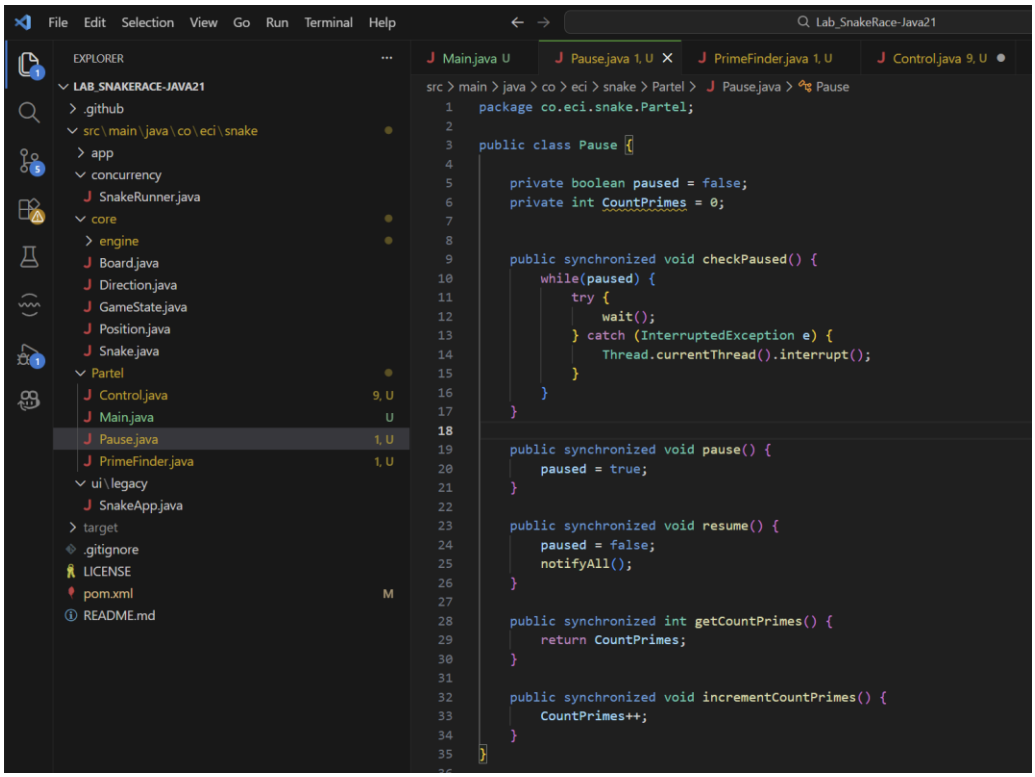
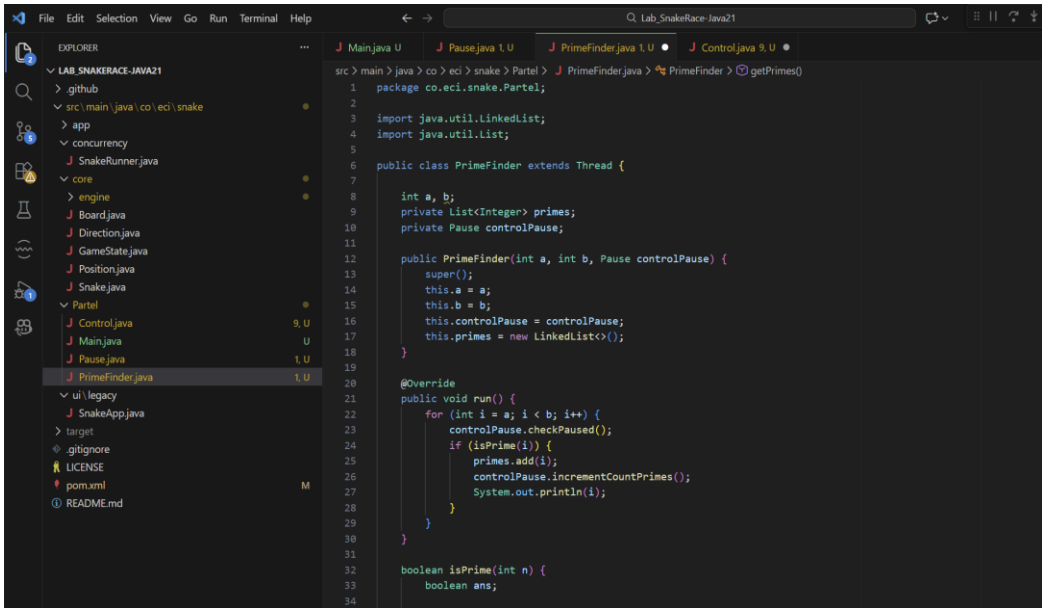
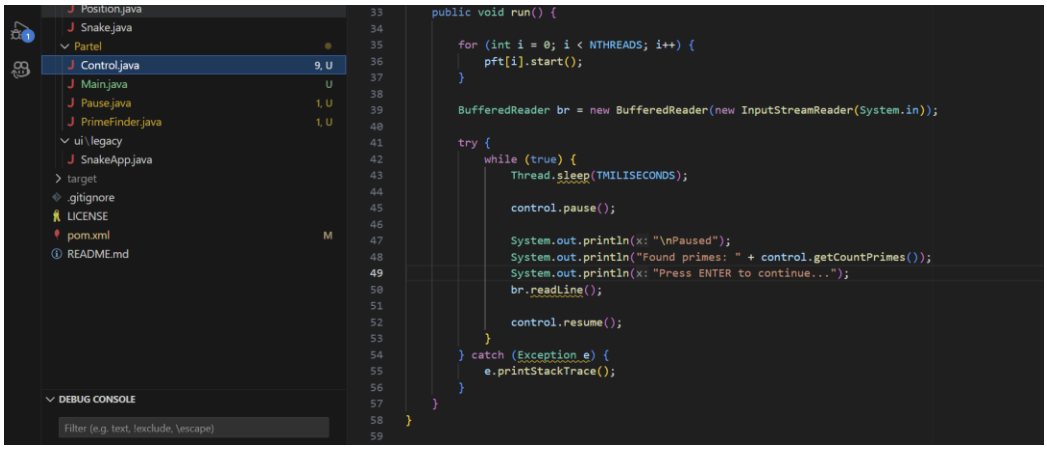
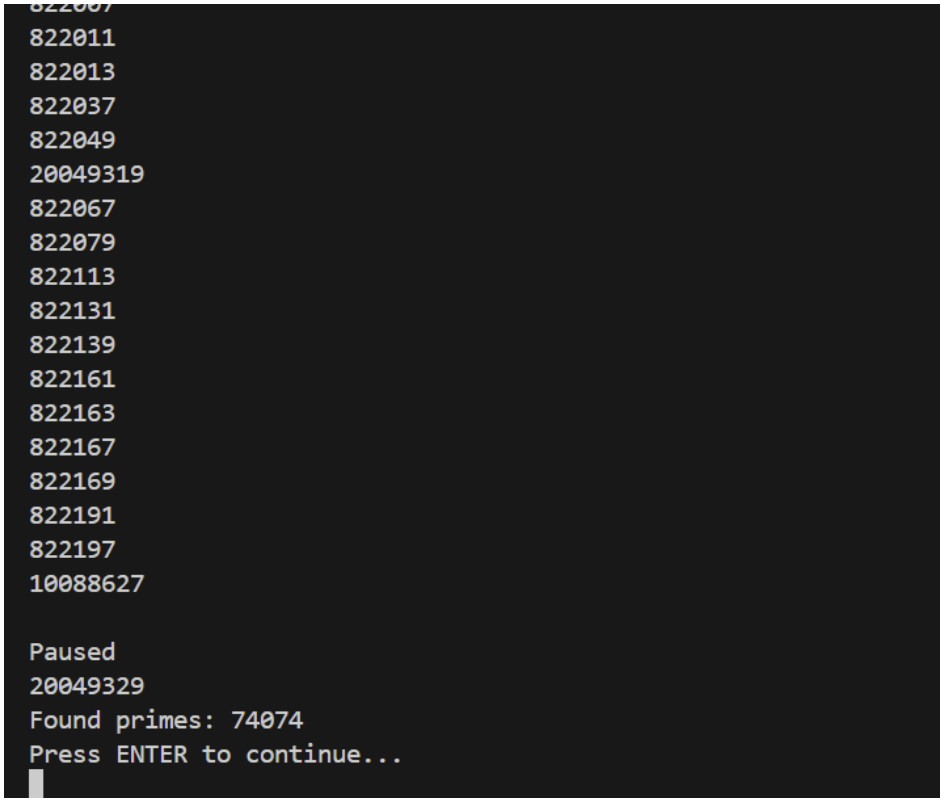
Parte I

Punto 2 y 3






Position.java
Snake.java
Partel
Control.java 9, U
Main.java U
Pause.java 1, U
PrimeFinder.java 1, U
ui \ legacy
SnakeApp.java
target
.gitignore
LICENSE
pom.xml M
README.md
DEBUG CONSOLE
Filter (e.g. text, !exclude, \escape)
public void run() {
for (int i = 0; i < NTHREADS; i++) {
pft[i].start();
}
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
try {
while (true) {
Thread.sleep(TMILISECONDS);
control.pause();
System.out.println(x: "\nPaused");
System.out.println("Found primes: " + control.getCountPrimes());
System.out.println(x: "Press ENTER to continue...");
br.readLine();
control.resume();
}
} catch (Exception e) {
e.printStackTrace();
}
}
}


822011
822013
822037
822049
20049319
822067
822079
822113
822131
822139
822161
822163
822167
822169
822191
822197
10088627
Paused
20049329
Found primes: 74074
Press ENTER to continue...

Punto 4

Para coordinar la ejecución de los hilos se utilizó un único objeto compartido que actúa como monitor. Este objeto es usado tanto por los hilos trabajadores como por el hilo controlador, y su candado (lock) se obtiene mediante métodos sincronizados (synchronized).

La pausa de los hilos se controla con una variable booleana que indica si el sistema está en estado de pausa. Cuando esta variable está activa, los hilos trabajadores se detienen de forma cooperativa llamando a wait(), lo que les permite quedar bloqueados sin consumir CPU. Cuando el sistema se reanuda, el hilo controlador cambia el estado de la condición y despierta a todos los hilos con notifyAll().

Para evitar problemas de *lost wakeups*, la llamada a wait() se realiza dentro de un ciclo while que vuelve a verificar la condición cada vez que el hilo se despierta. De esta forma se garantiza que ningún hilo continúe ejecutándose si la pausa aún está activa, incluso en caso de despertares, inesperados.