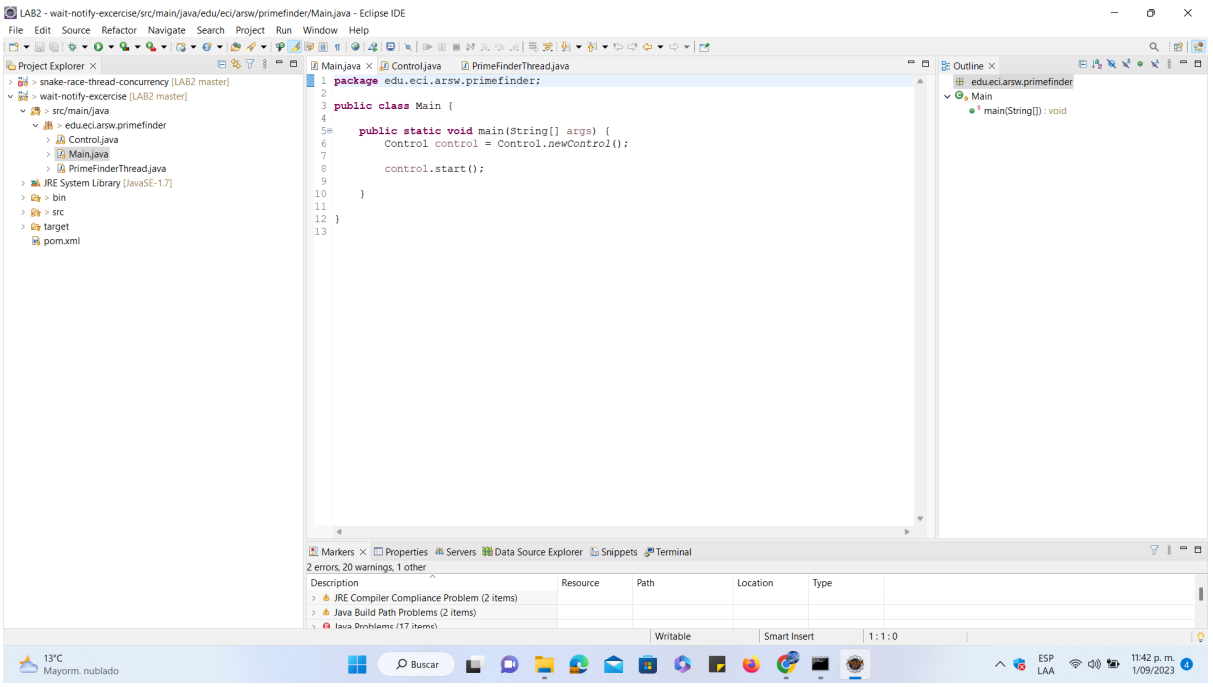
Laboratorio 2 ARSW

Johan Sebastian Garcia Martinez

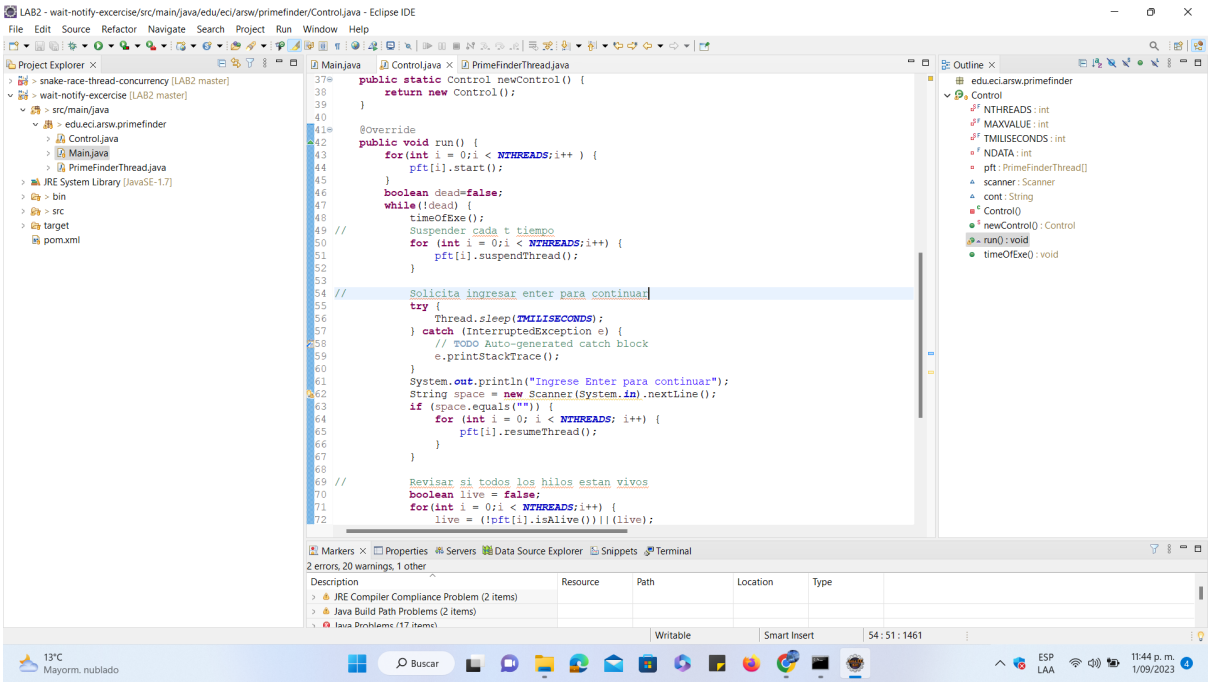
Introducción

Bitácora

Primero clonamos el proyecto de “prime finder” para poder dar solución



una vez en nuestro editor podemos identificar las secciones que se encuentran incompletas en el “orquestador” de hilos que vendría siendo la clase “Control”.



Teniendo esto en cuenta y una vez inicializados todos los hilos “PrimeFinderThreads” desde el vector “pft” que los contiene dada la cantidad “N” estipulada de hilos, podemos realizar con la implementación, donde la condición principal es que se deben entregar al usuario los primos encontrados por el programa cada “t” milisegundos, para esto debemos implementar un método que nos indique el tiempo de ejecución de cada hilo antes de “suspenderlo”.

el método es llamado “timeToExe()”

```
// Método para dejar con "vida" T milisegundos los hilos
public void timeOfExe() {
    boolean flag = false;
    long start = System.currentTimeMillis();
    long end = 0;
    while (!flag) {
        end = System.currentTimeMillis() - start;
        if (end >= TMILISECONDS) {
            flag = true;
        }
    }

}
```

este método lo que nos permite hacer es una “pausa” dentro del hilo principal u “orquestador” mientras los hilos realizan la búsqueda de los números primos que se encuentran en los rangos que se les estipularon

todo este procedimiento se hace si y sólo si los hilos aún no se encuentran en estado de “finalizado” para eso ponemos un bucle que identifique cuando estos se encuentran en estado de ejecución

Ahora desde este mismo ciclo, despues de transcurridos los “T” milisegundos, por medio de un ciclo for, se hace llamado a todos los hilos del vector ptf[] por medio del método “suspendThreads” para que con el comando “wait()” puedan quedar suspendidos o “dormidos” hasta nuevo aviso

```
Suspender cada t tiempo
for (int i = 0;i < NTHREADS;i++) {
    pft[i].suspendThread();
}
```

lo que realiza este llamado es cambiar la condición de carrera o del bucle dentro del bloque sincronizado para que pueda ingresar y así suspender por medio del comando wait() a todos los hilos que se encuentran en la cola

```
synchronized (this){
    while(suspend) {
        try {
            this.wait();
        }catch(InterruptedException e){
            throw new RuntimeException(e);
        }
    }
}
```

una vez hecho esto se implementa la entrada del usuario donde se le solicita presionar la tecla “enter” para poder continuar con la busqueda de los primos y es así como se reanuda la operación de todos los hilos y continuan imprimiendo en pantalla con normalidad

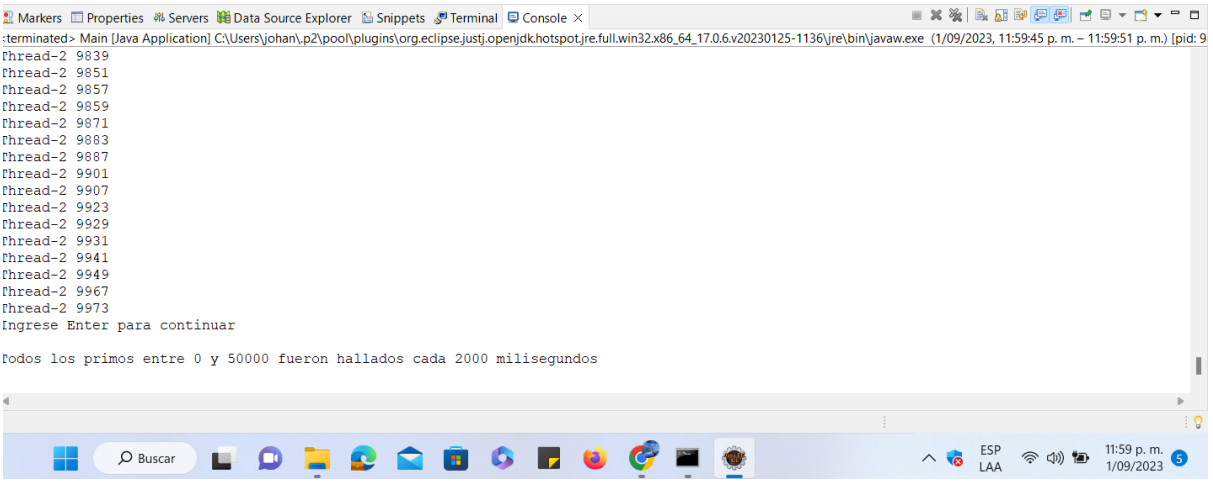
```
System.out.println("Ingrese Enter para continuar");
String space = new Scanner(System.in).nextLine();
if (space.equals("")) {
    for (int i = 0; i < NTHREADS; i++) {
        pft[i].resumeThread();
    }
}
```

Este método lo unico que hace es cambiar el estado de la condición del bloque sincronizado y llama al método notify() para reanudar la operación de los hilos que se encuentren en la cola, debido a que no existe un único hilo que llame a todos los demas a “despertar”

Luego de esto se verifica si los hilos aún no se encuentran en estado “finalizado” y de ser así se cumple con la condición del bucle y finaliza la operación del orquestador

```
Revisar si todos los hilos estan vivos
boolean live = false;
for(int i = 0;i < NTHREADS;i++) {
    live = (!pft[i].isAlive())||(live);
}
dead = live;
```

Donde al final mostrara en pantalla el hilo en ejecución y la lista de los primos encontrados, y un mensaje que indica la finalización del proceso



Conclusiones