve con el método run():



```
299
PS C:\Users\cv100\OneDrive\Documentos\Octavo Semestre\ARSW\Lab1 - Intruducción a Hilos\lab1arsw> c:; cd
Intruducción a Hilos\lab1arsw'; & 'C:\Program Files\Java\jdk1.8.0_202\bin\java.exe' '-cp' 'C:\Users\cv100
Hilos\lab1arsw\target\classes' 'edu.eci.arsw.threads.CountThreadsMain'
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
```

Parte II

Se agrega una nueva función, `getDigitsSection`, que tiene la funcionalidad de la antigua `getDigits`.

```
public static byte[] getDigitsSection(int start, int count) {
    if (start < 0) {
        throw new RuntimeException( message: "Invalid Interval");
    }

    if (count < 0) {
        throw new RuntimeException( message: "Invalid Interval");
    }

    byte[] digits = new byte[count];
    double sum = 0;

    for (int i = 0; i < count; i++) {
        if (i % DigitsPerSum == 0) {
            sum = 4 * sum( m: 1, n: start)
                - 2 * sum( m: 4, n: start)
                - sum( m: 5, n: start)
                - sum( m: 6, n: start);

            start += DigitsPerSum;
        }

        sum = 16 * (sum - Math.floor( a: sum));
        digits[i] = (byte) sum;
        PiDigits.DONE += 1;
        System.out.println( x: PiDigits.DONE);
    }

    return digits;
}
```

Dentro de `getDigits`, ahora se hace la gestión de hilos pertinente.

```
public static byte[] getDigits(int start, int count, int threads) {
    if (start < 0) {
        throw new RuntimeException( message: "Invalid Interval");
    }

    if (count < 0) {
        throw new RuntimeException( message: "Invalid Interval");
    }

    byte[] digits = new byte[count];
    int threadLength = count / threads;
    int lastThreadLength = threadLength + (count % threads);
    ArrayList<PIThread> pithreads = new ArrayList<>();
    int currentStart = start, currentCount = threadLength;

    for(int i = 0; i < threads; i++) {
        if(i == threads - 1)
            currentCount = lastThreadLength;

        PITHread pithread = new PITHread( start: currentStart, end: currentCount);
        pithreads.add( e: pithread);
        pithread.start();

        currentStart += currentCount;
    }

    try {
```

Se adicionan las pruebas nuevas por número de hilos

```
@Test
public void piGenTestTwoThread() throws Exception {

    byte[] expected = new byte[]{
        0x2, 0x4, 0x3, 0xF, 0x6, 0xA, 0x8, 0x8,
        0x8, 0x5, 0xA, 0x3, 0x0, 0x8, 0xD, 0x3,
        0x1, 0x3, 0x1, 0x9, 0x8, 0xA, 0x2, 0xE,
        0x0, 0x3, 0x7, 0x0, 0x7, 0x3, 0x4, 0x4,
        0xA, 0x4, 0x0, 0x9, 0x3, 0x8, 0x2, 0x2,
        0x2, 0x9, 0x9, 0xF, 0x3, 0x1, 0xD, 0x0,
        0x0, 0x8, 0x2, 0xE, 0xF, 0xA, 0x9, 0x8,
        0xE, 0xC, 0x4, 0xE, 0x6, 0xC, 0x8, 0x9,
        0x4, 0x5, 0x2, 0x8, 0x2, 0x1, 0xE, 0x6,
        0x3, 0x8, 0xD, 0x0, 0x1, 0x3, 0x7, 0x7,};

    for (int start = 0; start < expected.length; start++) {
        for (int count = 0; count < expected.length - start; count++) {
            byte[] digits = PiDigits.getDigits(start, count, threads: 2);
            assertEquals("expected: count, actual: digits.length",
                expected[start + count], digits.length);

            for (int i = 0; i < digits.length; i++) {
                assertEquals(expected[start + i], digits[i]);
            }
        }
    }
}

@Test
public void piGenTestThreeThread() throws Exception {
```

Los tres tests pasan

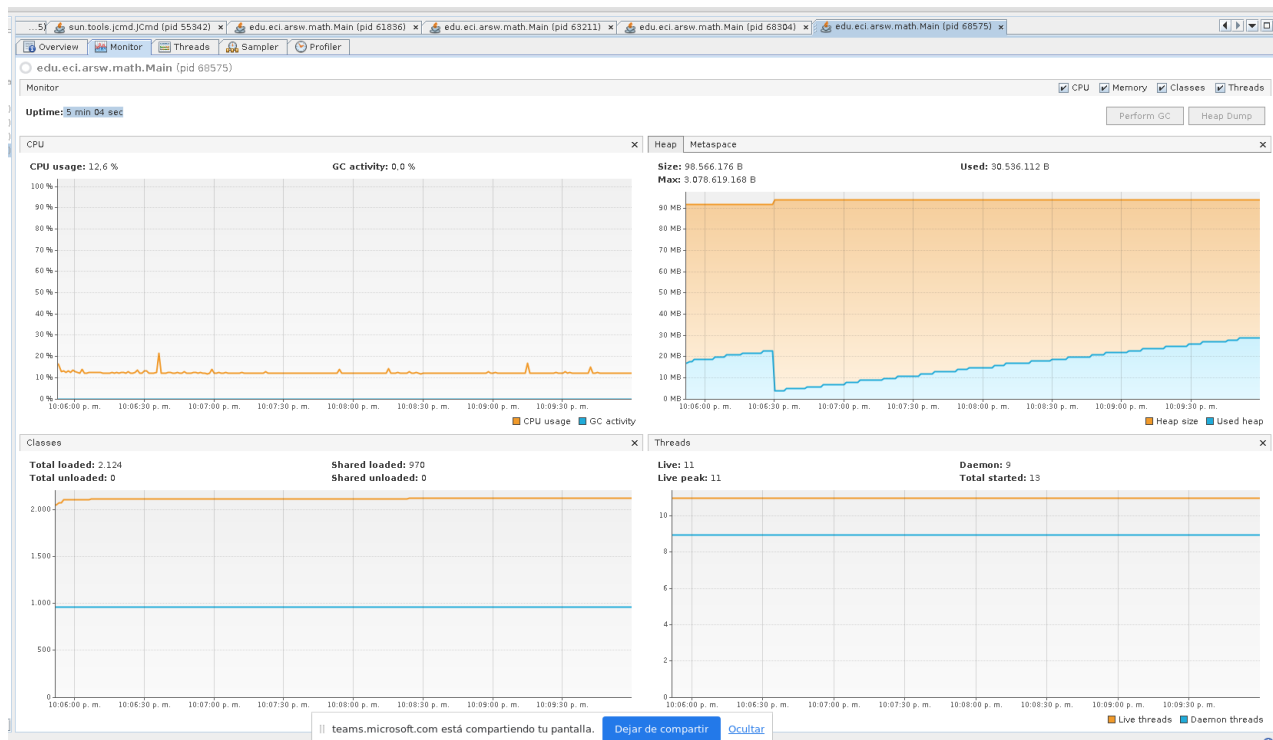
```
Archivo Acciones Editar Vista Ayuda
edhawk@edhawk: ~/Documentos/Proyectos/arsw/lab1arsw
[INFO] Total time: 11.159 s
[INFO] Finished at: 2023-01-31T22:39:29-05:00
[INFO] -----
edhawk@edhawk:~/Documentos/Proyectos/arsw/lab1arsw$ mvn package
[INFO] Scanning for projects...
[INFO] -----< edu.ecl.arsw:PiDigits >-----
[INFO] Building PiDigits 1.0-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ PiDigits ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /home/edhawk/Documentos/Proyectos/arsw/lab1arsw/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ PiDigits ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ PiDigits ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /home/edhawk/Documentos/Proyectos/arsw/lab1arsw/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ PiDigits ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ PiDigits ---
[INFO] Surefire report directory: /home/edhawk/Documentos/Proyectos/arsw/lab1arsw/target/surefire-reports
-----
T E S T S
-----
Running edu.ecl.arsw.math.PiCalcTest
Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.775 sec
Results :
Tests run: 3, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ PiDigits ---
[INFO] Building jar: /home/edhawk/Documentos/Proyectos/arsw/lab1arsw/target/PiDigits-1.0-SNAPSHOT.jar
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 3.428 s
[INFO] Finished at: 2023-01-31T22:40:07-05:00
[INFO] -----
edhawk@edhawk:~/Documentos/Proyectos/arsw/lab1arsw$
```

Parte III

PARA ESTAS PRUEBAS, DISMINUIMOS EL 1000000 DE NÚMEROS PEDIDOS A 100000 POR TIEMPO.

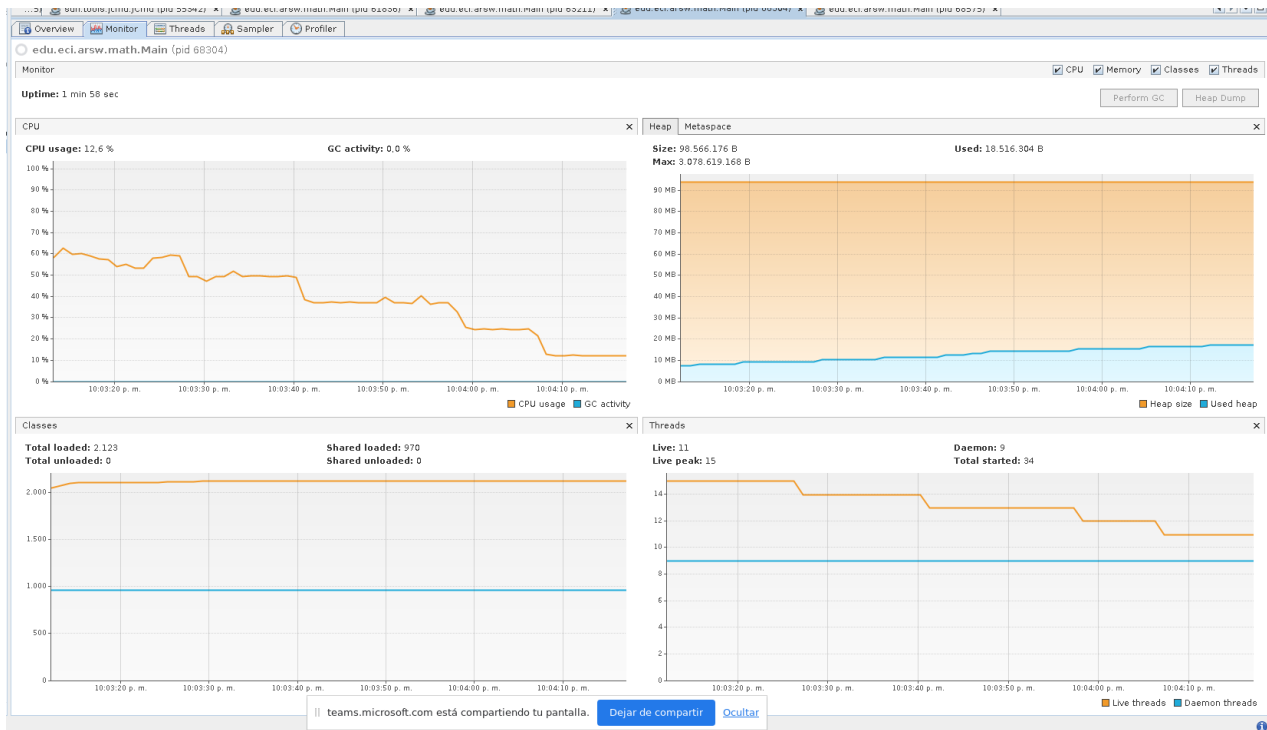
Para un solo hilo

5m 04s



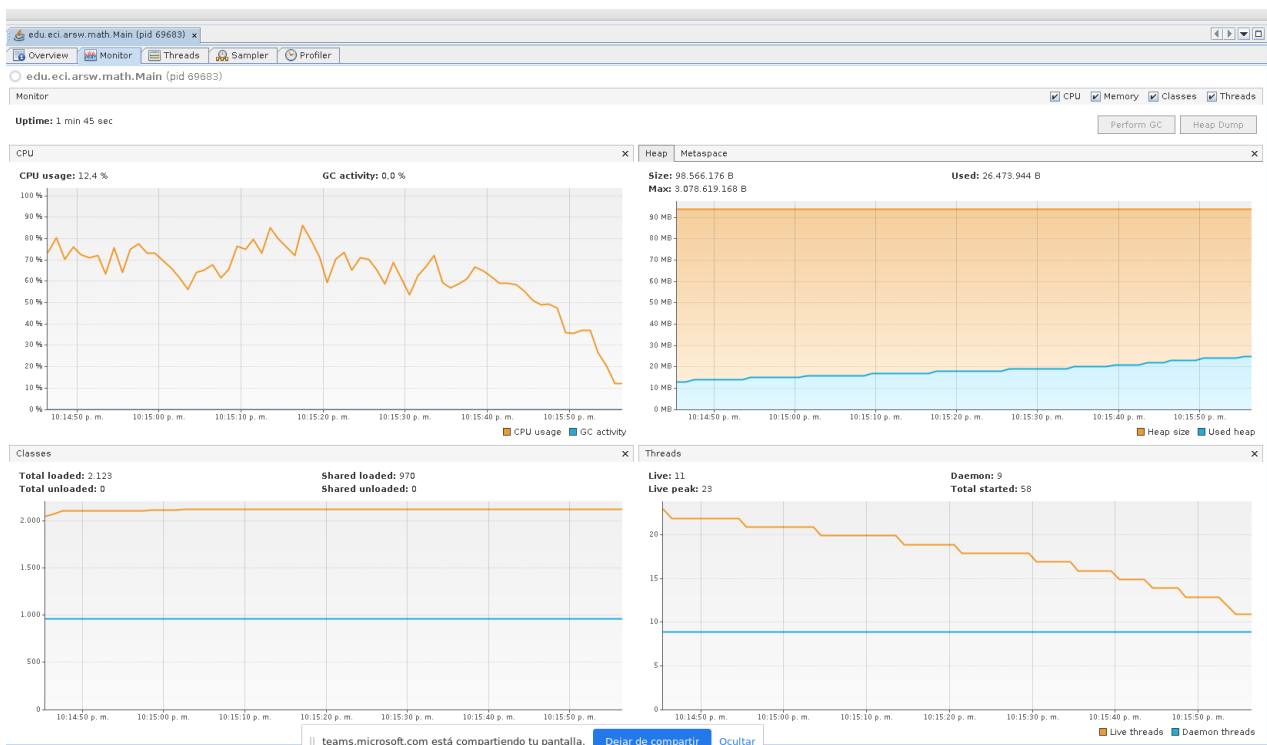
Para el número de núcleos de procesamiento (8)

1m 58s



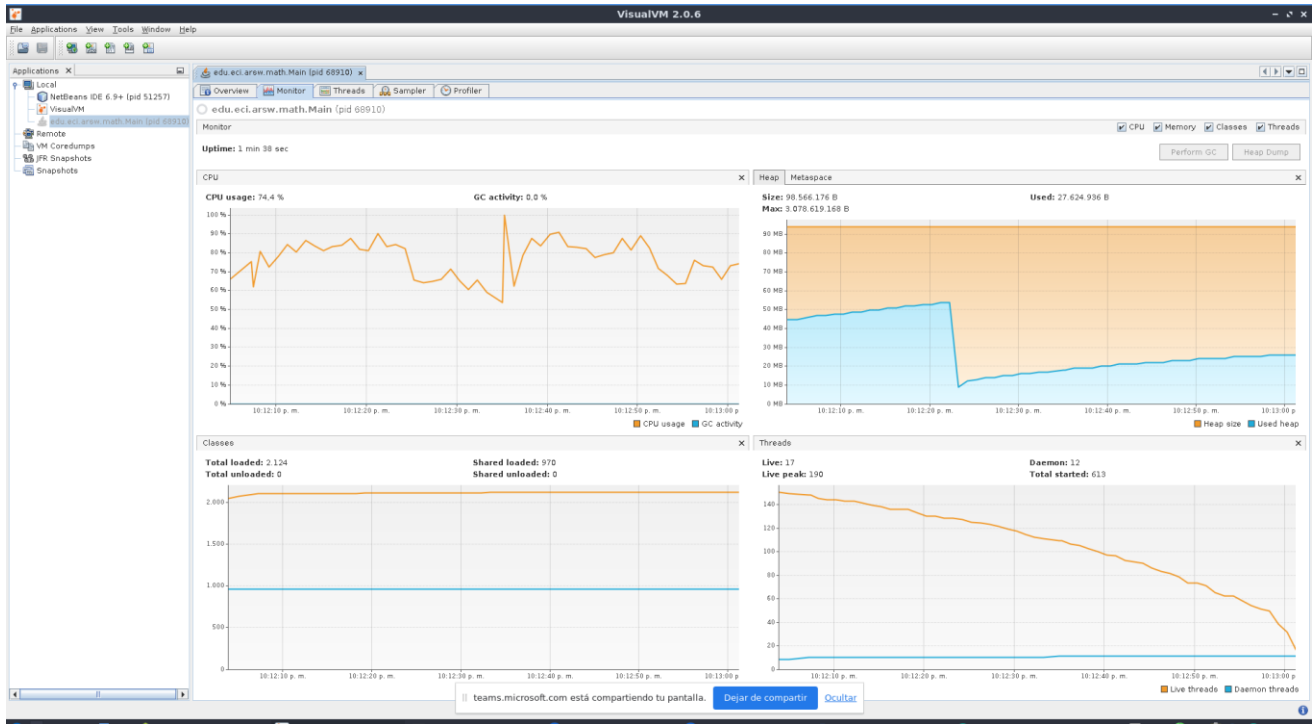
Para el doble de núcleos de procesamiento (16)

1m 45s



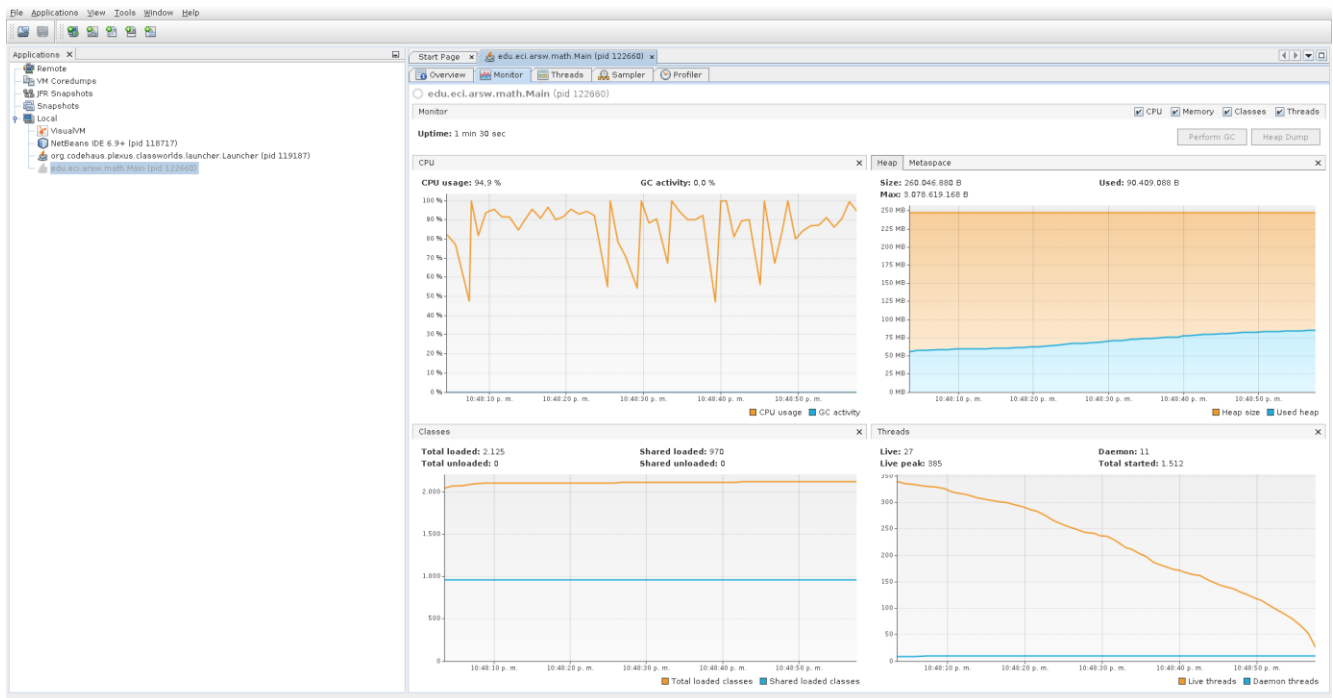
Para 200 hilos

1m 38s



Para 500 hilos

1m 35s



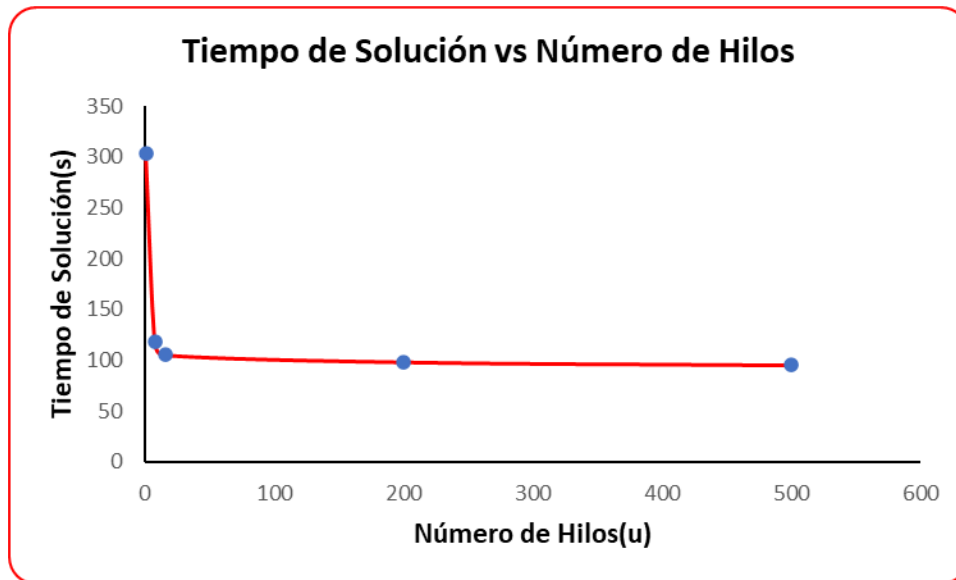
Hipótesis

$$S(n) = \frac{1}{(1 - P) + \frac{P}{n}}$$

1. **RTA:** Según la ecuación, con un número infinito de hilos, tendrías una infinita mejora, sin embargo, esto no tiene en cuenta la capacidad de los computadores. Para el computador también es pesado y es un trabajo extra mantener un número alto de hilos, tiempo que no se tiene en cuenta dentro de la ecuación, y por lo tanto, sólo es válida dentro del rango razonable y no tan lejano del número real de núcleos dentro del computador. En el caso de 500 hilos, es más eficiente que el de 200, sin embargo, la diferencia es prácticamente nula.
2. **RTA:** Es más eficiente usar el doble de núcleos que el mismo número de núcleos, sigue siendo un cambio significativo, sin embargo, no es tan significativo como el cambio entre 1 a el número de núcleos del computador.
3. **RTA:** Si se pudiera crear una máquina por hilo, la ecuación sería prácticamente exacta. En el caso de 500/c depende de la arquitectura completa del computador, puesto que cosas como la RAM pueden actuar como cuello de botella y cambiar el resultado final. La ecuación solo tiene en cuenta el procesador.

A continuación, realizamos la gráfica de tiempo de solución vs. número de hilos.

Número de Hilos(u)	Tiempo de Solución(s)
1	304
8	118
16	105
200	98
500	95



CONCLUSIONES

- Se logró comprender el uso de hilos en Java. La gran mejora en la capacidad de procesamiento y efectividad en el funcionamiento de un programa, junto con la importancia que esto conlleva.
- Se comprendió los rangos en los que es más eficiente adicionar nuevos hilos dentro de una solución concurrente.
- Fue posible distinguir la relevancia de la arquitectura de software dentro de una solución concurrente y paralela, comprendiendo de manera correcta como procesar y ejecutar más de un hilo en ejecución en un mismo tiempo.