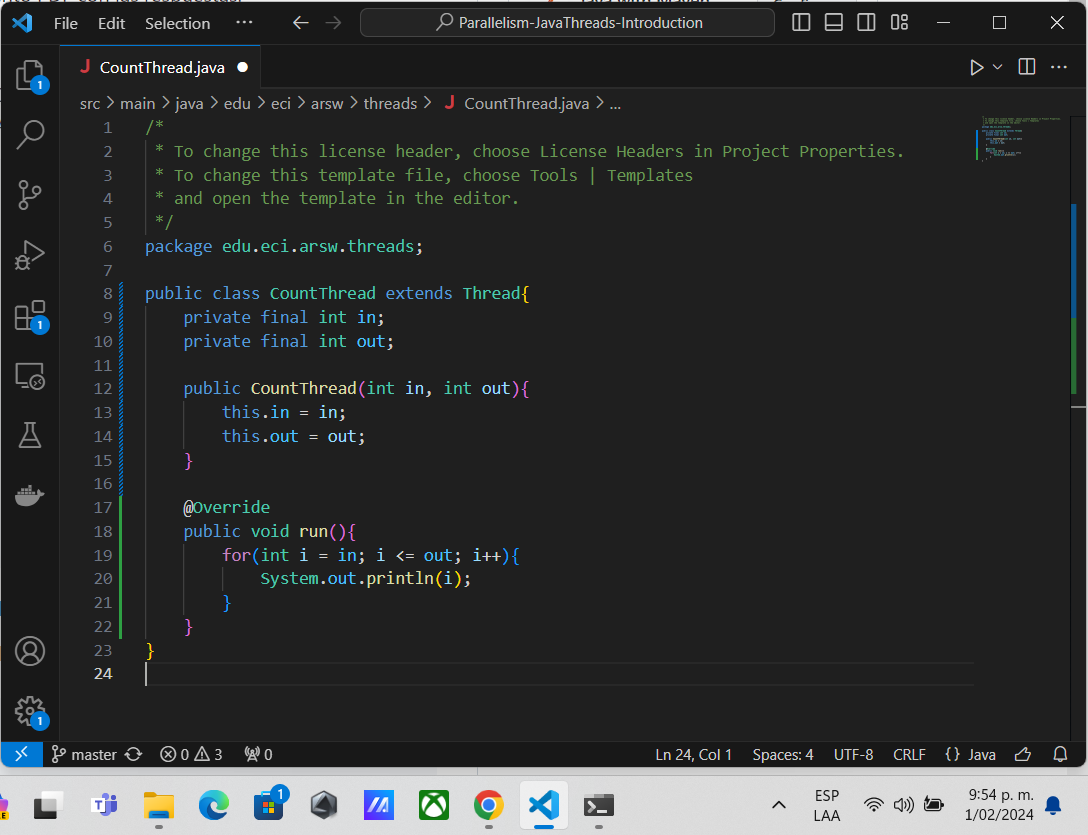
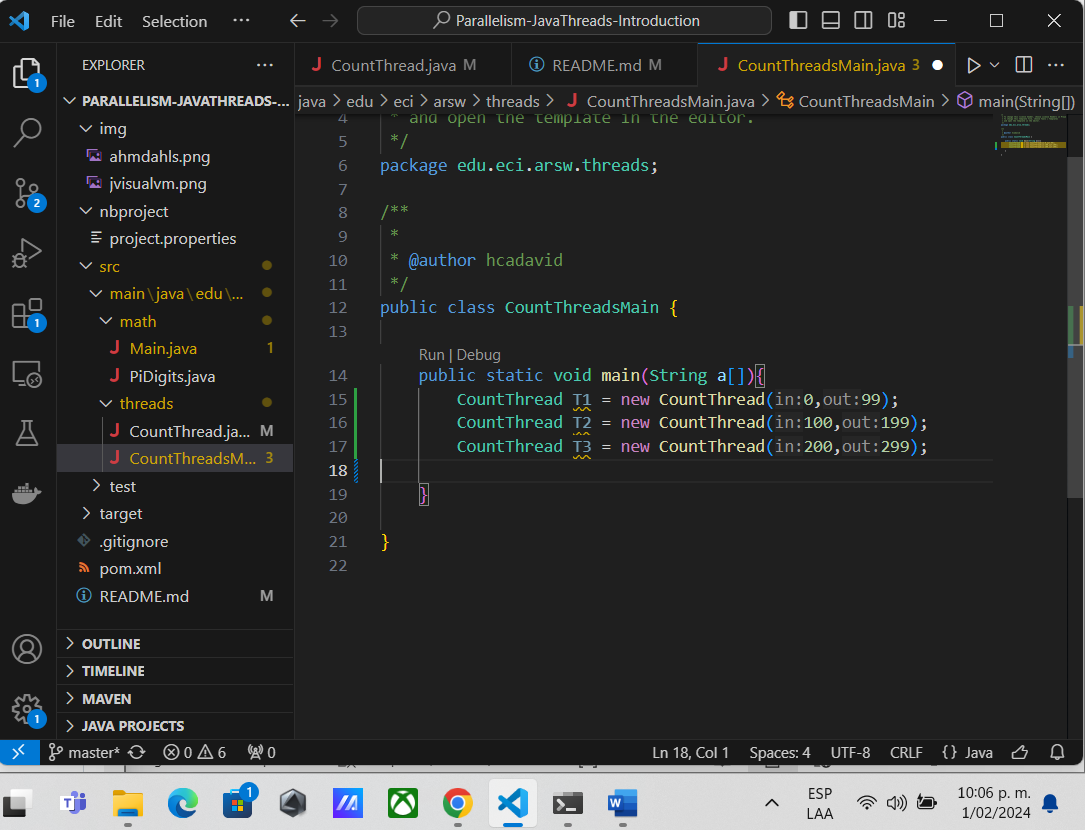
Parte 1 Hilos Java:

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.

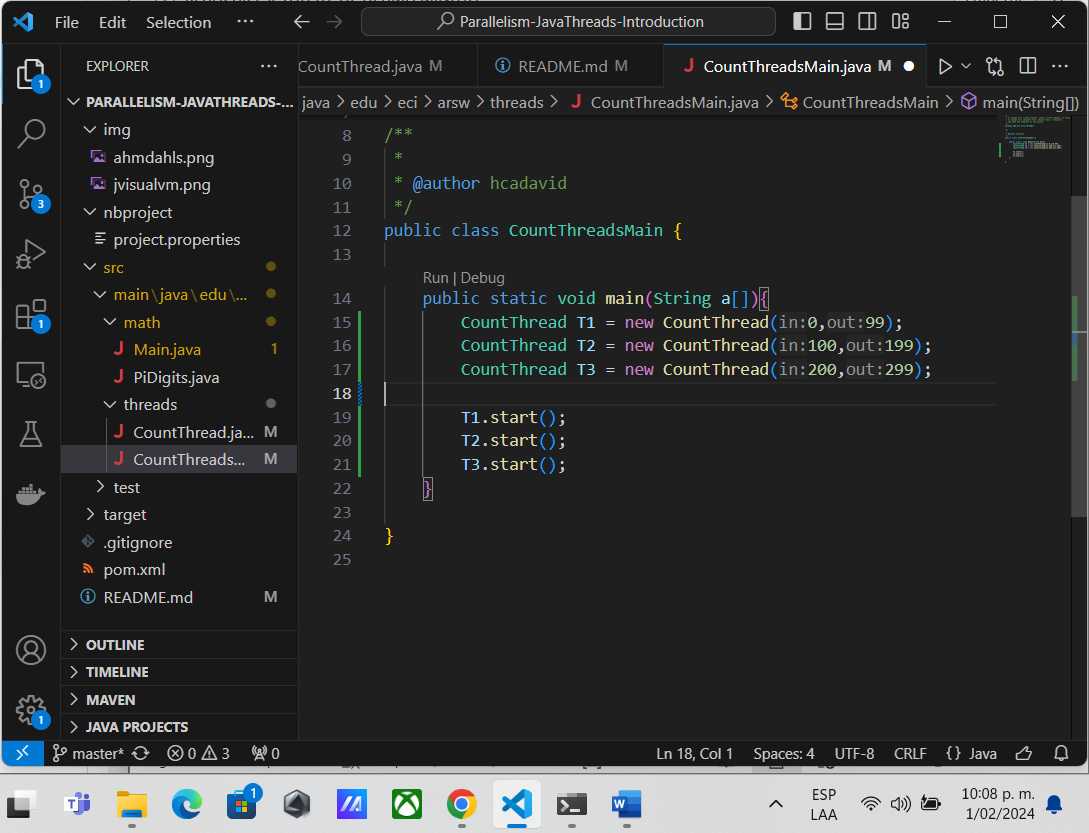


1. Complete el método **Main** de la clase CountMainThreads para que:

* Cree 3 hilos de tipo CountThread, asignándole al primero el intervalo [0..99], al segundo [99..199] y al tercero [200..299].

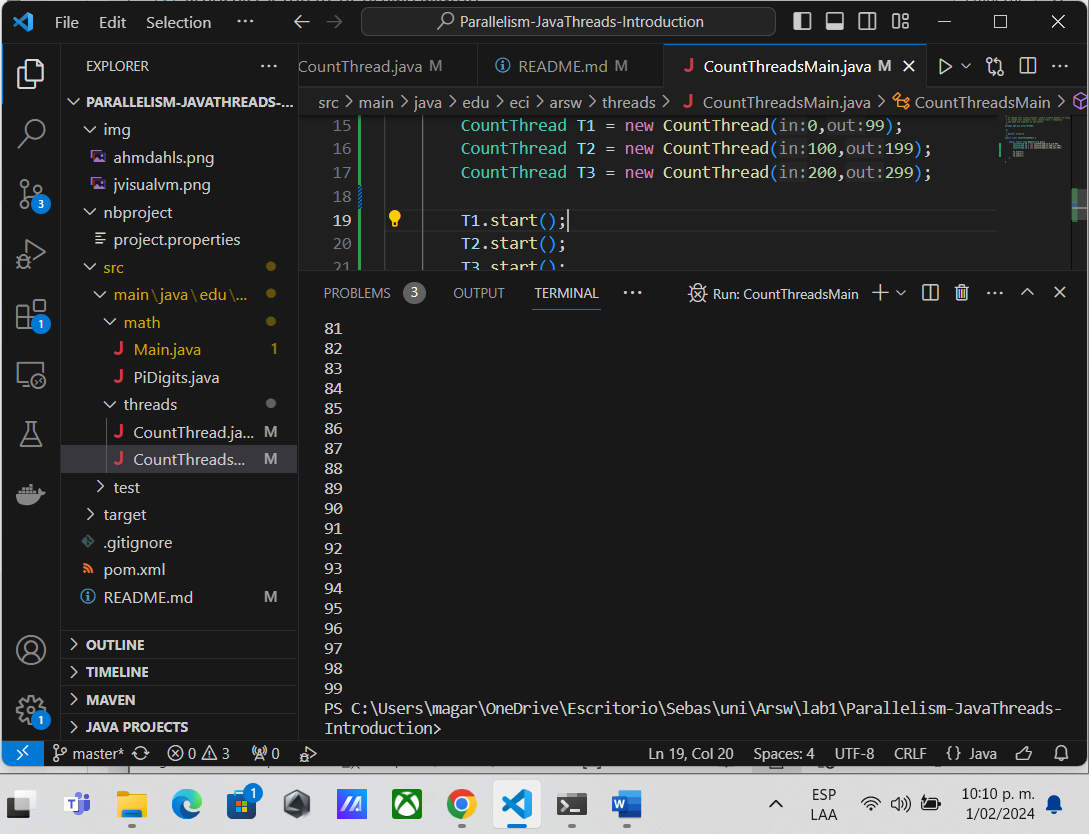


* Inicie los tres hilos con “start()”:



* Ejecute y revise la salida por pantalla.

La respuesta del código es iniciar los 3 hilos consecutivamente e imprimir los números de cada rango asignado a cada hilo

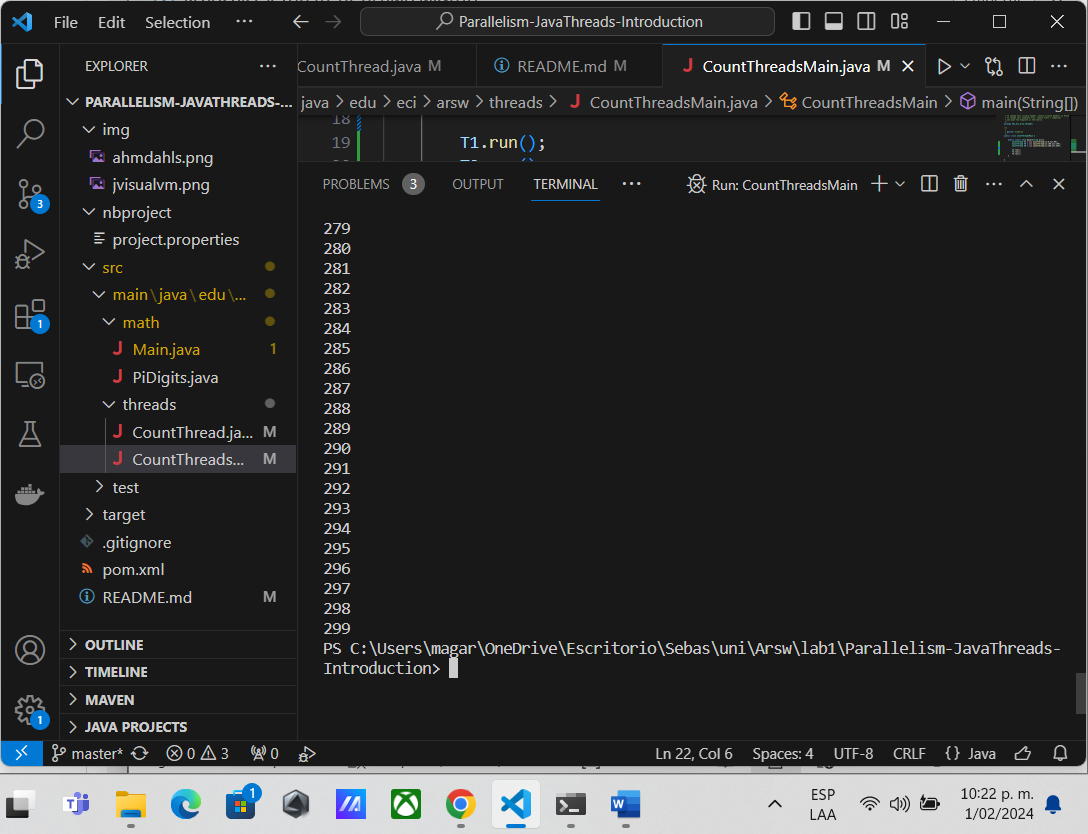


* Cambie el inicio con “start()” por “run()”. ¿Cómo cambia la salida?, por qué?

La salida cambia debido a que con el método run los hilos se ejecutan únicamente hasta que termine el anterior, por lo que siguen un orden específico

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

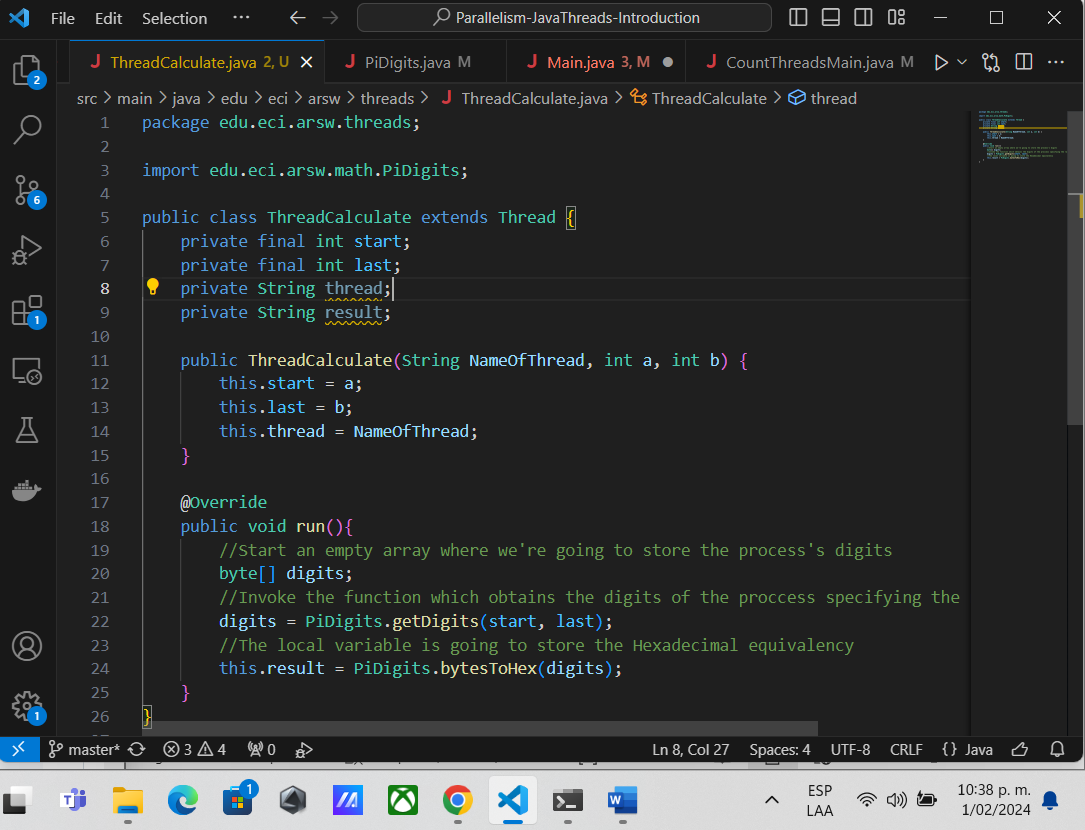


**Parte 2 Hilos Java**

1. Cree una clase de tipo Thread que represente el ciclo de vida de un hilo que calcule una parte de los dígitos requeridos.

**Solución**

El constructor de la clase ThreadCalculate recibe tres parámetros, los cuales son el nombre o número de thread **(NameOfThread)** que utilizamos para poder identificar cada Hilo y su resultado, además de el valor entero **(a)** que indica el punto de partida del hilo y el valor entero **(b)** que indica el final o número de valores que el hilo debe ocupar.



1. Haga que la función PiDigits.getDigits() reciba como parámetro adicional un valor N, correspondiente al número de hilos entre los que se va a paralelizar la solución. Haga que dicha función espere hasta que los N hilos terminen de resolver el problema para combinar las respuestas y entonces retornar el resultado. Para esto, revise el método **join** del API de concurrencia de java.

**Solución:**

Para realizar este ejercicio utilizamos la clase **PiDigits** como orquestador, enviándole al constructor el parámetro N que indica la cantidad de hilos que debe utilizar el programa y así creamos un método aparte **(orchestor())** el cual se encarga de crear todos los hilos y arrancarlos así obteniendo y retornando el resultado esperado.

El método bytesToHex() Se movió del main a la clase PiDigits

Constructor:

