

ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO

ARQUITECTURA DE SOFTWARE – ARSW

**LABORATORIO 1**

INTRODUCCIÓN AL PARALELISMO - HILOS

**PROFESOR**

JAVIER IVÁN TOQUICA BARRERA

**ESTUDIANTES**

NICOLÁS ARIZA BARBOSA

MATEO OLAYA GARZÓN

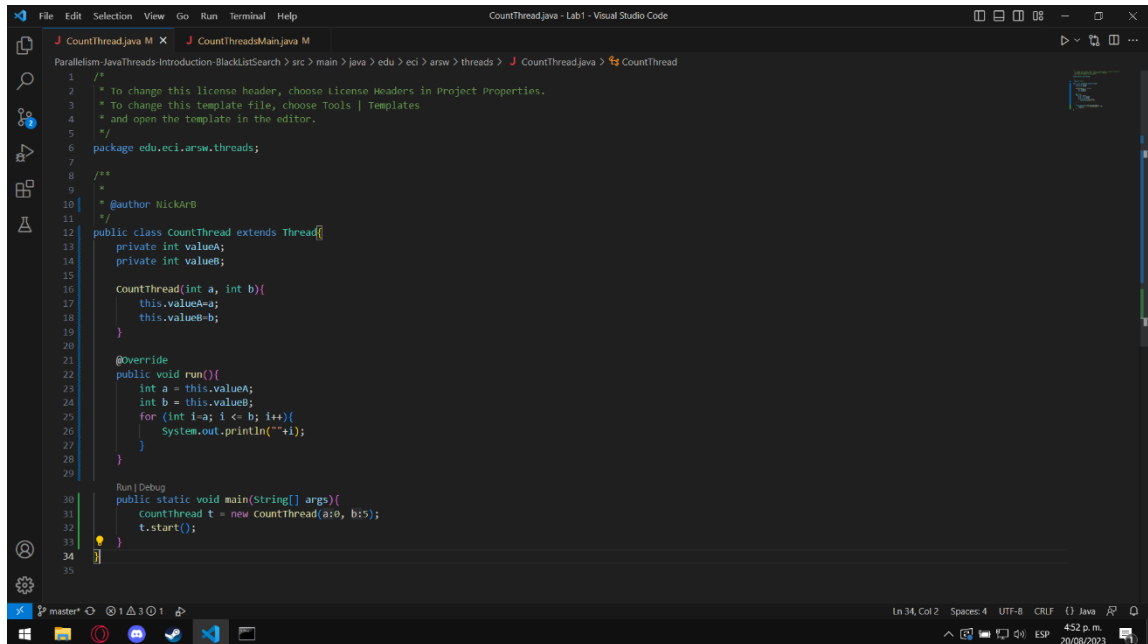
SANTIAGO ANDRÉS ROCHA CRISTANCHO

DAVID EDUARDO VALENCIA CARDONA

BOGOTÁ D.C, AGOSTO 2023

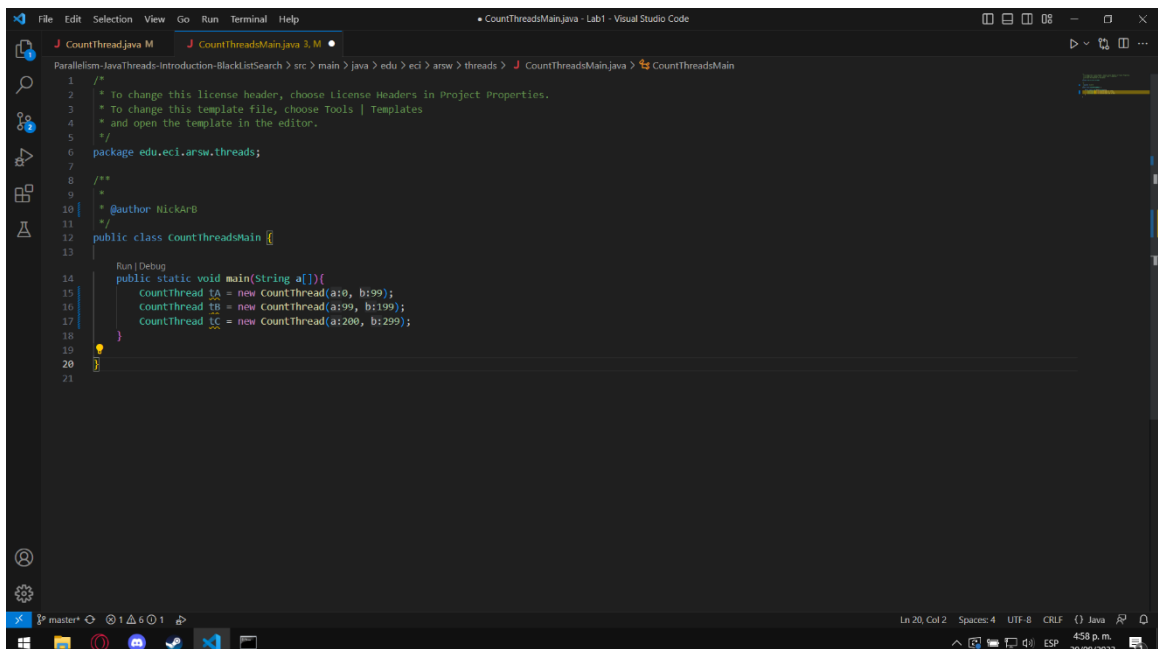
## Primera parte

1. 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.



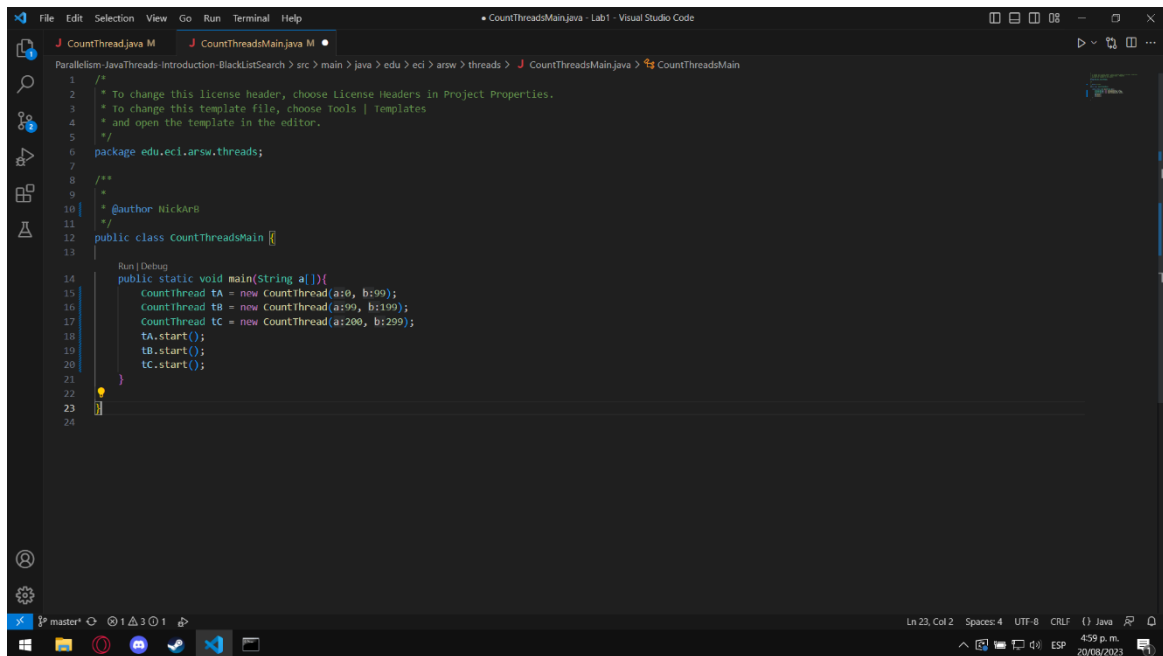
```
1  /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6  package edu.eci.arsw.threads;
7
8  /**
9   *
10   * @author NickArB
11   */
12  public class CountThread extends Thread {
13      private int valueA;
14      private int valueB;
15
16      CountThread(int a, int b) {
17          this.valueA = a;
18          this.valueB = b;
19      }
20
21      @Override
22      public void run() {
23          int a = this.valueA;
24          int b = this.valueB;
25          for (int i = a; i <= b; i++) {
26              System.out.println(i);
27          }
28      }
29
30      Run | Debug
31      public static void main(String[] args) {
32          CountThread t = new CountThread(a10, b15);
33          t.start();
34      }
35  }
```

2. 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].



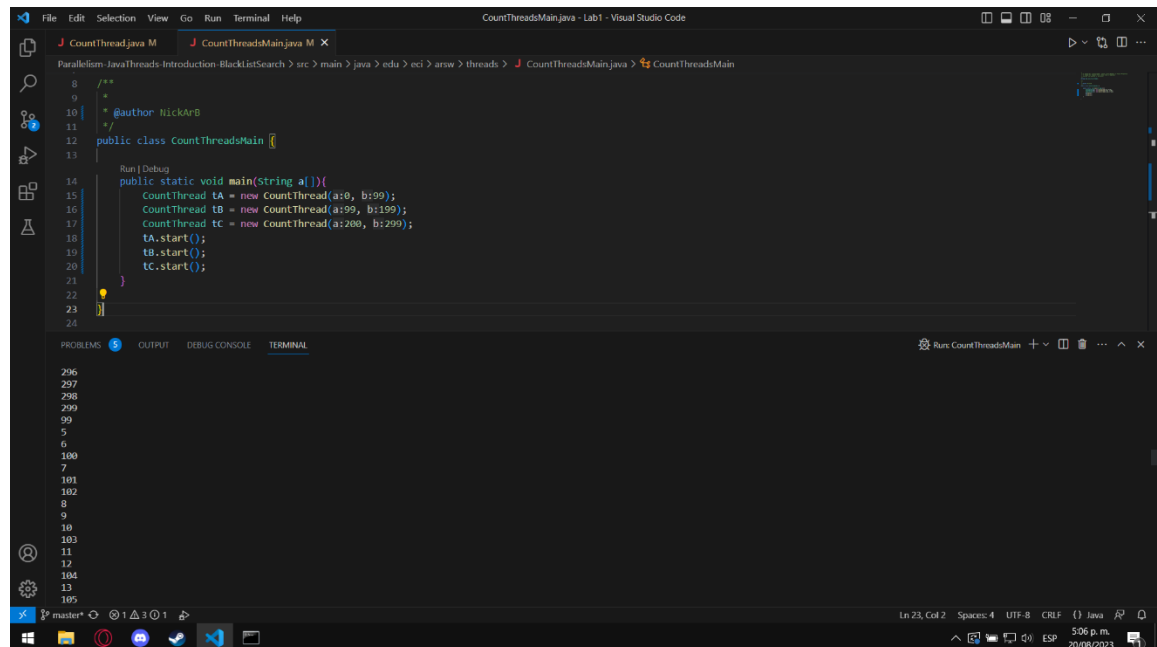
```
1  /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6  package edu.eci.arsw.threads;
7
8  /**
9   *
10   * @author NickArB
11   */
12  public class CountThreadsMain {
13
14      Run | Debug
15      public static void main(String a[]) {
16          CountThread tA = new CountThread(a10, b99);
17          CountThread tB = new CountThread(a99, b199);
18          CountThread tC = new CountThread(a200, b299);
19      }
20  }
```

- 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   *
11   * @author NickAr8
12   */
13  public class CountThreadsMain {
14
15      Run | Debug
16      public static void main(String a[]){
17          CountThread ta = new CountThread(a[0], b[99]);
18          CountThread tb = new CountThread(a[99], b[199]);
19          CountThread tc = new CountThread(a[200], b[299]);
20          ta.start();
21          tb.start();
22          tc.start();
23      }
24  }
```

- Ejecute y revise la salida por pantalla.



```
296
297
298
299
99
5
6
100
7
101
102
8
9
10
103
11
12
104
13
105
```

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

Con el metodo start() los threads se ejecutan en paralelo, los números impresos en consola aparecen en desorden. Esta salida cambia si se ejecuta nuevamente.

The screenshot shows the Visual Studio Code editor with a Java file named `CountThreadsMain.java`. The code defines a `CountThreadsMain` class with a `main` method. Inside the `main` method, three `CountThread` objects are created: `TA` (range 0-99), `TB` (range 99-199), and `TC` (range 200-299). Each object is then started by calling `start()`. The `main` method is highlighted with a blue selection bar. The bottom panel shows the `TERMINAL` tab, which displays the output of the program, showing the counts for each thread: 278, 38, 279, 139, 280, 281, 282, 49, 283, 284, 140, 285, 40, 286, 141, 287, 41, 288, 289, 142.

```
8 /**
9  *
10  * @author NickArB
11  */
12 public class CountThreadsMain {
13
14     public static void main(String a[]){
15         CountThread TA = new CountThread(a:0, b:99);
16         CountThread TB = new CountThread(a:99, b:199);
17         CountThread TC = new CountThread(a:200, b:299);
18         TA.start();
19         TB.start();
20         TC.start();
21     }
22 }
23
24
```

PROBLEMS 5 OUTPUT DEBUG CONSOLE TERMINAL

Run: CountThreadsMain

278  
38  
279  
139  
280  
281  
282  
49  
283  
284  
140  
285  
40  
286  
141  
287  
41  
288  
289  
142

Ln 23, Col 2 Spaces: 4 UTF-8 CRLF Java 20/06/2023 5:12 p.m.

Mientras que si se cambia la ejecución mediante el método `run()` los threads ya no correrán en paralelo si no que lo harán de forma secuencial como si fuese un llamado de métodos de una clase ordinaria.

The screenshot shows the Visual Studio Code editor with the same `CountThreadsMain.java` file. In this version, the `main` method calls `run()` instead of `start()` for the three `CountThread` objects. The `main` method is highlighted with a blue selection bar. The bottom panel shows the `TERMINAL` tab, which displays the output of the program, showing the counts for each thread: 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28.

```
8 /**
9  *
10  * @author NickArB
11  */
12 public class CountThreadsMain {
13
14     public static void main(String a[]){
15         CountThread TA = new CountThread(a:0, b:99);
16         CountThread TB = new CountThread(a:99, b:199);
17         CountThread TC = new CountThread(a:200, b:299);
18         TA.run();
19         TB.run();
20         TC.run();
21     }
22 }
23
24
```

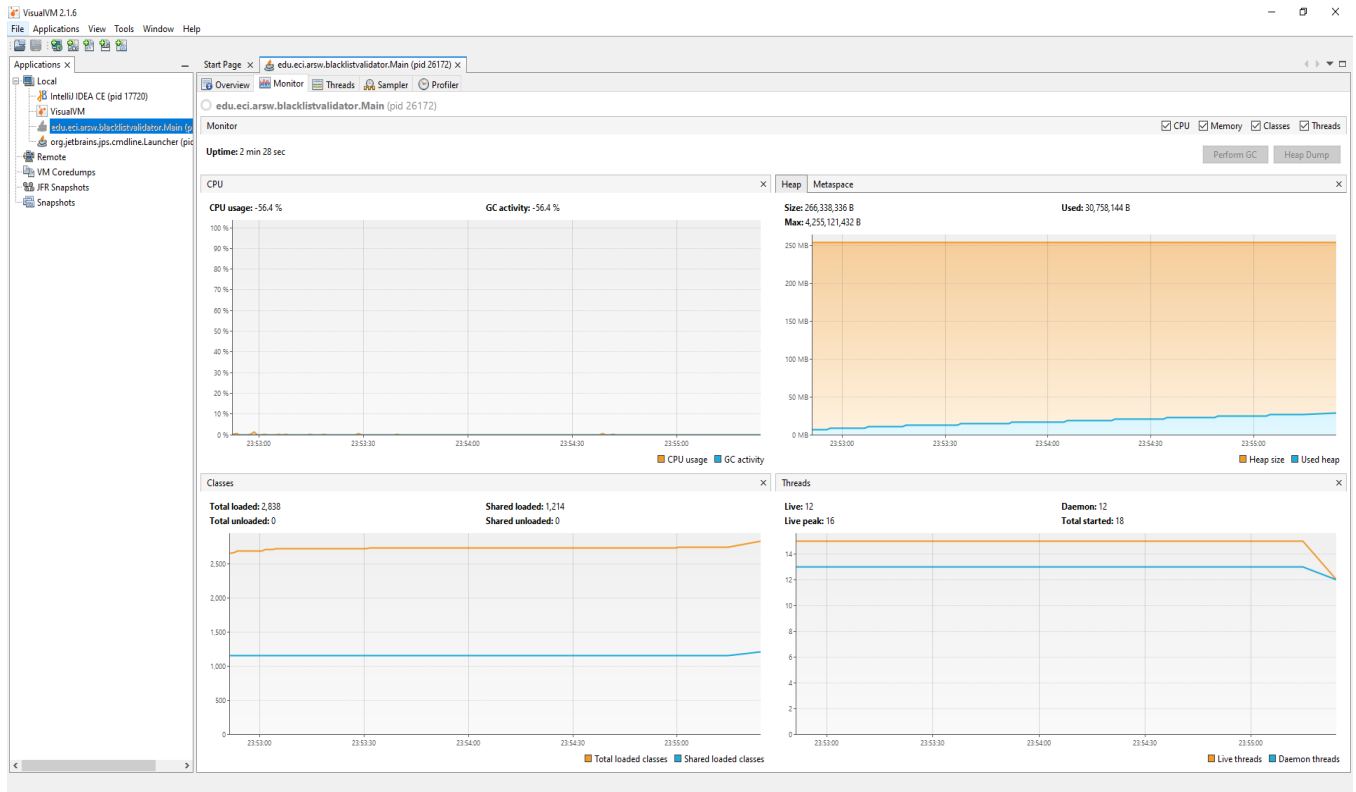
PROBLEMS 5 OUTPUT DEBUG CONSOLE TERMINAL

Run: CountThreadsMain

9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28

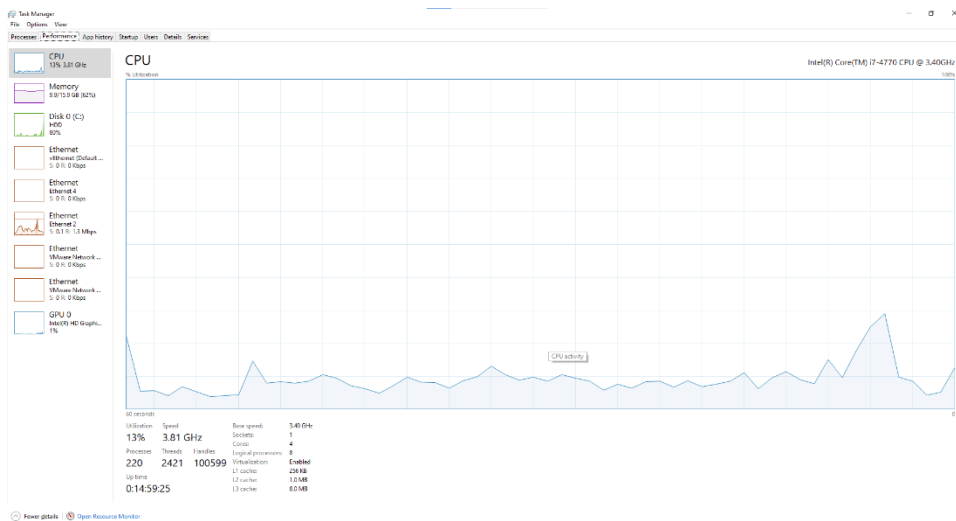
Ln 20, Col 15 Spaces: 4 UTF-8 CRLF Java 20/06/2023 5:15 p.m.

## Tercera Parte: Con un hilo:



Tantos hilos como núcleos de procesamiento:

Usando un computador del laboratorio de sistemas con las siguientes características de procesador:

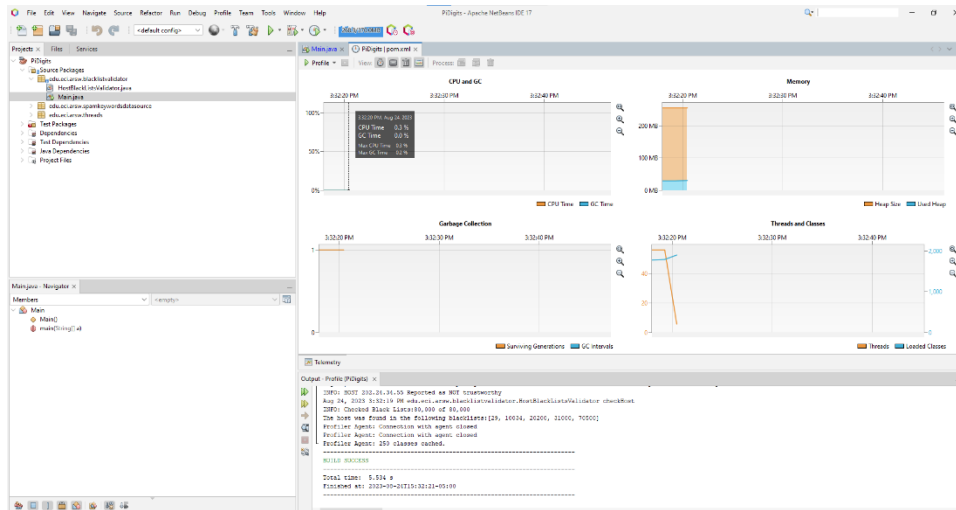


The screenshot displays the VisualVM 2.1.6 interface with the Monitoring tab selected. The application being monitored is edu.csi.smc.blockscheduler.Main (pid 8728). The CPU usage graph shows 0.0% usage and 0.0% GC activity. The Memory usage graph shows a steady increase in memory usage, reaching approximately 100 MB. The Threads graph shows a steady increase in the number of live threads, reaching approximately 20 threads.

[illegible]

Esto se tuvo que usar usando el perfilador de NETBEANS:

50 Hilos:



100 Hilos

