Laboratorio 2

Programación concurrente, condiciones de carrera, esquemas de sincronización, colecciones sincronizadas y concurrentes.

url repositorio:

https://github.com/ARSW2023-2/ARSW_Lab02.git

Integrantes:

Luisa Fernanda Bermúdez Girón

Karol Daniela Ladino Ladino

Squad:

Inside Out

Profesor:

Javier Iván Toquica Barrera

Curso:

ARSW – 1

Fecha De Entrega:

01-09-2023

I. Introducción

En este laboratorio, exploraremos conceptos fundamentales de programación concurrente, condiciones de carrera y esquemas de sincronización. Estudiaremos cómo manejar múltiples hilos de ejecución de manera eficiente y segura en entornos paralelos.

A través de ejercicios prácticos, nos adentraremos en el uso de herramientas cruciales como wait y notify, perfeccionaremos nuestras habilidades para identificar y resolver condiciones de carrera, y aplicaremos estos conocimientos al enriquecimiento y optimización de una aplicación de juego interactiva.

II. Desarrollo del laboratorio

Parte I

Control de hilos con wait/notify.

1. Descargue el proyecto *PrimeFinder*. Este es un programa que calcula números primos entre 0 y M (Control.MAXVALUE), concurrentemente, distribuyendo la búsqueda de los mismos entre n (Control.NTHREADS) hilos independientes.

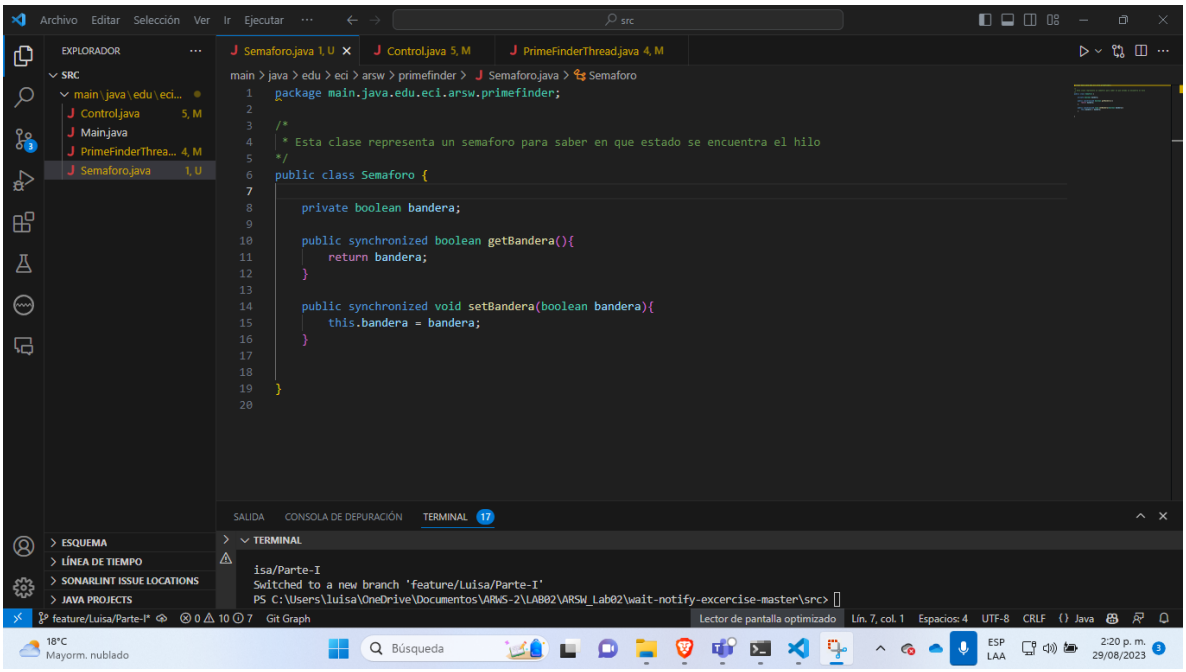
2. Se necesita modificar la aplicación de manera que cada t milisegundos de ejecución de los threads, se detengan todos los hilos y se muestre el número de primos encontrados hasta el momento. Luego, se debe esperar a que el usuario presione ENTER para reanudar la ejecución de los mismos. Utilice los mecanismos de sincronización provistos por el lenguaje (wait y notify, notifyAll).

Tenga en cuenta:

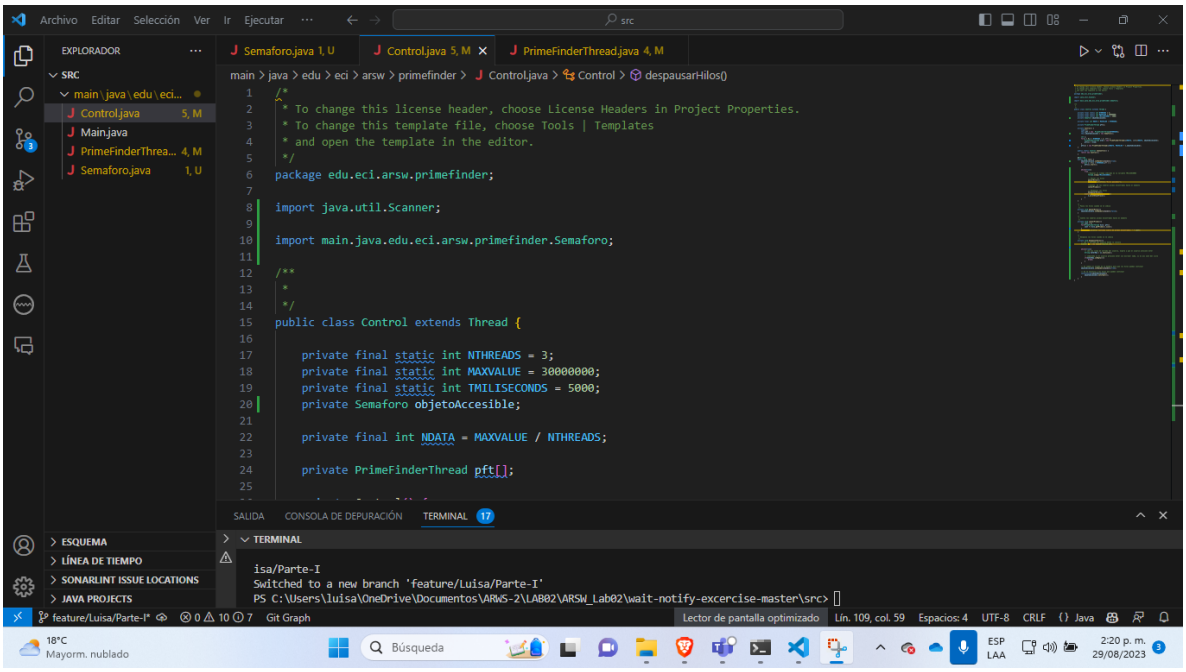
- La construcción synchronized se utiliza para obtener acceso exclusivo a un objeto.
- La instrucción A.wait() ejecutada en un hilo B pone en modo suspendido al hilo B (independientemente de qué objeto 'A' sea usado como 'lock'). Para reanudarlo, otro hilo activo puede reanudar a B haciendo 'notify()' al objeto usado como 'lock' (es decir, A).
- La instrucción notify(), despierta el primer hilo que hizo wait() sobre el objeto.
- La instrucción notifyAll(), despierta todos los hilos que estan esperando por el objeto (hicieron wait()sobre el objeto).

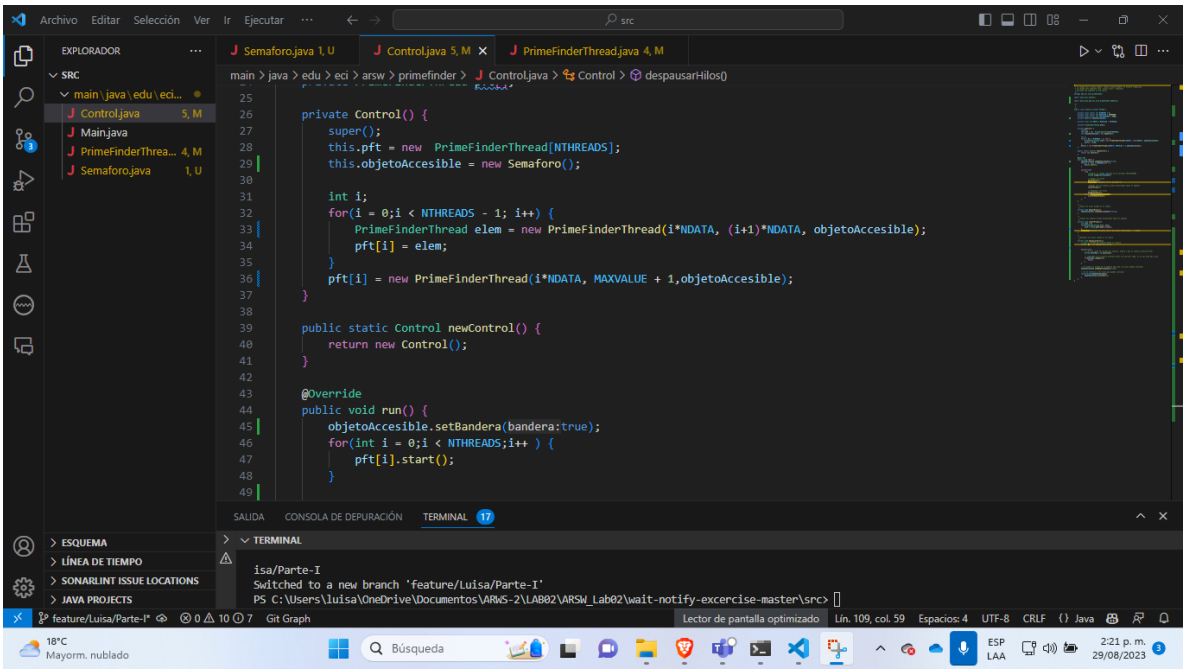
Nota: El último commit de esta parte fue: Pantallazos parte I. El tag es v1.0

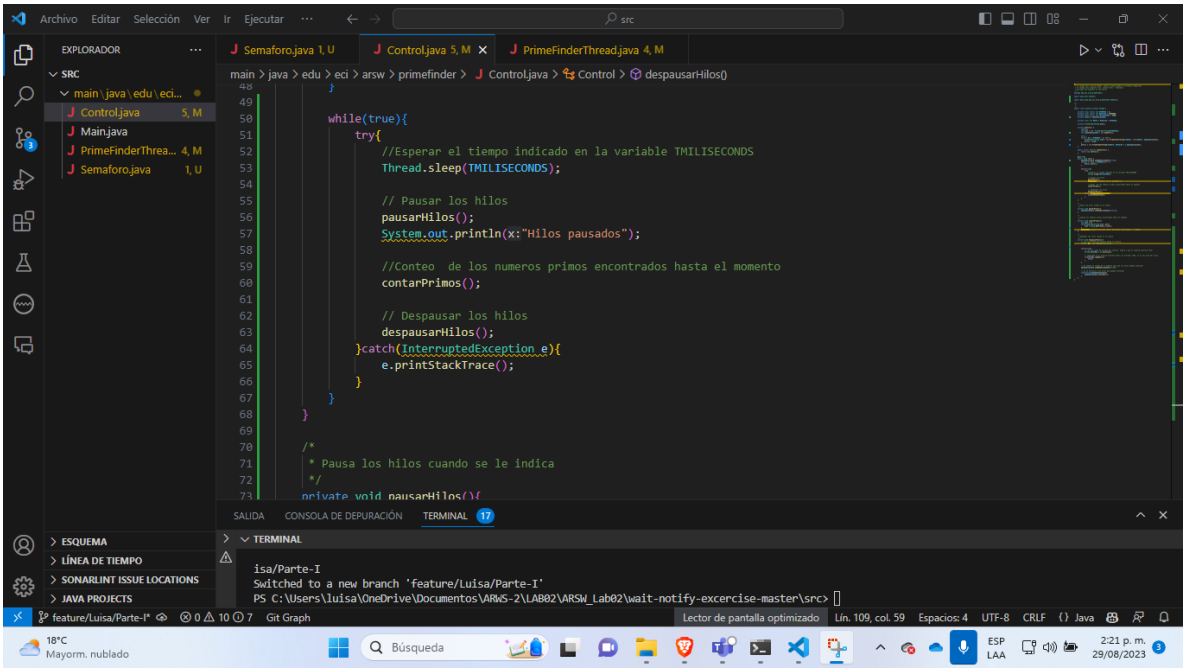
- Creamos la clase Semaforo la cual nos permitirá conocer en que estado se encuentra el hilo

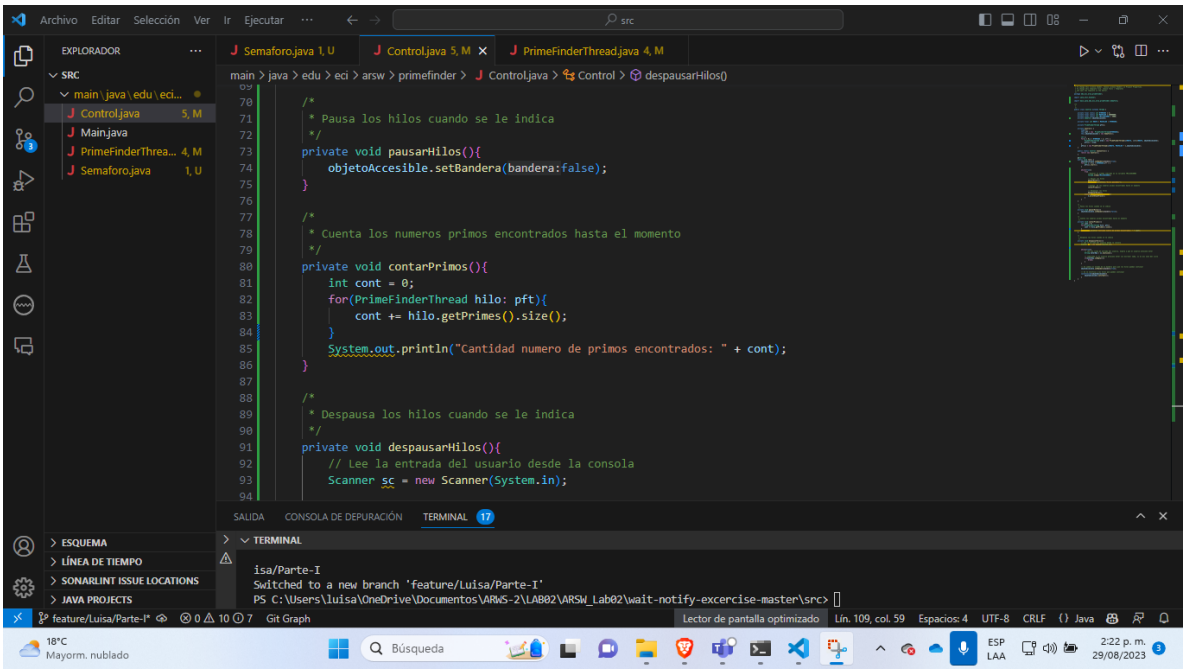


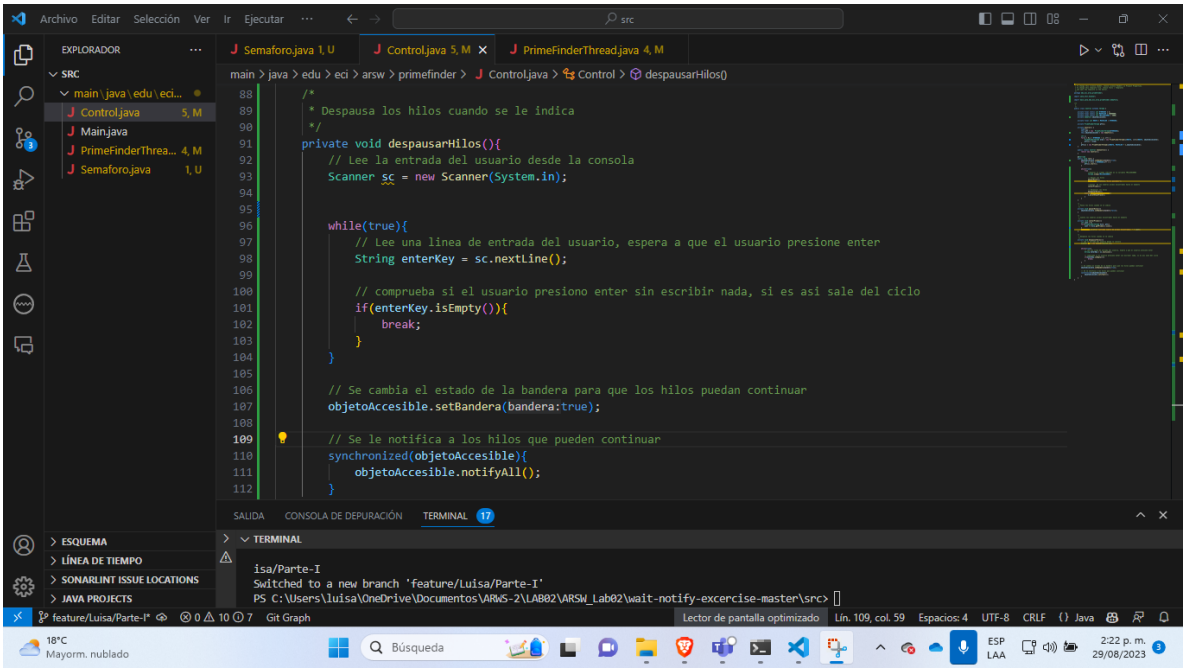
- Modificación de la clase Control

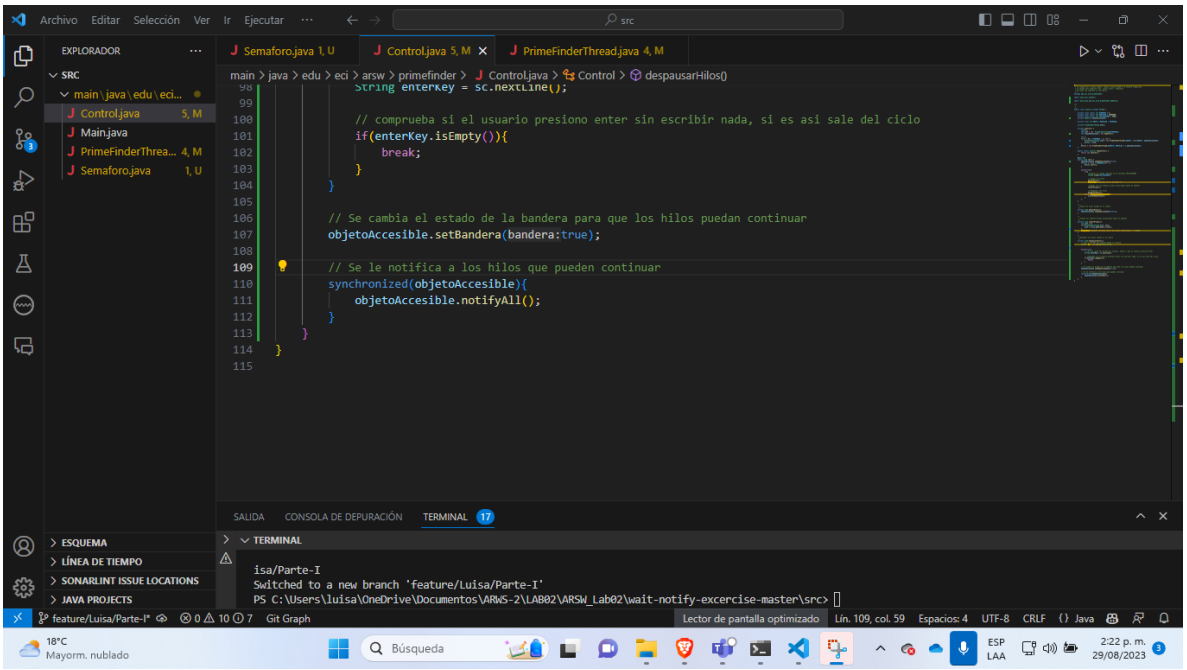



Archivo Editar Selección Ver Ir Ejecutar
src
EXPLORADOR
SRC
main \ java \ edu \ eci...
Control.java 5, M
Main.java
PrimeFinderThrea... 4, M
Semaforo.java 1, U
Semaforo.java 1, U
Control.java 5, M
PrimeFinderThread.java 4, M
main > java > edu > eci > arsw > primefinder > Control.java > Control > despausarHilos()
25
26 private Control() {
27 super();
28 this.pft = new PrimeFinderThread[NTHREADS];
29 this.objetoAccesible = new Semaforo();
30
31 int i;
32 for(i = 0;i < NTHREADS - 1; i++) {
33 PrimeFinderThread elem = new PrimeFinderThread(i*NDATA, (i+1)*NDATA, objetoAccesible);
34 pft[i] = elem;
35 }
36 pft[i] = new PrimeFinderThread(i*NDATA, MAXVALUE + 1,objetoAccesible);
37 }
38
39 public static Control newControl() {
40 return new Control();
41 }
42
43 @Override
44 public void run() {
45 objetoAccesible.setBandera(bandera:true);
46 for(int i = 0;i < NTHREADS;i++) {
47 pft[i].start();
48 }
49
SALIDA CONSOLA DE DEPURACIÓN TERMINAL 17
ESQUEMA
LÍNEA DE TIEMPO
SONARLINT ISSUE LOCATIONS
JAVA PROJECTS
TERMINAL
isa/Parte-I
Switched to a new branch 'feature/Luisa/Parte-I'
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB02\ARSW_Lab02\wait-notify-excercise-master\src>
feature/Luisa/Parte-I* 0 10 7 Git Graph
Lector de pantalla optimizado Lín. 109, col. 59 Espacios: 4 UTF-8 CRLF Java
18°C Mayorm. nublado
Búsqueda
ESP LAA
2:21 p. m. 29/08/2023

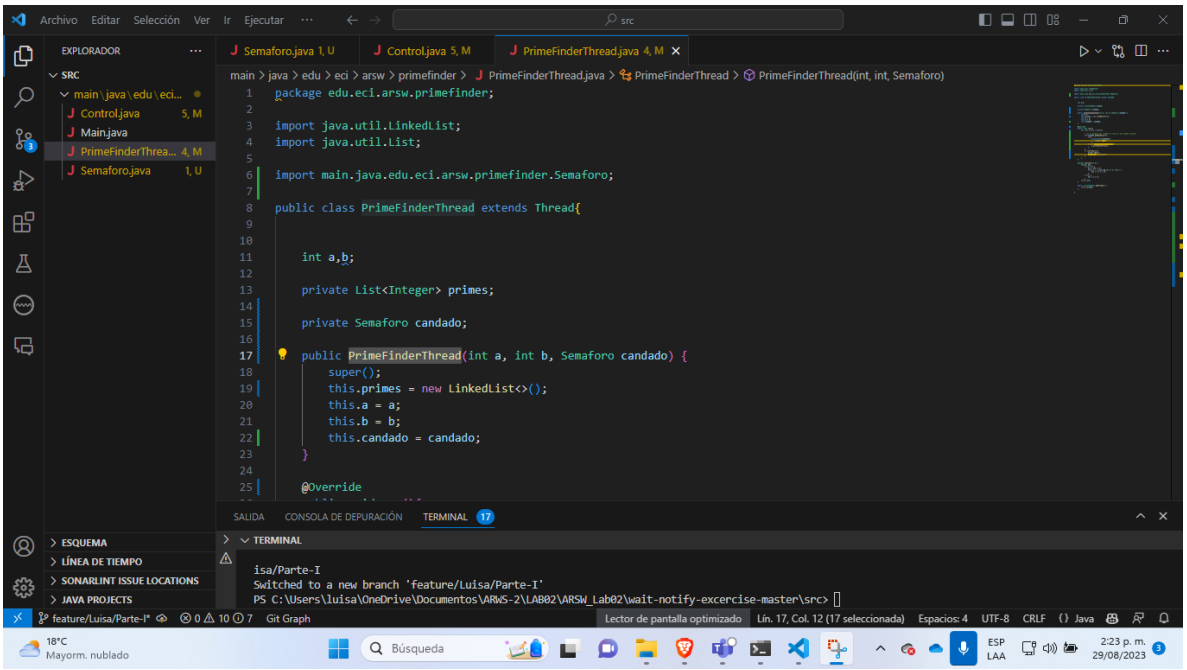

Archivo Editar Selección Ver Ir Ejecutar
src
EXPLORADOR
SRC
main \ java \ edu \ eci...
Control.java 5, M
Main.java
PrimeFinderThrea... 4, M
Semaforo.java 1, U
Semaforo.java 1, U
Control.java 5, M
PrimeFinderThread.java 4, M
main > java > edu > eci > arsw > primefinder > Control.java > Control > despausarHilos()
48 }
49
50 while(true){
51 try{
52 //Esperar el tiempo indicado en la variable TMILISECONDS
53 Thread.sleep(TMILISECONDS);
54
55 // Pausar los hilos
56 pausarHilos();
57 System.out.println(x:"Hilos pausados");
58
59 //Conteo de los numeros primos encontrados hasta el momento
60 contarPrimos();
61
62 // Despausar los hilos
63 despausarHilos();
64 }catch(InterruptedException e){
65 e.printStackTrace();
66 }
67 }
68 }
69
70 /*
71 * Pausa los hilos cuando se le indica
72 */
73 private void pausarHilos(){
SALIDA CONSOLA DE DEPURACIÓN TERMINAL 17
ESQUEMA
LÍNEA DE TIEMPO
SONARLINT ISSUE LOCATIONS
JAVA PROJECTS
TERMINAL
isa/Parte-I
Switched to a new branch 'feature/Luisa/Parte-I'
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB02\ARSW_Lab02\wait-notify-excercise-master\src>
feature/Luisa/Parte-I* 0 10 7 Git Graph
Lector de pantalla optimizado Lín. 109, col. 59 Espacios: 4 UTF-8 CRLF Java
18°C Mayorm. nublado
Búsqueda
ESP LAA
2:21 p. m. 29/08/2023

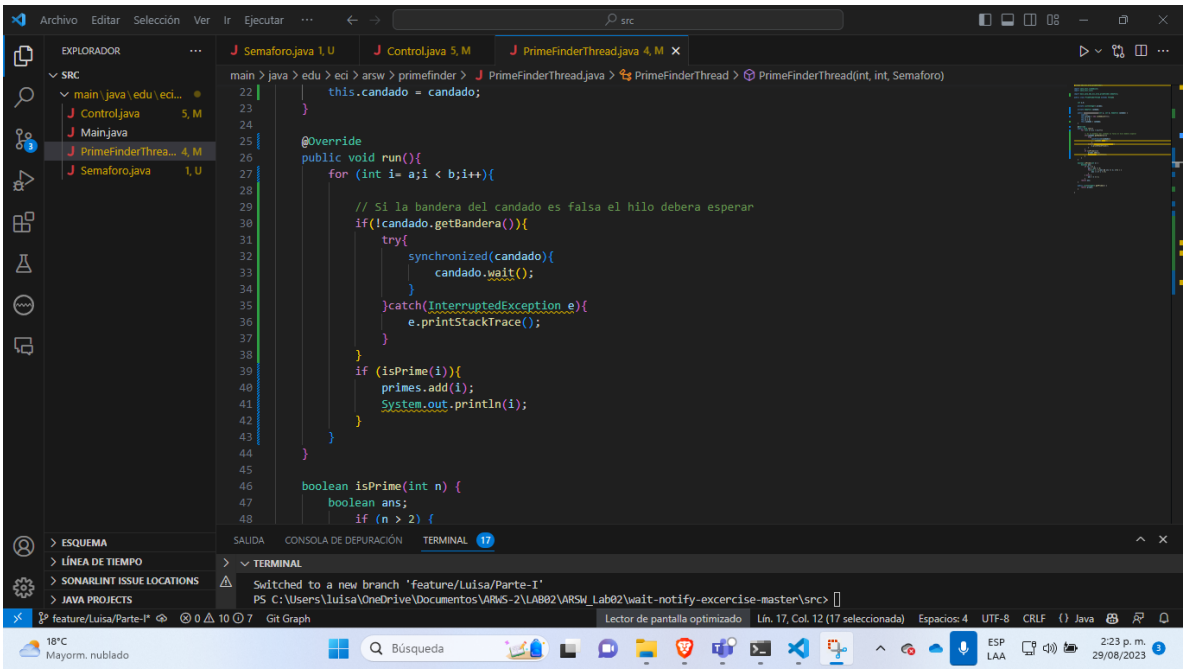

Archivo Editar Selección Ver Ir Ejecutar
src
EXPLORADOR
SRC
main \ java \ edu \ eci...
Control.java 5, M
Main.java
PrimeFinderThrea... 4, M
Semaforo.java 1, U
Semaforo.java 1, U
Control.java 5, M
PrimeFinderThread.java 4, M
main > java > edu > eci > arsw > primefinder > Control.java > Control > despausarHilos()
70 /*
71 * Pausa los hilos cuando se le indica
72 */
73 private void pausarHilos(){
74 objetoAccesible.setBandera(bandera:false);
75 }
76
77 /*
78 * Cuenta los numeros primos encontrados hasta el momento
79 */
80 private void contarPrimos(){
81 int cont = 0;
82 for(PrimeFinderThread hilo: pft){
83 cont += hilo.getPrimes().size();
84 }
85 System.out.println("Cantidad numero de primos encontrados: " + cont);
86 }
87
88 /*
89 * Despausa los hilos cuando se le indica
90 */
91 private void despausarHilos(){
92 // Lee la entrada del usuario desde la consola
93 Scanner sc = new Scanner(System.in);
94
SALIDA CONSOLA DE DEPURACIÓN TERMINAL 17
ESQUEMA
LÍNEA DE TIEMPO
SONARLINT ISSUE LOCATIONS
JAVA PROJECTS
TERMINAL
isa/Parte-I
Switched to a new branch 'feature/Luisa/Parte-I'
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB02\ARSW_Lab02\wait-notify-excercise-master\src>
feature/Luisa/Parte-I* 0 10 7 Git Graph
Lector de pantalla optimizado Lín. 109, col. 59 Espacios: 4 UTF-8 CRLF Java
18°C Mayorm. nublado
Búsqueda
ESP LAA
2:22 p. m. 29/08/2023

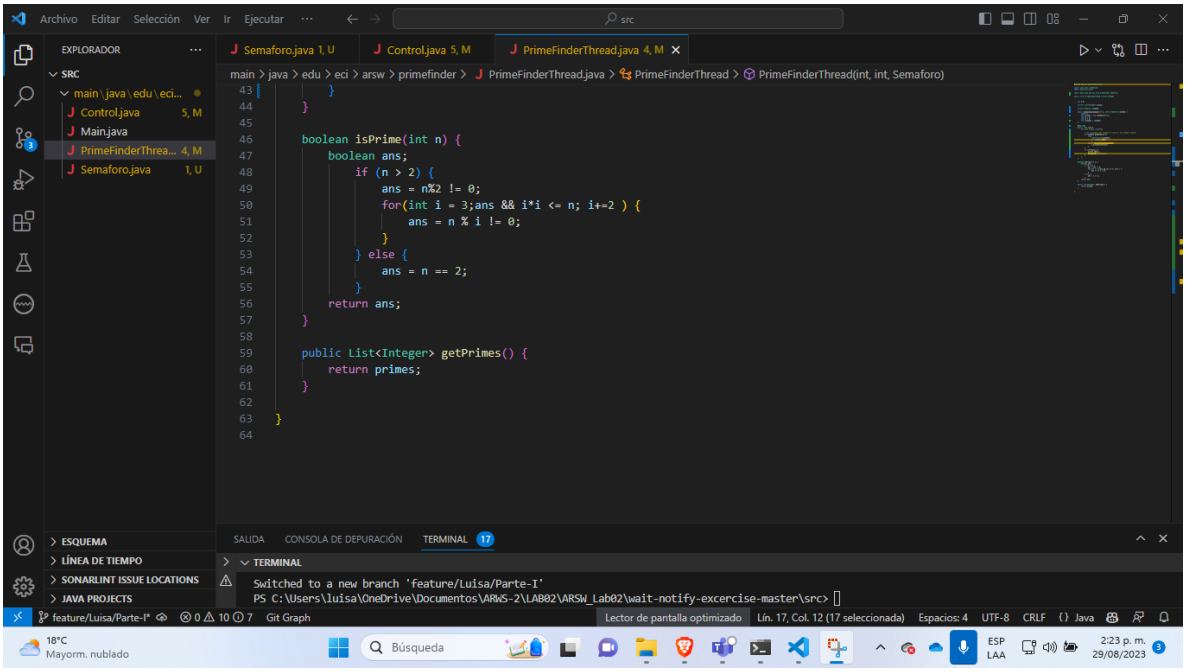

Archivo Editar Selección Ver Ir Ejecutar
src
EXPLORADOR
SRC
main \ java \ edu \ eci...
Control.java 5, M
Main.java
PrimeFinderThrea... 4, M
Semaforo.java 1, U
Semaforo.java 1, U
Control.java 5, M
PrimeFinderThread.java 4, M
main > java > edu > eci > arsw > primefinder > Control.java > Control > despausarHilos()
88 /*
89 * Despausa los hilos cuando se le indica
90 */
91 private void despausarHilos(){
92 // Lee la entrada del usuario desde la consola
93 Scanner sc = new Scanner(System.in);
94
95
96 while(true){
97 // Lee una linea de entrada del usuario, espera a que el usuario presione enter
98 String enterKey = sc.nextLine();
99
100 // comprueba si el usuario presiono enter sin escribir nada, si es asi sale del ciclo
101 if(enterKey.isEmpty()){
102 break;
103 }
104 }
105
106 // Se cambia el estado de la bandera para que los hilos puedan continuar
107 objetoAccesible.setBandera(bandera:true);
108
109 // Se le notifica a los hilos que pueden continuar
110 synchronized(objetoAccesible){
111 objetoAccesible.notifyAll();
112 }
SALIDA CONSOLA DE DEPURACIÓN TERMINAL 17
ESQUEMA
LÍNEA DE TIEMPO
SONARLINT ISSUE LOCATIONS
JAVA PROJECTS
TERMINAL
isa/Parte-I
Switched to a new branch 'feature/Luisa/Parte-I'
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB02\ARSW_Lab02\wait-notify-excercise-master\src>
feature/Luisa/Parte-I* 0 10 7 Git Graph
Lector de pantalla optimizado Lín. 109, col. 59 Espacios: 4 UTF-8 CRLF Java
18°C Mayorm. nublado
Búsqueda
ESP LAA
2:22 p. m. 29/08/2023



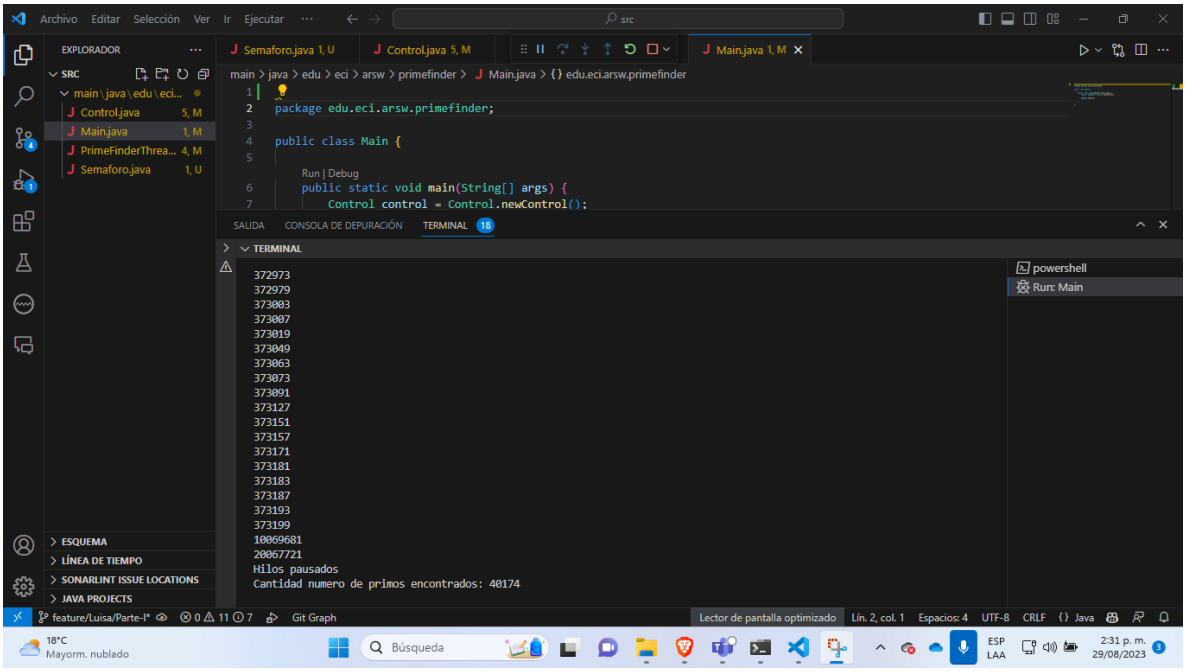
➢ Modificación de la clase PrimeFinderThread

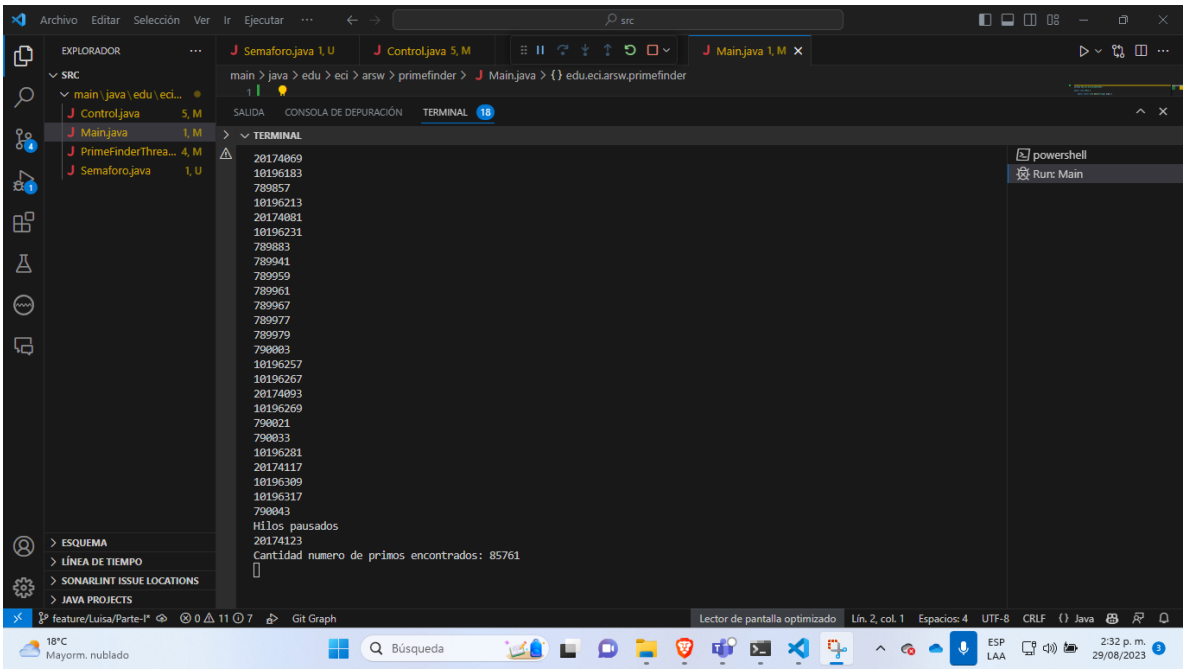



Archivo Editar Selección Ver Ir Ejecutar
src
EXPLORADOR
SRC
main \ java \ edu \ eci...
Control.java 5, M
Main.java
PrimeFinderThrea... 4, M
Semaforo.java 1, U
Semaforo.java 1, U
Control.java 5, M
PrimeFinderThread.java 4, M
main > java > edu > eci > arsw > primefinder > PrimeFinderThread.java > PrimeFinderThread > PrimeFinderThread(int, int, Semaforo)
22 this.candado = candado;
23 }
24
25 @Override
26 public void run(){
27 for (int i= a;i < b;i++){
28
29 // Si la bandera del candado es falsa el hilo debera esperar
30 if(!candado.getBandera()){
31 try{
32 synchronized(candado){
33 candado.wait();
34 }
35 }catch(InterruptedException e){
36 e.printStackTrace();
37 }
38 }
39 if (isPrime(i)){
40 primes.add(i);
41 System.out.println(i);
42 }
43 }
44 }
45
46 boolean isPrime(int n) {
47 boolean ans;
48 if (n > 2) {
ESQUEMA
LÍNEA DE TIEMPO
SONARLINT ISSUE LOCATIONS
JAVA PROJECTS
SALIDA CONSOLA DE DEPURACIÓN TERMINAL 17
TERMINAL
Switched to a new branch 'feature/Luisa/Parte-I'
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB02\ARSW_Lab02\wait-notify-excercise-master\src>
feature/Luisa/Parte-I* 0 10 7 Git Graph
Lector de pantalla optimizado Lín. 17, Col. 12 (17 seleccionada) Espacios: 4 UTF-8 CRLF Java
18°C Mayorm. nublado
Búsqueda
ESP LAA
2:23 p. m. 29/08/2023


Archivo Editar Selección Ver Ir Ejecutar
src
EXPLORADOR
SRC
main \ java \ edu \ eci...
Control.java 5, M
Main.java
PrimeFinderThrea... 4, M
Semaforo.java 1, U
Semaforo.java 1, U
Control.java 5, M
PrimeFinderThread.java 4, M
main > java > edu > eci > arsw > primefinder > PrimeFinderThread.java > PrimeFinderThread > PrimeFinderThread(int, int, Semaforo)
43 }
44 }
45
46 boolean isPrime(int n) {
47 boolean ans;
48 if (n > 2) {
49 ans = n%2 != 0;
50 for(int i = 3;ans && i*i <= n; i+=2) {
51 ans = n % i != 0;
52 }
53 } else {
54 ans = n == 2;
55 }
56 return ans;
57 }
58
59 public List<Integer> getPrimes() {
60 return primes;
61 }
62
63 }
64
ESQUEMA
LÍNEA DE TIEMPO
SONARLINT ISSUE LOCATIONS
JAVA PROJECTS
SALIDA CONSOLA DE DEPURACIÓN TERMINAL 17
TERMINAL
Switched to a new branch 'feature/Luisa/Parte-I'
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB02\ARSW_Lab02\wait-notify-excercise-master\src>
feature/Luisa/Parte-I* 0 10 7 Git Graph
Lector de pantalla optimizado Lín. 17, Col. 12 (17 seleccionada) Espacios: 4 UTF-8 CRLF Java
18°C Mayorm. nublado
Búsqueda
ESP LAA
2:23 p. m. 29/08/2023

➢ Resultados ejecución:

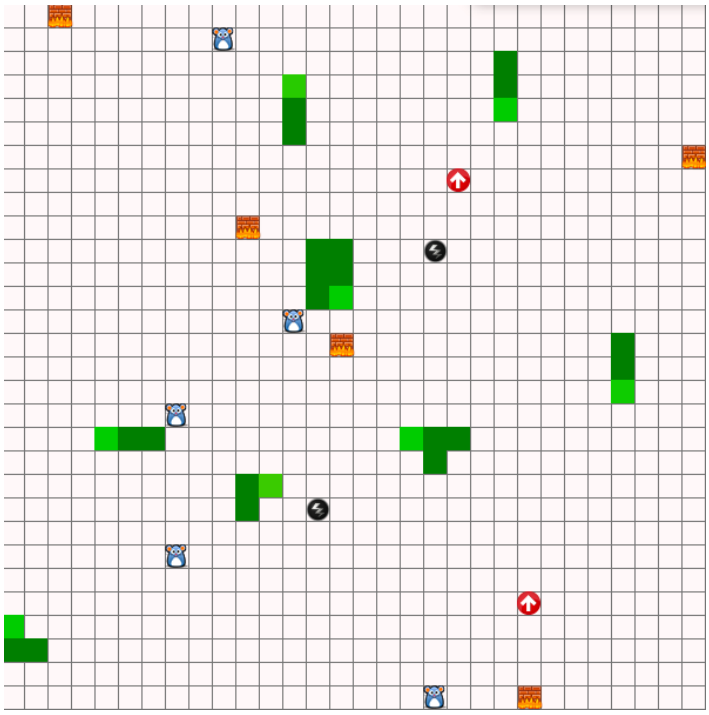




Parte II

SnakeRace es una versión autónoma, multi-serpiente del famoso juego 'snake', basado en el proyecto de João Andrade -este ejercicio es un 'fork' del mismo-. En este juego:

- N serpientes funcionan de manera autónoma.
- No existe el concepto de colisión entre las mismas. La única forma de que mueran es estrellándose contra un muro.
- Hay ratones distribuídos a lo largo del juego. Como en el juego clásico, cada vez que una serpiente se come a un ratón, ésta crece.
- Existen unos puntos (flechas rojas) que teletransportan a las serpientes.
- Los rayos hacen que la serpiente aumente su velocidad.



Ejercicio

1. Analice el código para entender cómo hace uso de hilos para crear un comportamiento autónomo de las N serpientes.

   La clase Snake implementa la interfaz Runnable, lo que significa que cada instancia de serpiente puede ejecutarse en su propio hilo el cual ejecuta la lógica de movimiento y comportamiento de cada una de las serpientes.

   SnakeApp crea instancias de serpientes y hilos para cada serpiente, permitiendo que todas las serpientes se muevan y actúen autónomamente en paralelo.

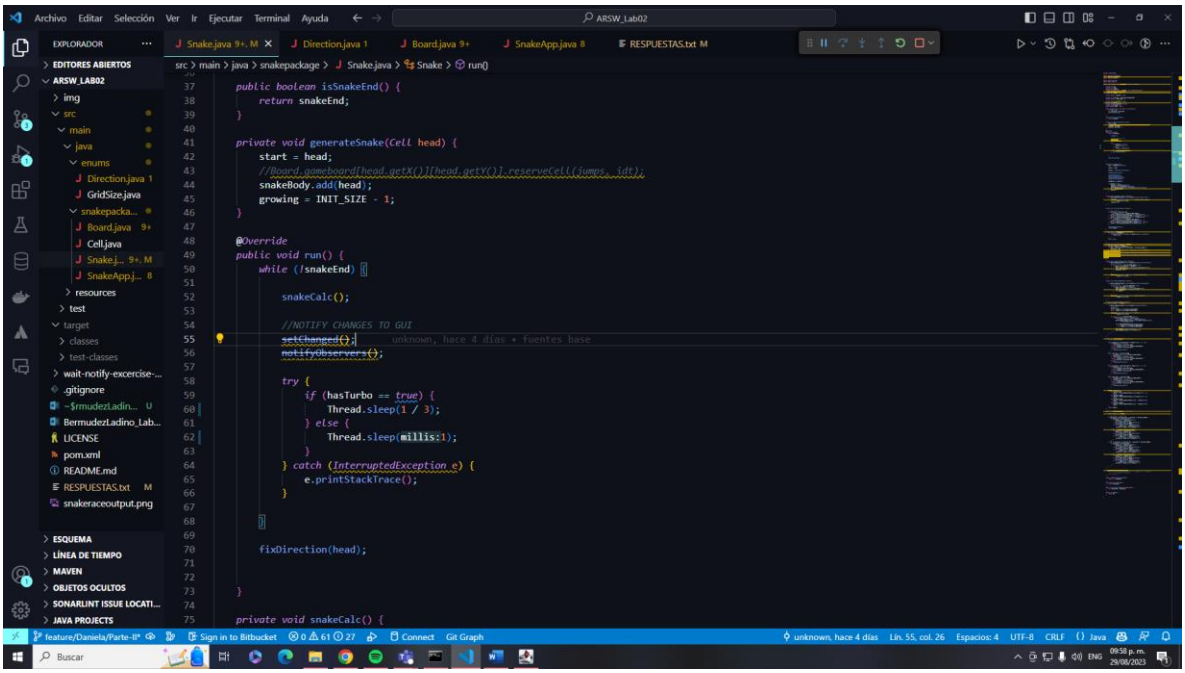
2. De acuerdo con lo anterior, y con la lógica del juego, identifique y escriba claramente (archivo RESPUESTAS.txt):

   - Posibles condiciones de carrera.

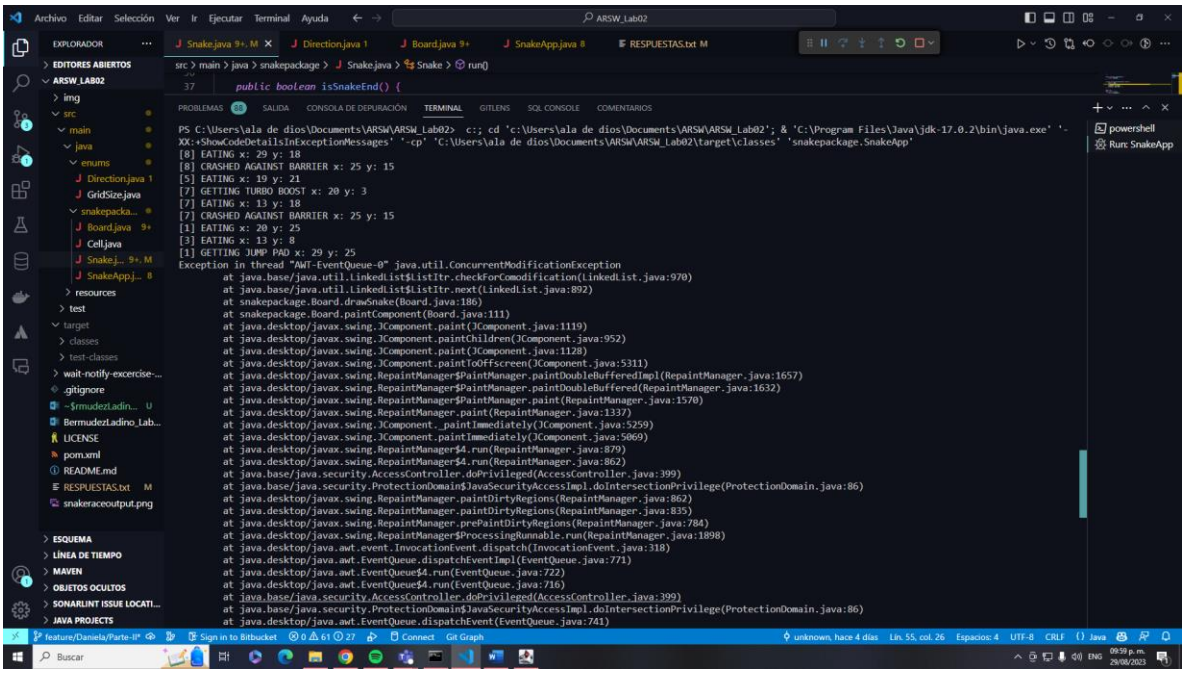
     1. Revisar si una celda esta libre o no.
     2. Si hay comida disponible en una casilla.
     3. Si una flecha roja está disponible o no.
     4. Si un rayo está disponible o no.

   - Uso inadecuado de colecciones, considerando su manejo concurrente (para esto, aumente la velocidad del juego y ejecútelo varias veces hasta que se genere un error).

     Dentro de la clase Snake, ajustamos los valores de los tiempos de espera en el método run(). Para esto reducimos los valores cambiando de 500 milisegundos a 1 milisegundo dentro del método Thread.sleep() para que las serpientes se movieran más rápido.



     Como podemos ver en la siguiente imagen luego de ejecutar el juego el error que sale es ConcurrentModificationException, este error se puede dar en el momento en que se está dibujando la serpiente ya que la referencia que pide el tablero para dibujarla se está iterando mientras la serpiente se modifica (agregando o eliminando elementos).



- Uso innecesario de esperas activas.

  La espera innecesaria está en la clase SnakeApp en el while del metodo init, ya que este bucle verifica constantemente si todas las serpientes han llegado al final del juego. Cuando todas las serpientes hayan terminado, el bucle se rompe.

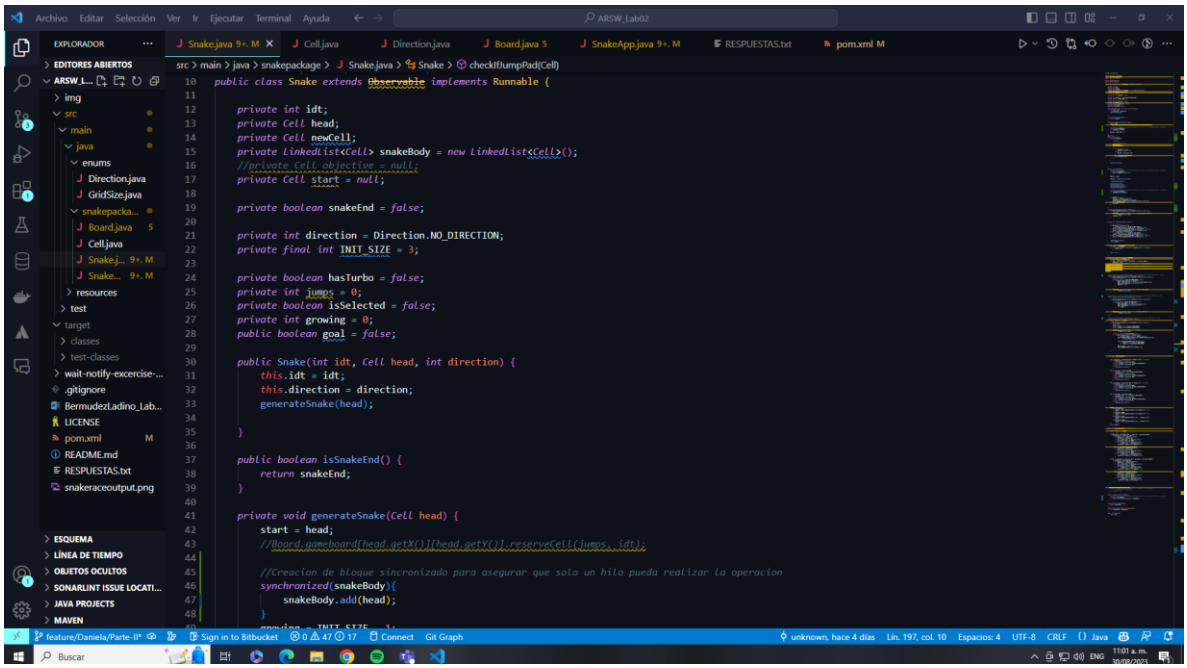
3. Identifique las regiones críticas asociadas a las condiciones de carrera, y haga algo para eliminarlas. Tenga en cuenta que se debe sincronizar estrictamente LO NECESARIO. En su documento de respuestas indique, la solución realizada para cada ítem del punto 2. Igualmente tenga en cuenta que en los siguientes puntos NO se deben agregar más posibles condiciones de carrera.

   Para evitar las posibles condiciones de carrera que se encontraban en la clase Snake mencionadas anteriormente:

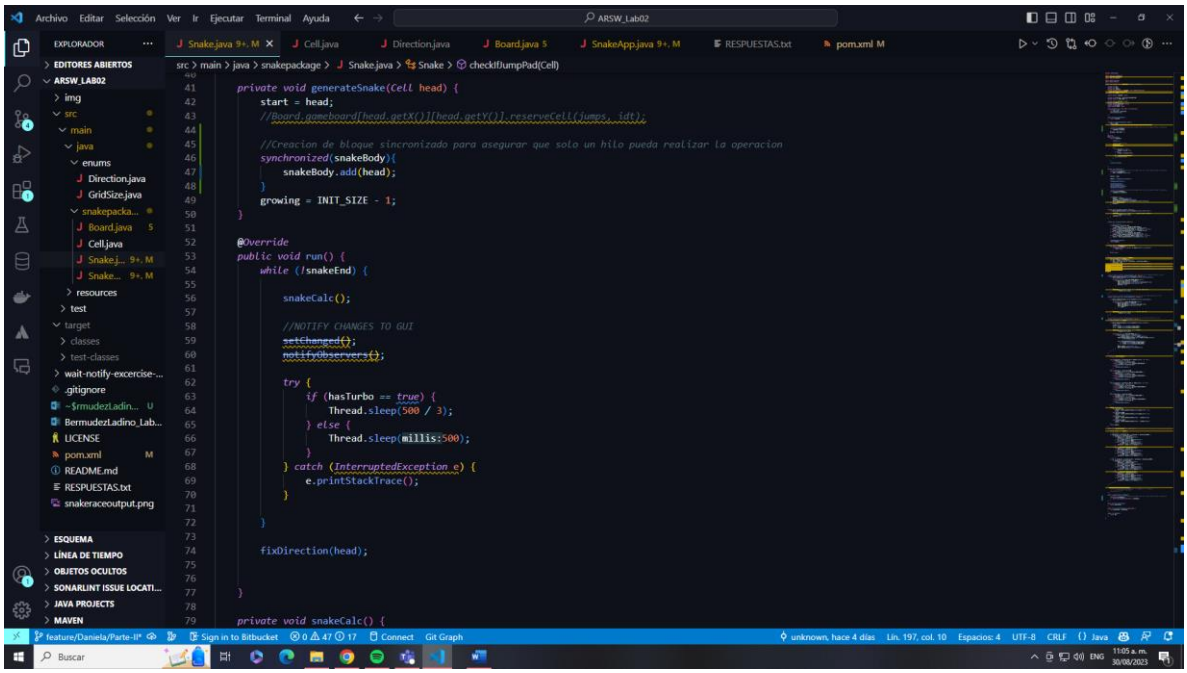
   - Utilizamos el **bloque synchronized** para sincronizar el acceso a la colección snakeBody en todos los lugares donde se modifica o itera sobre esta.

   - Sincronizamos los métodos checkIfTurboBoost, chekIfJumpPad y checkIfFood utilizando la palabra clave synchronized. De esta manera se asegura que solo un hilo pueda ejecutar cualquiera de estos métodos en un momento dado.

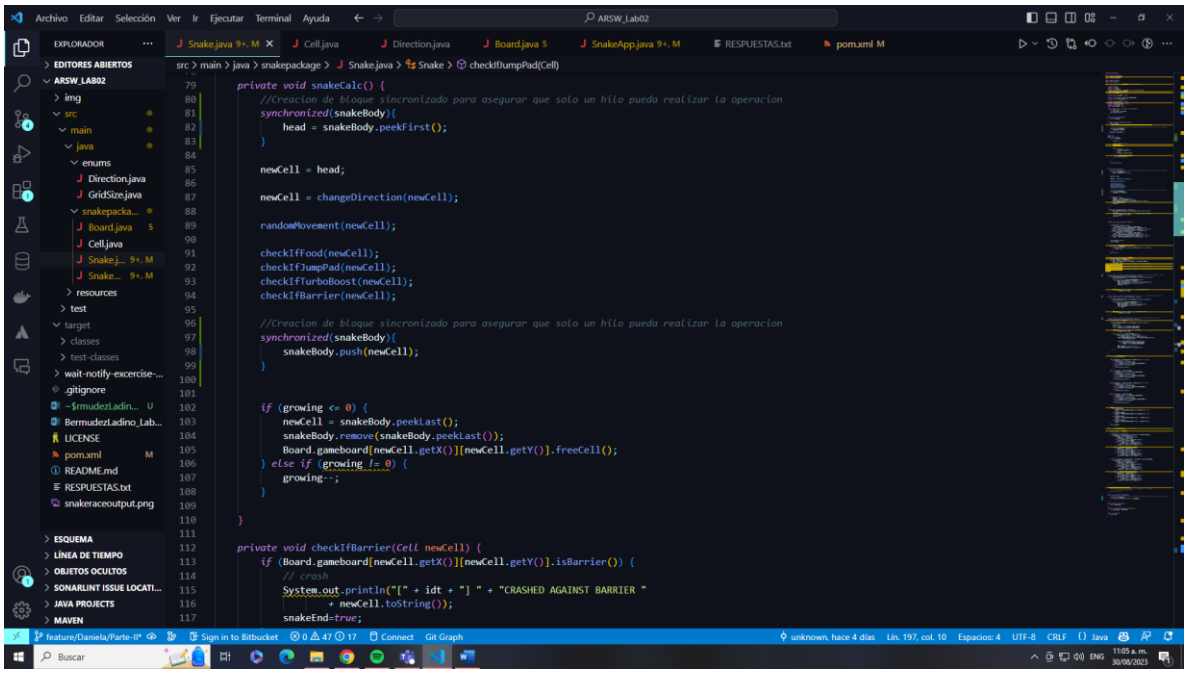
- Modificación de la clase Snake



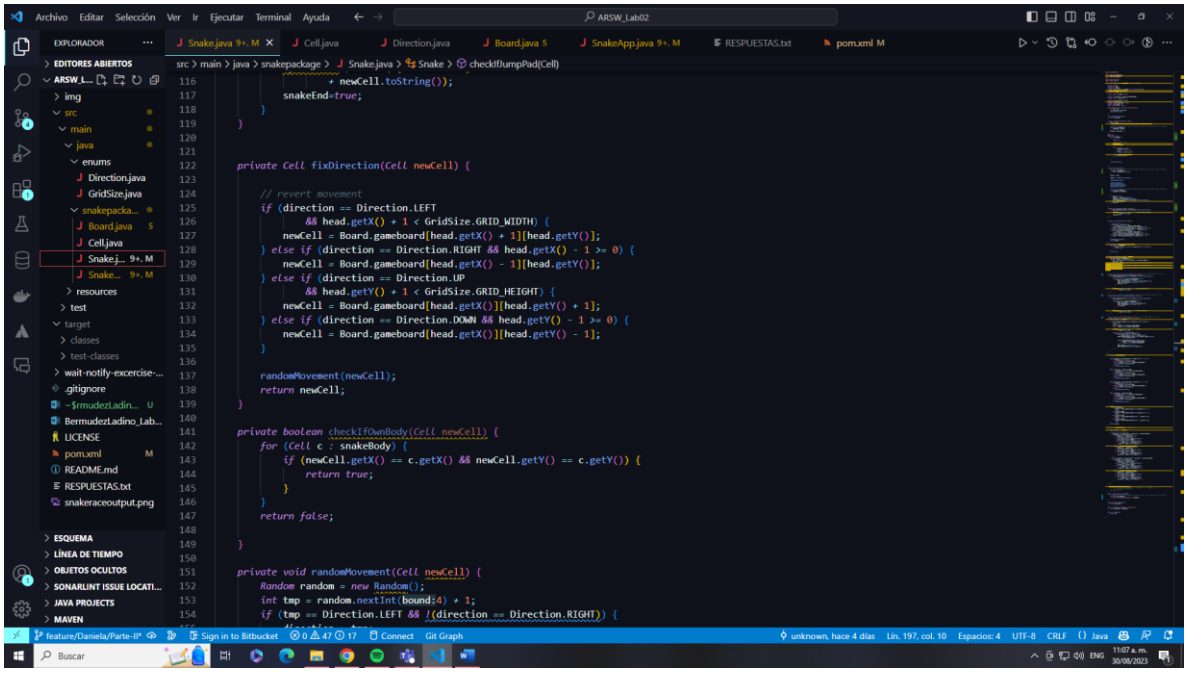
```
public class Snake extends Observable implements Runnable {

    private int idt;
    private Cell head;
    private Cell newCell;
    private LinkedList<Cell> snakeBody = new LinkedList<Cell>();
    //private Cell objective = null;
    private Cell start = null;

    private boolean snakeEnd = false;

    private int direction = Direction.NO_DIRECTION;
    private final int INIT_SIZE = 3;

    private boolean hasTurbo = false;
    private int jumps = 0;
    private boolean isSelected = false;
    private int growing = 0;
    public boolean goal = false;

    public Snake(int idt, Cell head, int direction) {
        this.idt = idt;
        this.direction = direction;
        generateSnake(head);

    }

    public boolean isSnakeEnd() {
        return snakeEnd;
    }

    private void generateSnake(Cell head) {
        start = head;
        //Board.gameboard[head.getX()][head.getY()].reserveCell(jumps, idt);

        //Creacion de bloque sincronizado para asegurar que solo un hilo pueda realizar la operacion
        synchronized(snakeBody){
            snakeBody.add(head);
        }
```



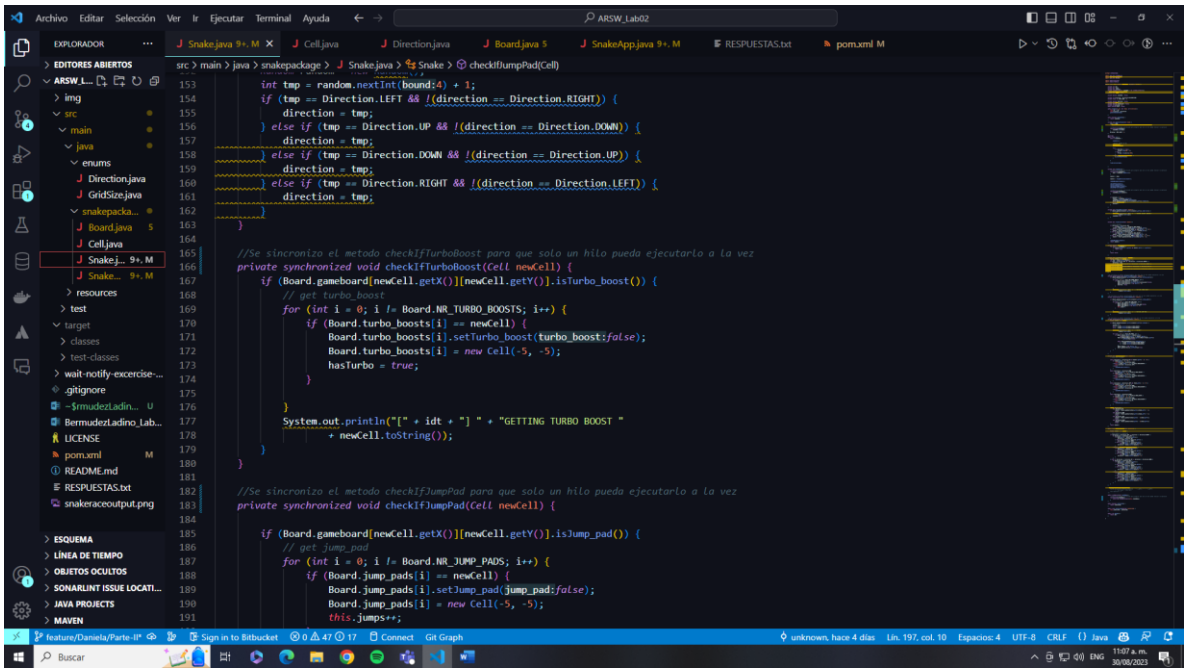
```
    private void generateSnake(Cell head) {
        start = head;
        //Board.gameboard[head.getX()][head.getY()].reserveCell(jumps, idt);

        //Creacion de bloque sincronizado para asegurar que solo un hilo pueda realizar la operacion
        synchronized(snakeBody){
            snakeBody.add(head);
        }
        growing = INIT_SIZE - 1;
    }

    @Override
    public void run() {
        while (!snakeEnd) {

            snakeCalc();

            //NOTIFY CHANGES TO GUI
            setChanged();
            notifyObservers();

            try {
                if (hasTurbo == true) {
                    Thread.sleep(500 / 3);
                } else {
                    Thread.sleep(millis:500);
                }
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

        }

        fixDirection(head);


    }

    private void snakeCalc() {
```



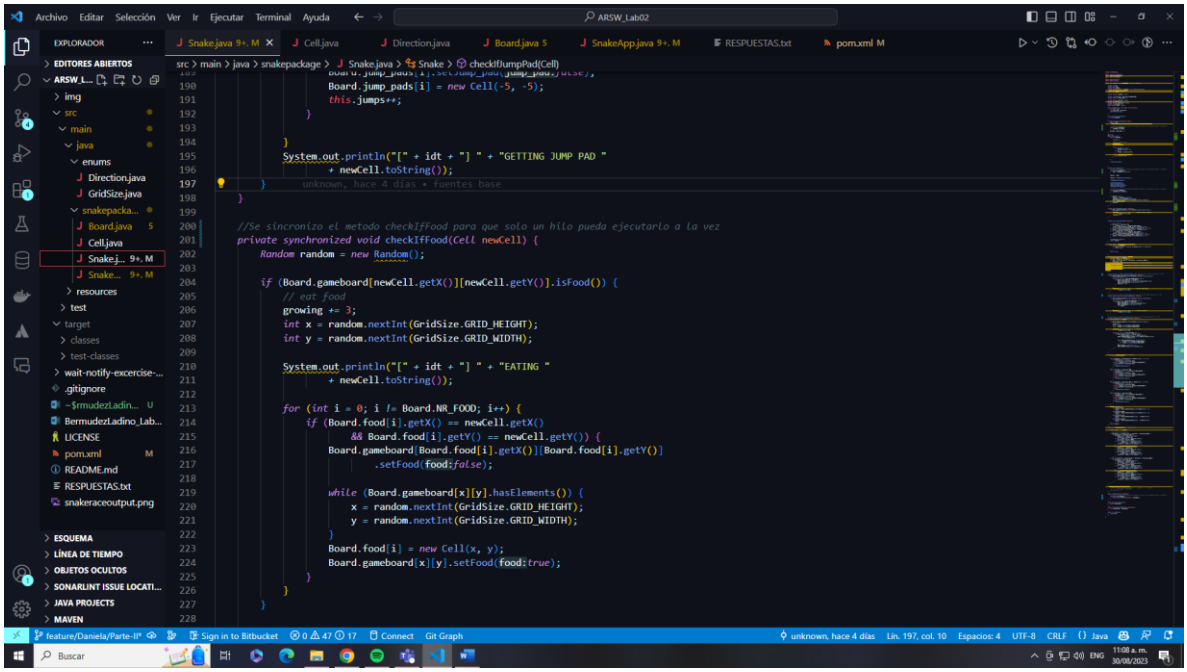
```
    private void snakeCalc() {
        //Creacion de bloque sincronizado para asegurar que solo un hilo pueda realizar la operacion
        synchronized(snakeBody){
            head = snakeBody.peekFirst();
        }

        newCell = head;

        newCell = changeDirection(newCell);

        randomMovement(newCell);

        checkIfFood(newCell);
        checkIfJumpPad(newCell);
        checkIfTurboBoost(newCell);
        checkIfBarrier(newCell);

        //Creacion de bloque sincronizado para asegurar que solo un hilo pueda realizar la operacion
        synchronized(snakeBody){
            snakeBody.push(newCell);
        }


        if (growing <= 0) {
            newCell = snakeBody.peekLast();
            snakeBody.remove(snakeBody.peekLast());
            Board.gameboard[newCell.getX()][newCell.getY()].freeCell();
        } else if (growing != 0) {
            growing--;
        }

    }

    private void checkIfBarrier(Cell newCell) {
        if (Board.gameboard[newCell.getX()][newCell.getY()].isBarrier()) {
            // crash
            System.out.println("[" + idt + "] " + "CRASHED AGAINST BARRIER "
                    + newCell.toString());
            snakeEnd=true;
```



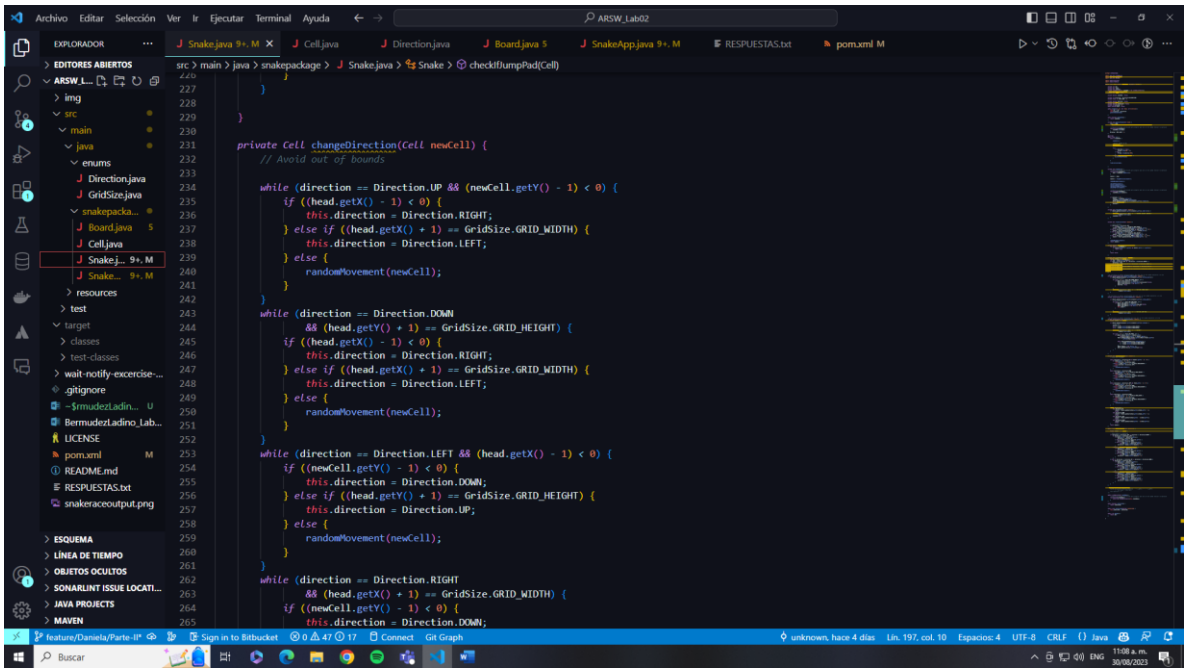
```
                    + newCell.toString());
            snakeEnd=true;
        }
    }


    private Cell fixDirection(Cell newCell) {

        // revert movement
        if (direction == Direction.LEFT
                && head.getX() + 1 < GridSize.GRID_WIDTH) {
            newCell = Board.gameboard[head.getX() + 1][head.getY()];
        } else if (direction == Direction.RIGHT && head.getX() - 1 >= 0) {
            newCell = Board.gameboard[head.getX() - 1][head.getY()];
        } else if (direction == Direction.UP
                && head.getY() + 1 < GridSize.GRID_HEIGHT) {
            newCell = Board.gameboard[head.getX()][head.getY() + 1];
        } else if (direction == Direction.DOWN && head.getY() - 1 >= 0) {
            newCell = Board.gameboard[head.getX()][head.getY() - 1];
        }

        randomMovement(newCell);
        return newCell;
    }

    private boolean checkIfOwnBody(Cell newCell) {
        for (Cell c : snakeBody) {
            if (newCell.getX() == c.getX() && newCell.getY() == c.getY()) {
                return true;
            }
        }
        return false;

    }

    private void randomMovement(Cell newCell) {
        Random random = new Random();
        int tmp = random.nextInt(bound:4) + 1;
        if (tmp == Direction.LEFT && !(direction == Direction.RIGHT)) {
```



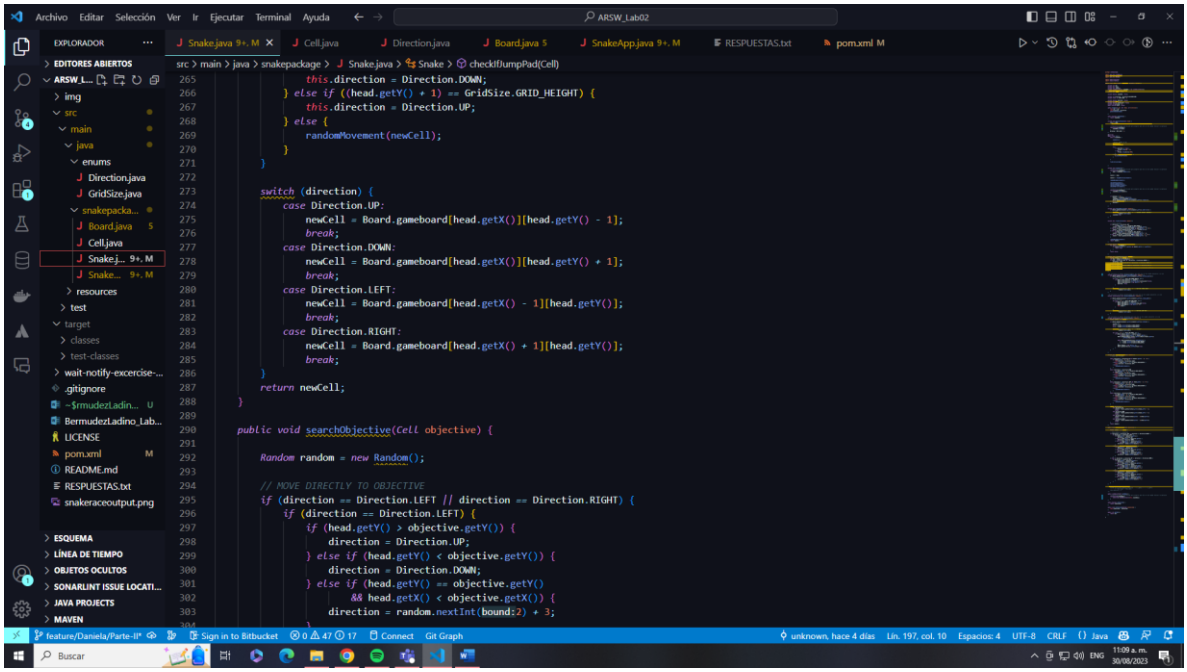
```
        int tmp = random.nextInt(bound:4) + 1;
        if (tmp == Direction.LEFT && !(direction == Direction.RIGHT)) {
            direction = tmp;
        } else if (tmp == Direction.UP && !(direction == Direction.DOWN)) {
            direction = tmp;
        } else if (tmp == Direction.DOWN && !(direction == Direction.UP)) {
            direction = tmp;
        } else if (tmp == Direction.RIGHT && !(direction == Direction.LEFT)) {
            direction = tmp;
        }
    }

    //Se sincronizo el metodo checkIfTurboBoost para que solo un hilo pueda ejecutarlo a la vez
    private synchronized void checkIfTurboBoost(Cell newCell) {
        if (Board.gameboard[newCell.getX()][newCell.getY()].isTurbo_boost()) {
            // get turbo_boost
            for (int i = 0; i != Board.NR_TURBO_BOOSTS; i++) {
                if (Board.turbo_boosts[i] == newCell) {
                    Board.turbo_boosts[i].setTurbo_boost(turbo_boost:false);
                    Board.turbo_boosts[i] = new Cell(-5, -5);
                    hasTurbo = true;
                }

            }
            System.out.println("[" + idt + "] " + "GETTING TURBO BOOST "
                    + newCell.toString());
        }
    }

    //Se sincronizo el metodo checkIfJumpPad para que solo un hilo pueda ejecutarlo a la vez
    private synchronized void checkIfJumpPad(Cell newCell) {

        if (Board.gameboard[newCell.getX()][newCell.getY()].isJump_pad()) {
            // get jump_pad
            for (int i = 0; i != Board.NR_JUMP_PADS; i++) {
                if (Board.jump_pads[i] == newCell) {
                    Board.jump_pads[i].setJump_pad(jump_pad:false);
                    Board.jump_pads[i] = new Cell(-5, -5);
                    this.jumps++;
```



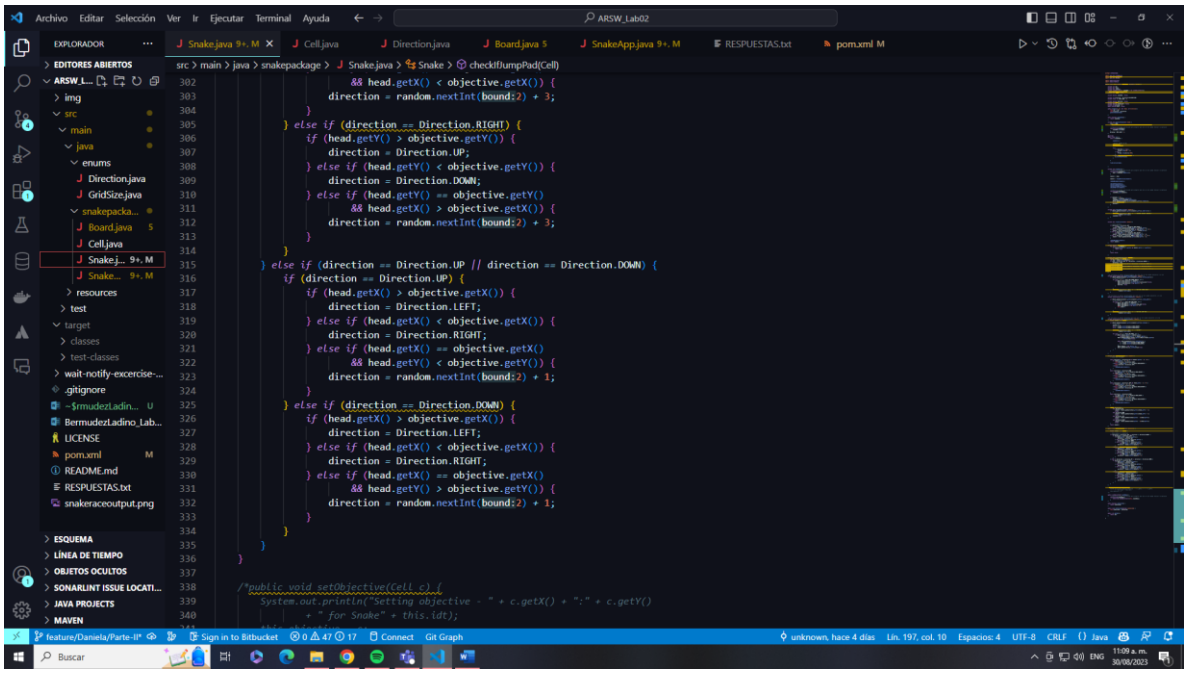
```
                    Board.jump_pads[i] = new Cell(-5, -5);
                    this.jumps++;
                }

            }
            System.out.println("[" + idt + "] " + "GETTING JUMP PAD "
                    + newCell.toString());
        }       unknown, hace 4 días • fuentes base
    }

    //Se sincronizo el metodo checkIfFood para que solo un hilo pueda ejecutarlo a la vez
    private synchronized void checkIfFood(Cell newCell) {
        Random random = new Random();

        if (Board.gameboard[newCell.getX()][newCell.getY()].isFood()) {
            // eat food
            growing += 3;
            int x = random.nextInt(GridSize.GRID_HEIGHT);
            int y = random.nextInt(GridSize.GRID_WIDTH);

            System.out.println("[" + idt + "] " + "EATING "
                    + newCell.toString());

            for (int i = 0; i != Board.NR_FOOD; i++) {
                if (Board.food[i].getX() == newCell.getX()
                        && Board.food[i].getY() == newCell.getY()) {
                    Board.gameboard[Board.food[i].getX()][Board.food[i].getY()]
                            .setFood(food:false);

                    while (Board.gameboard[x][y].hasElements()) {
                        x = random.nextInt(GridSize.GRID_HEIGHT);
                        y = random.nextInt(GridSize.GRID_WIDTH);
                    }
                    Board.food[i] = new Cell(x, y);
                    Board.gameboard[x][y].setFood(food:true);
                }
            }
        }
```

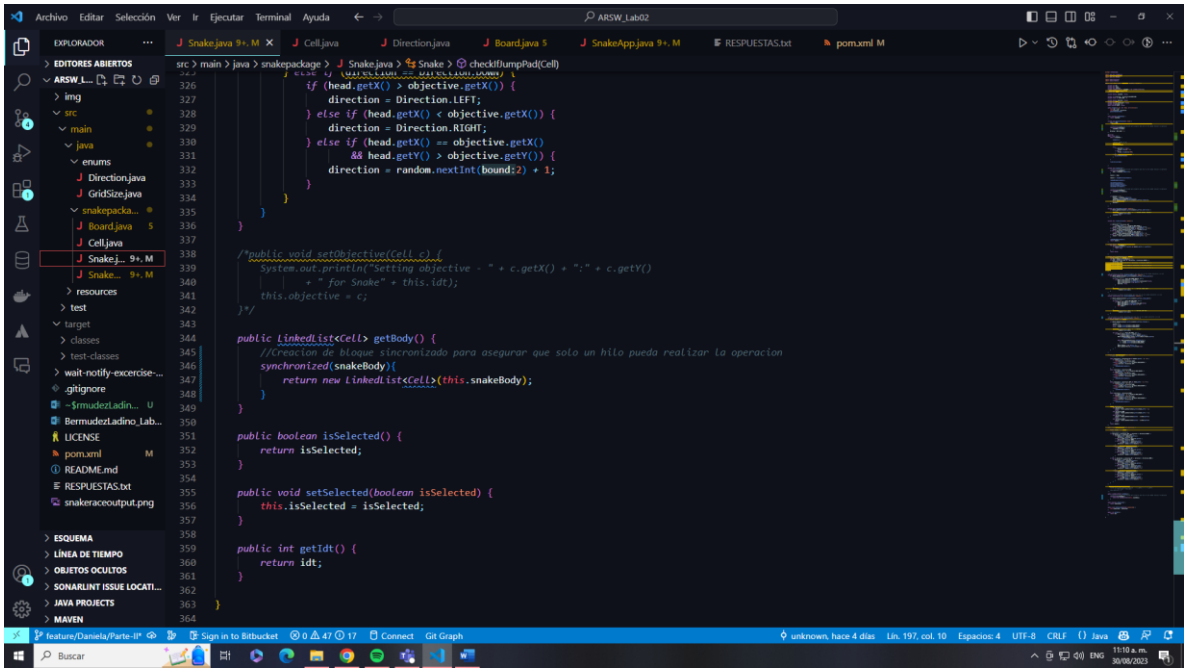


```
        }
    }

    private Cell changeDirection(Cell newCell) {
        // Avoid out of bounds

        while (direction == Direction.UP && (newCell.getY() - 1) < 0) {
            if ((head.getX() - 1) < 0) {
                this.direction = Direction.RIGHT;
            } else if ((head.getX() + 1) == GridSize.GRID_WIDTH) {
                this.direction = Direction.LEFT;
            } else {
                randomMovement(newCell);
            }
        }
        while (direction == Direction.DOWN
                && (head.getY() + 1) == GridSize.GRID_HEIGHT) {
            if ((head.getX() - 1) < 0) {
                this.direction = Direction.RIGHT;
            } else if ((head.getX() + 1) == GridSize.GRID_WIDTH) {
                this.direction = Direction.LEFT;
            } else {
                randomMovement(newCell);
            }
        }
        while (direction == Direction.LEFT && (head.getX() - 1) < 0) {
            if ((newCell.getY() - 1) < 0) {
                this.direction = Direction.DOWN;
            } else if ((head.getY() + 1) == GridSize.GRID_HEIGHT) {
                this.direction = Direction.UP;
            } else {
                randomMovement(newCell);
            }
        }
        while (direction == Direction.RIGHT
                && (head.getX() + 1) == GridSize.GRID_WIDTH) {
            if ((newCell.getY() - 1) < 0) {
                this.direction = Direction.DOWN;
```



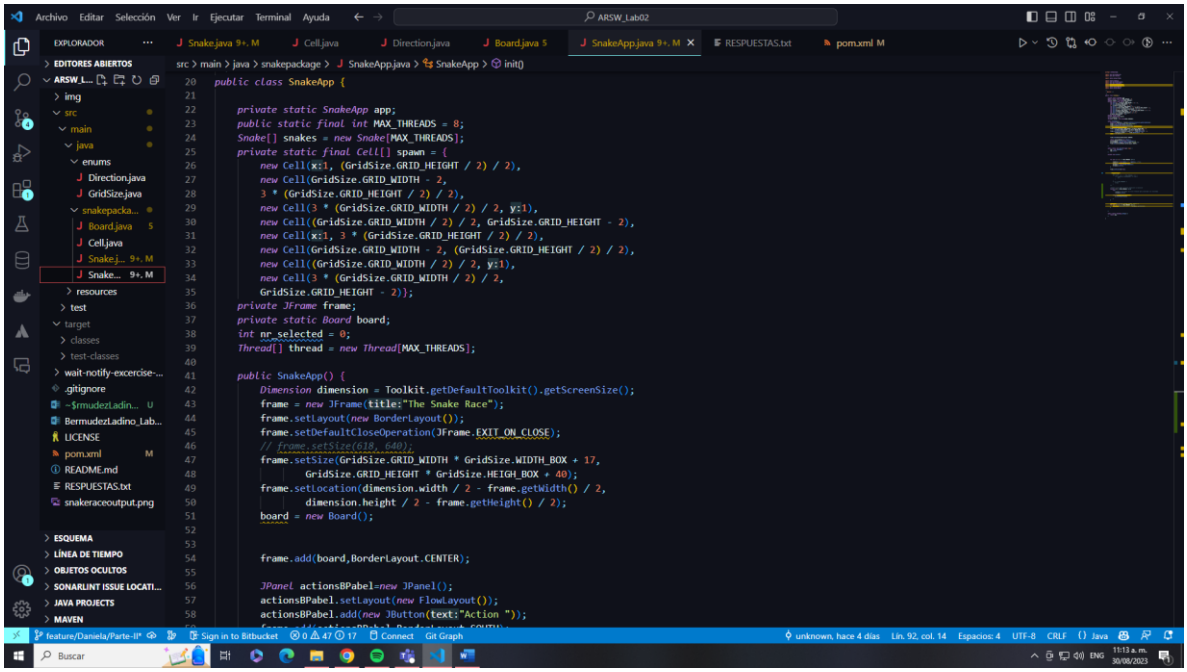
```
                this.direction = Direction.DOWN;
            } else if ((head.getY() + 1) == GridSize.GRID_HEIGHT) {
                this.direction = Direction.UP;
            } else {
                randomMovement(newCell);
            }
        }

        switch (direction) {
            case Direction.UP:
                newCell = Board.gameboard[head.getX()][head.getY() - 1];
                break;
            case Direction.DOWN:
                newCell = Board.gameboard[head.getX()][head.getY() + 1];
                break;
            case Direction.LEFT:
                newCell = Board.gameboard[head.getX() - 1][head.getY()];
                break;
            case Direction.RIGHT:
                newCell = Board.gameboard[head.getX() + 1][head.getY()];
                break;
        }
        return newCell;
    }

    public void searchObjective(Cell objective) {

        Random random = new Random();

        // MOVE DIRECTLY TO OBJECTIVE
        if (direction == Direction.LEFT || direction == Direction.RIGHT) {
            if (direction == Direction.LEFT) {
                if (head.getY() > objective.getY()) {
                    direction = Direction.UP;
                } else if (head.getY() < objective.getY()) {
                    direction = Direction.DOWN;
                } else if (head.getY() == objective.getY()
                        && head.getX() < objective.getX()) {
                    direction = random.nextInt(bound:2) + 3;
```



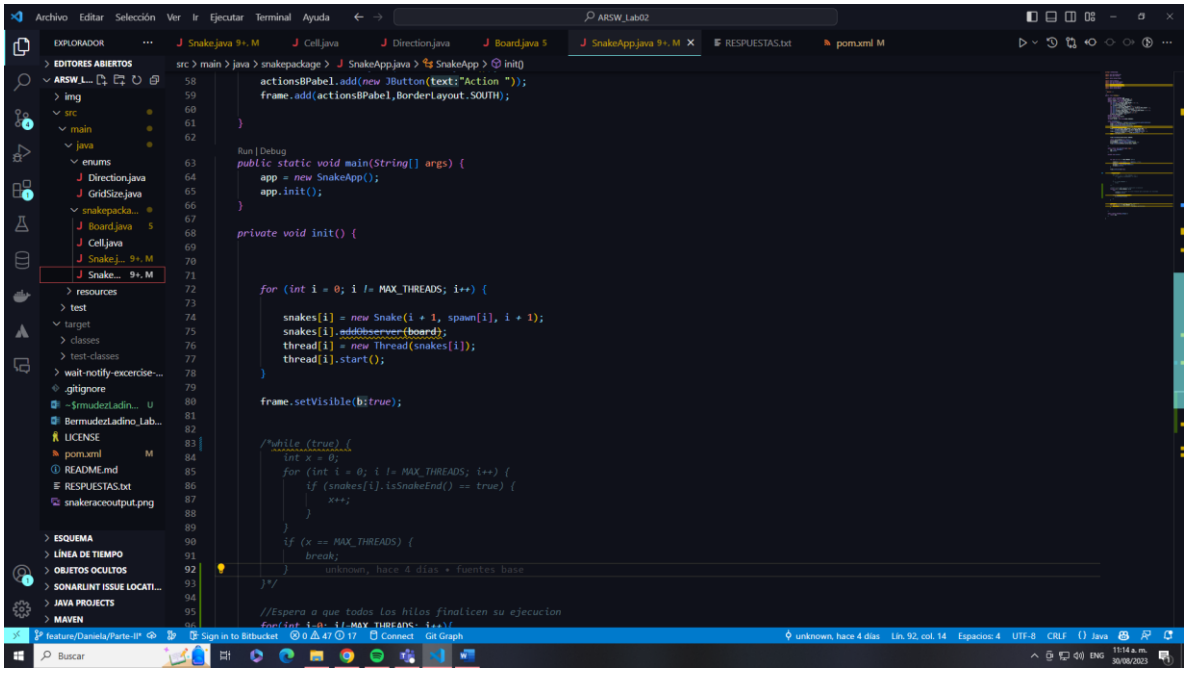


Se modifico la clase SnakeApp realizando un bucle for para esperar que todos los hilos de las serpientes finalicen su ejecución. Dentro de este bucle, llamamos el método join en cada hilo para que cuando todas las serpientes terminen se consoliden los resultados y así poder imprimir el estado de los hilos. A demás de esto comentamos el while del método init para evitar el uso innecesario de esperas activas.

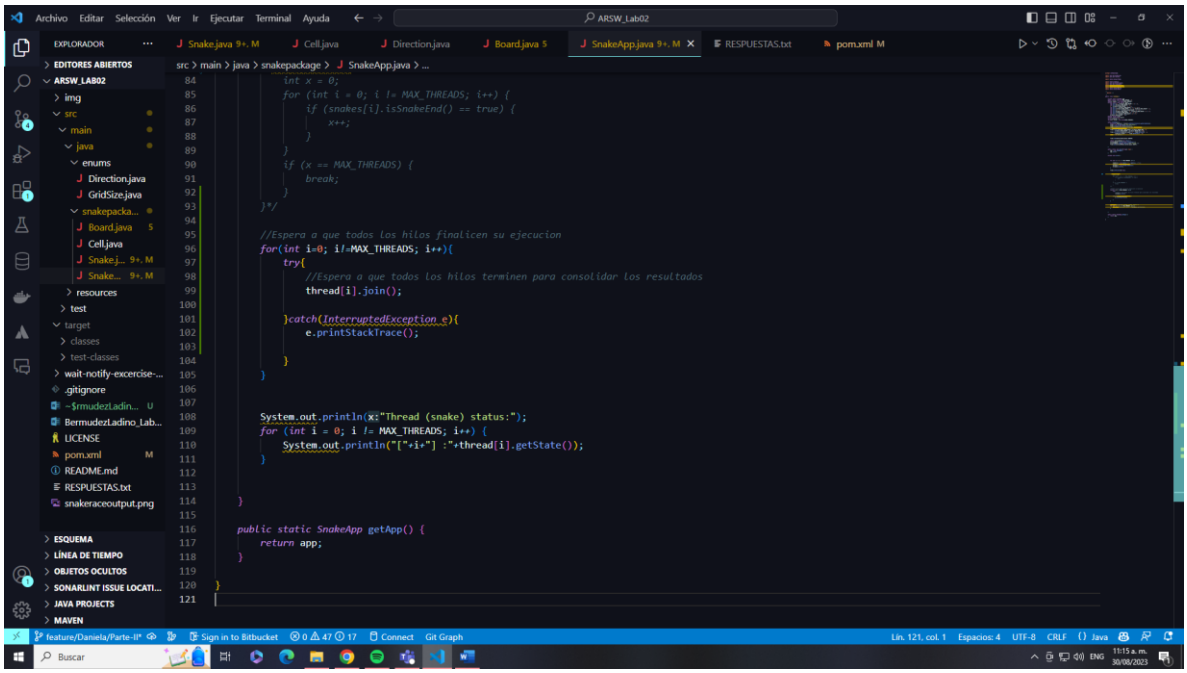
- Modificación de la clase SnakeApp



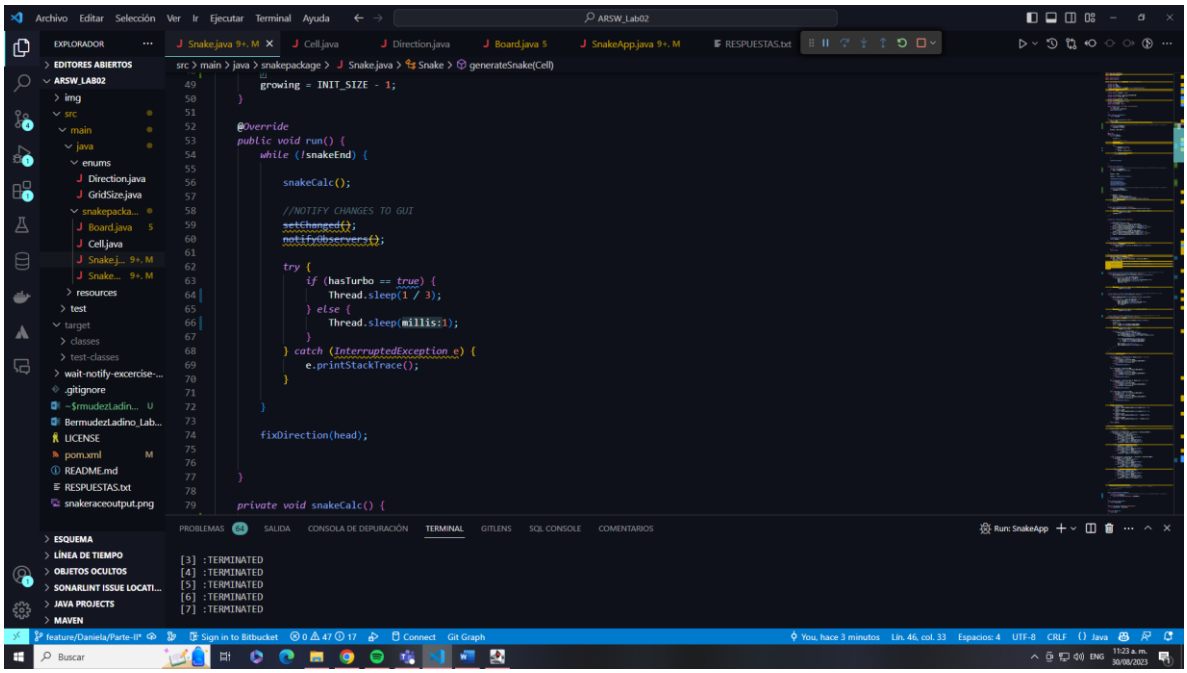
```
public class SnakeApp {

    private static SnakeApp app;
    public static final int MAX_THREADS = 8;
    Snake[] snakes = new Snake[MAX_THREADS];
    private static final Cell[] spawn = {
        new Cell(x:1, (GridSize.GRID_HEIGHT / 2) / 2),
        new Cell(GridSize.GRID_WIDTH - 2,
        3 * (GridSize.GRID_HEIGHT / 2) / 2),
        new Cell(3 * (GridSize.GRID_WIDTH / 2) / 2, y:1),
        new Cell((GridSize.GRID_WIDTH / 2) / 2, GridSize.GRID_HEIGHT - 2),
        new Cell(x:1, 3 * (GridSize.GRID_HEIGHT / 2) / 2),
        new Cell(GridSize.GRID_WIDTH - 2, (GridSize.GRID_HEIGHT / 2) / 2),
        new Cell((GridSize.GRID_WIDTH / 2) / 2, y:1),
        new Cell(3 * (GridSize.GRID_WIDTH / 2) / 2,
        GridSize.GRID_HEIGHT - 2)};
    private JFrame frame;
    private static Board board;
    int nr_selected = 0;
    Thread[] thread = new Thread[MAX_THREADS];

    public SnakeApp() {
        Dimension dimension = Toolkit.getDefaultToolkit().getScreenSize();
        frame = new JFrame(title:"The Snake Race");
        frame.setLayout(new BorderLayout());
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        // frame.setSize(618, 640);
        frame.setSize(GridSize.GRID_WIDTH * GridSize.WIDTH_BOX + 17,
                GridSize.GRID_HEIGHT * GridSize.HEIGH_BOX + 40);
        frame.setLocation(dimension.width / 2 - frame.getWidth() / 2,
                dimension.height / 2 - frame.getHeight() / 2);
        board = new Board();


        frame.add(board,BorderLayout.CENTER);

        JPanel actionsBPabel=new JPanel();
        actionsBPabel.setLayout(new FlowLayout());
        actionsBPabel.add(new JButton(text:"Action "));
```

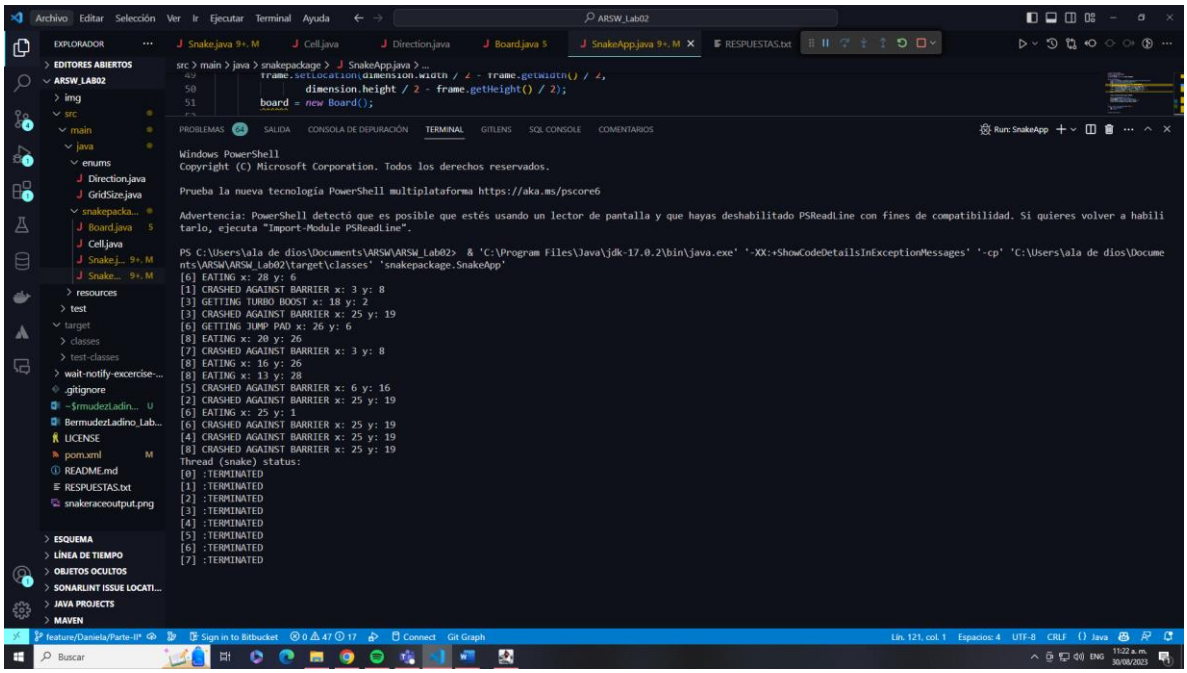


```
        actionsBPabel.add(new JButton(text:"Action "));
        frame.add(actionsBPabel,BorderLayout.SOUTH);

    }

    Run | Debug
    public static void main(String[] args) {
        app = new SnakeApp();
        app.init();
    }

    private void init() {



        for (int i = 0; i != MAX_THREADS; i++) {

            snakes[i] = new Snake(i + 1, spawn[i], i + 1);
            snakes[i].addObserver(board);
            thread[i] = new Thread(snakes[i]);
            thread[i].start();
        }

        frame.setVisible(b:true);


        /*while (true) {
            int x = 0;
            for (int i = 0; i != MAX_THREADS; i++) {
                if (snakes[i].isSnakeEnd() == true) {
                    x++;
                }
            }
            if (x == MAX_THREADS) {
                break;
            }       unknown, hace 4 días • fuentes base
        }*/

        //Espera a que todos los hilos finalicen su ejecucion
        for(int i=0; i!=MAX_THREADS; i++){
```



Resultados ejecución:





Al ejecutar nuevamente la aplicación con la misma velocidad del experimento del punto 2 y luego de haber realizado las respectivas modificaciones mencionadas anteriormente podemos observar que ya no nos sale el error de ConcurrentModificationException.

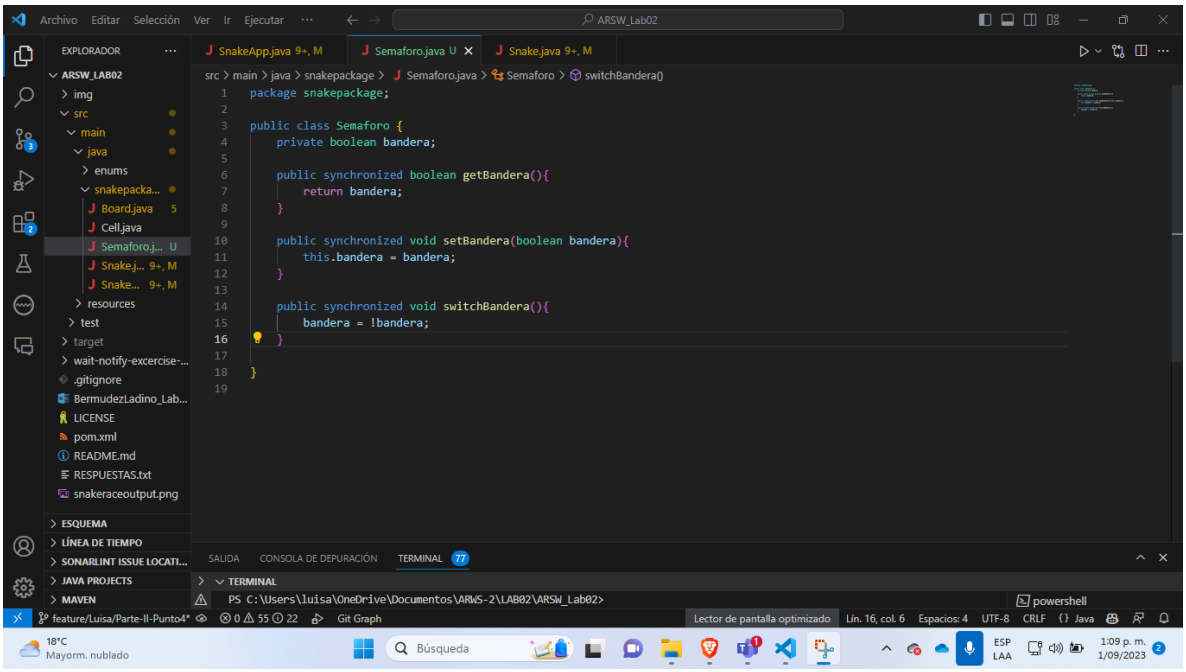
4. Como se puede observar, el juego está incompleto. Haga los ajustes necesarios para que a través de botones en la interfaz se pueda Iniciar/Pausar/Reanudar el juego: iniciar el juego no se ha iniciado aún, suspender el juego si está en ejecución, reactivar el juego si está suspendido. Para esto tenga en cuenta:

   - Al pausar (suspender) el juego, en alguna parte de la interfaz (agregue los componentes que desee) se debe mostrar:

     - La serpiente viva más larga.
     - La peor serpiente: la que primero murió.

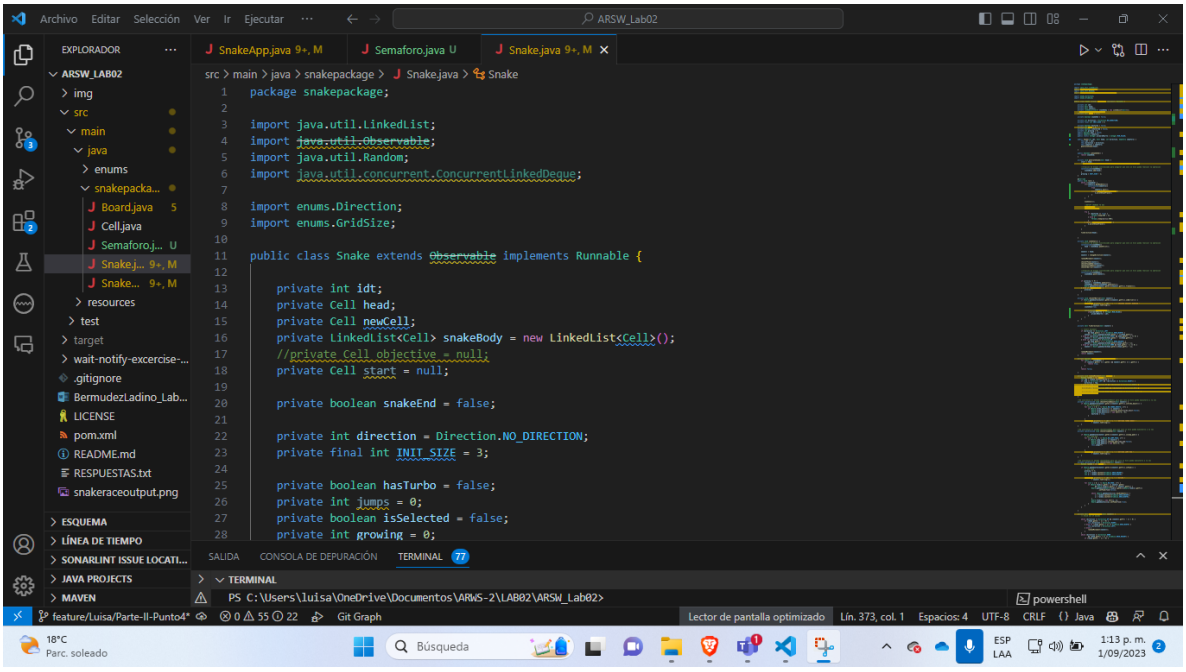
   Recuerde que la suspensión de las serpientes NO es instantánea, y que se debe garantizar que se muestre información consistente.

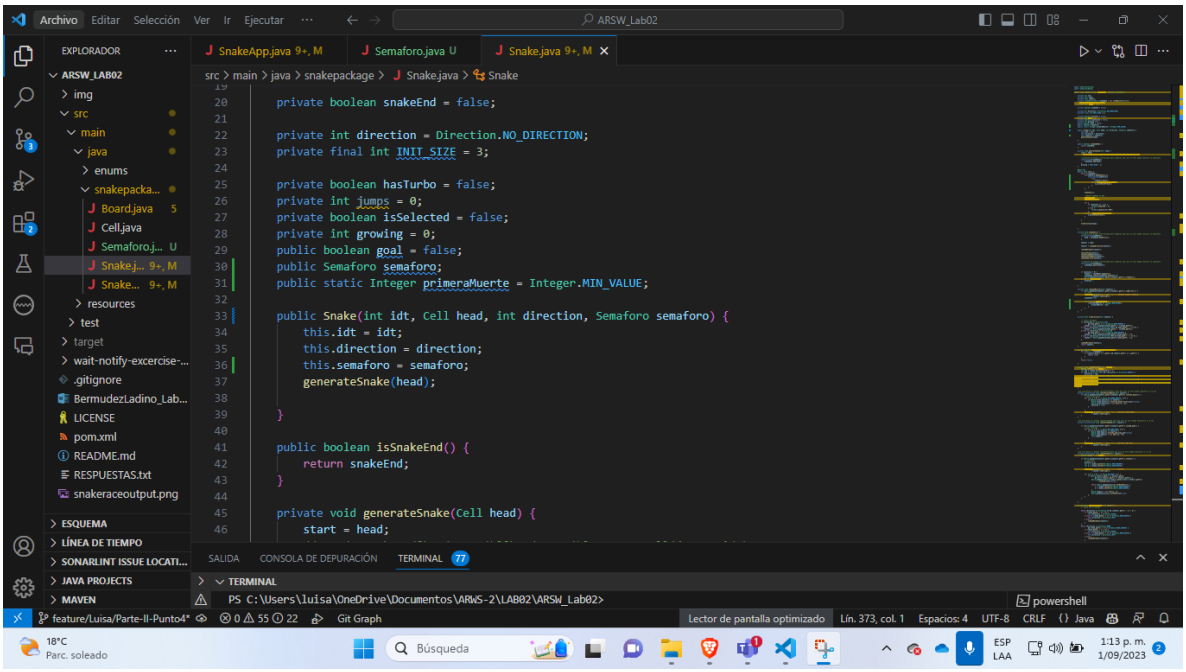
- Creamos la clase Semaforo la cual nos permitirá conocer en qué estado se encuentra el hilo, así mismo nos proporciona una forma de controlar el acceso a ciertos recursos o secciones críticas, en las cuales los hilos podrán consultar o modificar el estado de la bandera para coordinar sus acciones y evitar problemas de concurrencia.

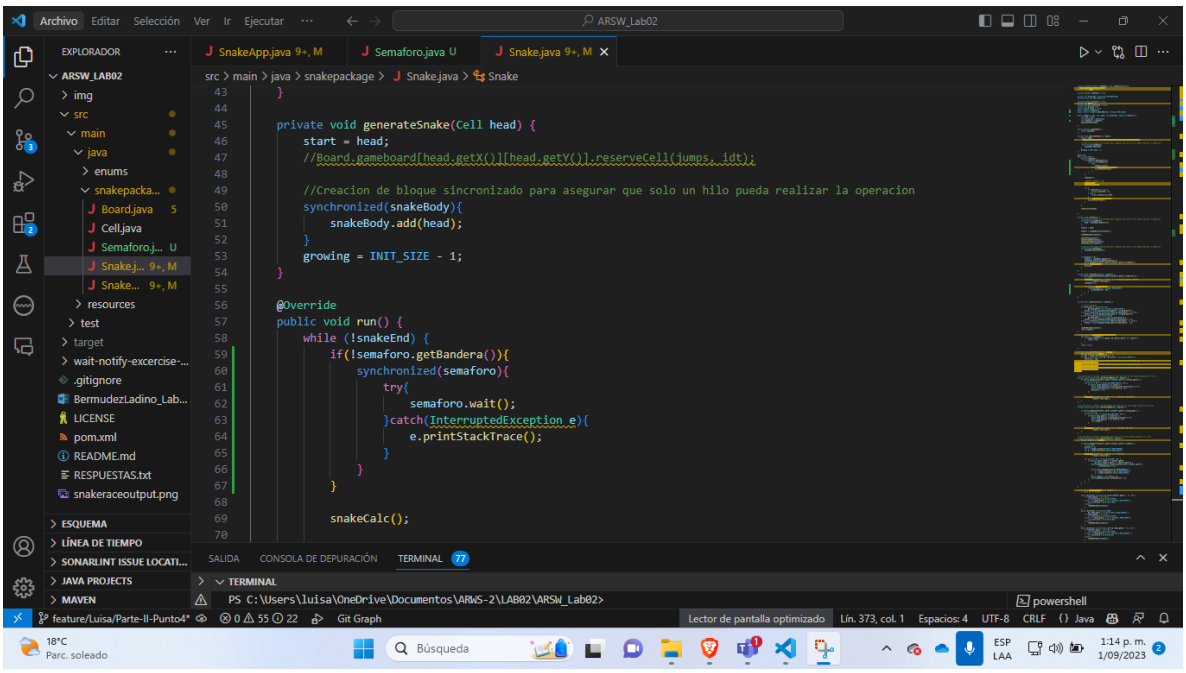


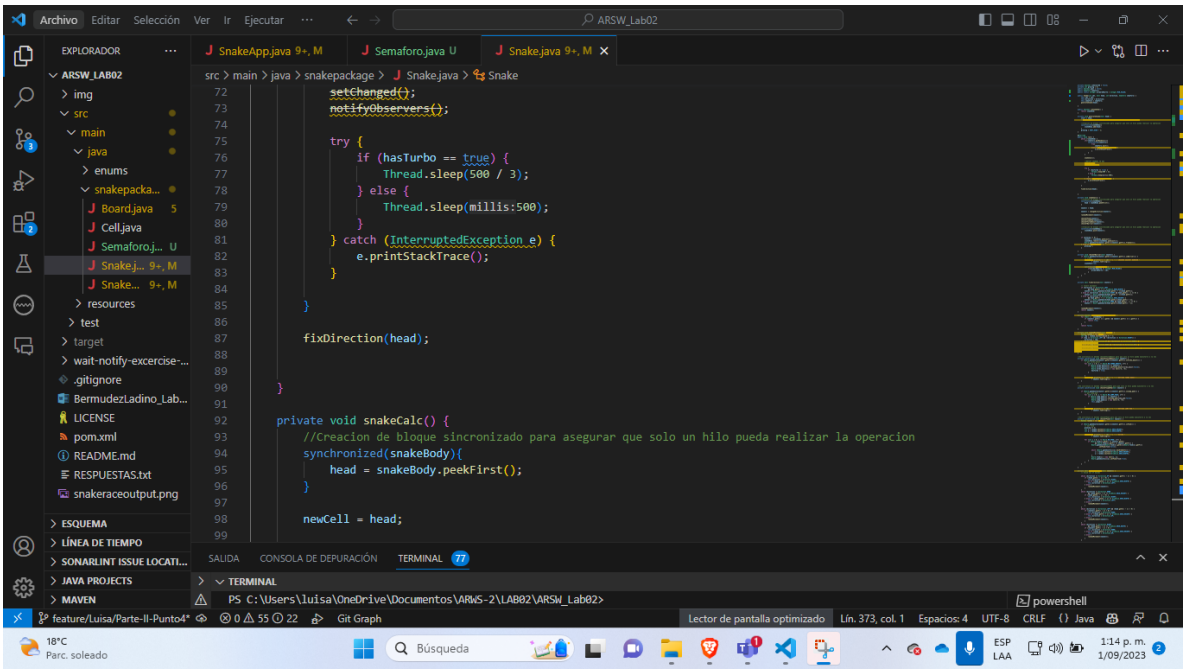
- Hicimos los ajustes necesarios para que a través de botones en la interfaz se pueda Iniciar, Pausar y Reanudar el juego como se indica anteriormente.

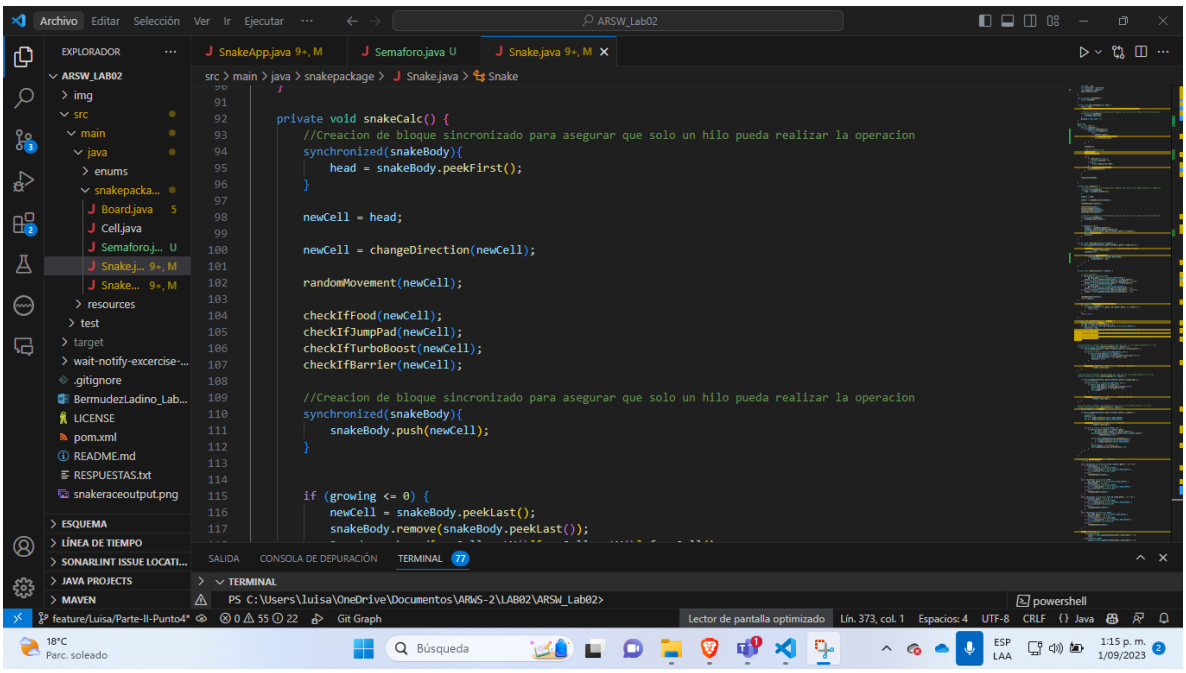
- **Modificación de la clase Snake para mostrar la serpiente que primero murió**
  Vamos a realizar un seguimiento del identificador del primer hilo de la serpiente que muere en el juego. Adicional a esto tenemos un objeto de tipo Semaforo que se utiliza para sincronizar el hilo de la serpiente, para que de esta forma se conozca el estado del semáforo y así saber si el semáforo nos permite avanzar o se encuentra bloqueado y el hilo deba esperar hasta que el semáforo le permita continuar.

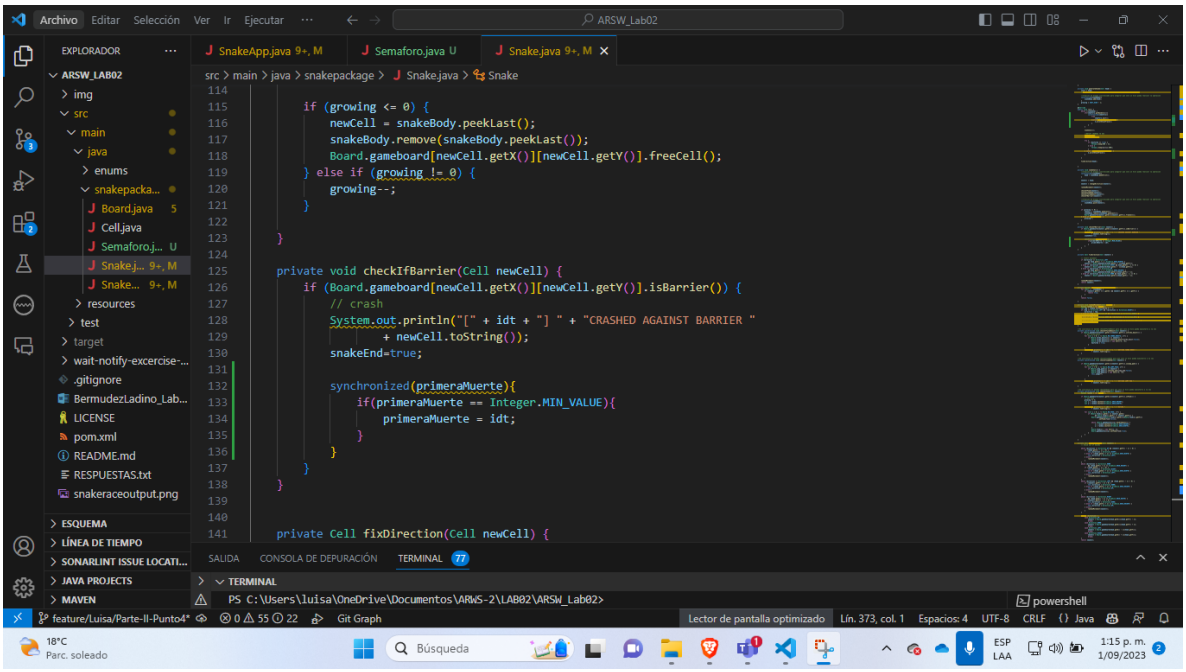


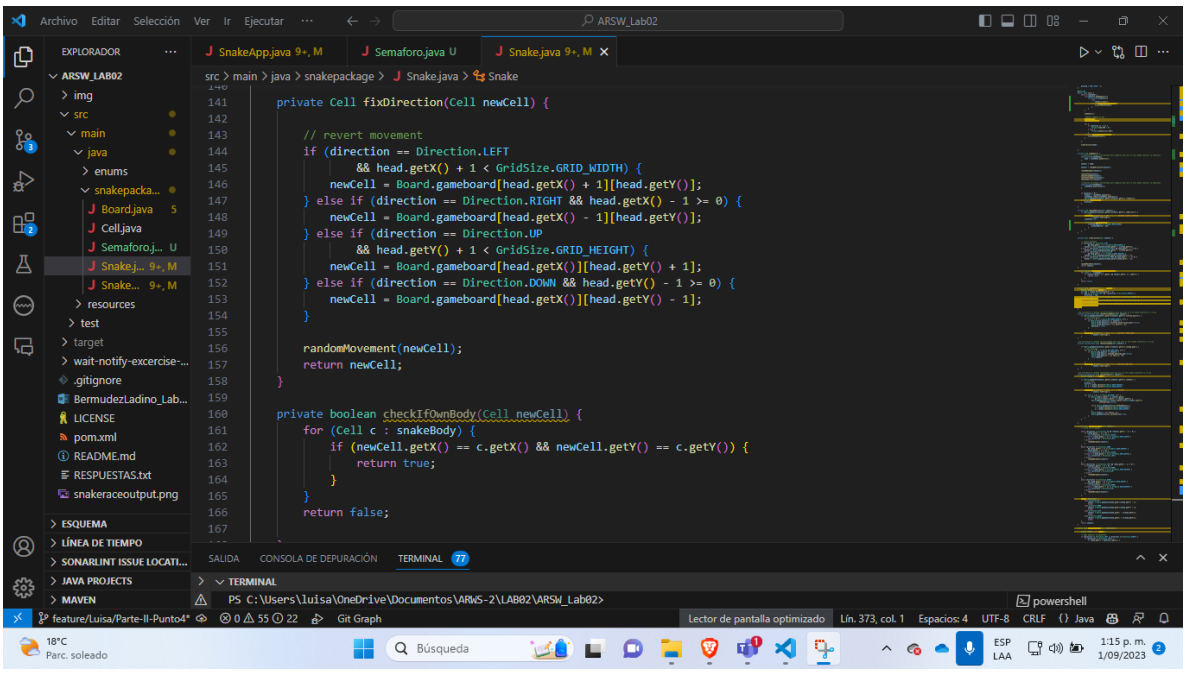

Archivo Editar Selección Ver Ir Ejecutar
ARSW_Lab02
EXPLORADOR
SnakeApp.java 9+, M
Semaforo.java U
Snake.java 9+, M
src > main > java > snakepackage > Snake.java > Snake
private boolean snakeEnd = false;
private int direction = Direction.NO_DIRECTION;
private final int INIT_SIZE = 3;
private boolean hasTurbo = false;
private int jumps = 0;
private boolean isSelected = false;
private int growing = 0;
public boolean goal = false;
public Semaforo semaforo;
public static Integer primeraMuerte = Integer.MIN_VALUE;
public Snake(int idt, Cell head, int direction, Semaforo semaforo) {
this.idt = idt;
this.direction = direction;
this.semaforo = semaforo;
generateSnake(head);
}
public boolean isSnakeEnd() {
return snakeEnd;
}
private void generateSnake(Cell head) {
start = head;
SALIDA
CONSOLA DE DEPURACIÓN
TERMINAL
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB02\ARSW_Lab02>
powershell
feature/Luisa/Parte-II-Punto4*
Git Graph
Lector de pantalla optimizado
Lín. 373, col. 1
Espacios: 4
UTF-8
CRLF
Java
18°C
Parc. soleado
Búsqueda
ESP
LAA
1:13 p. m.
1/09/2023

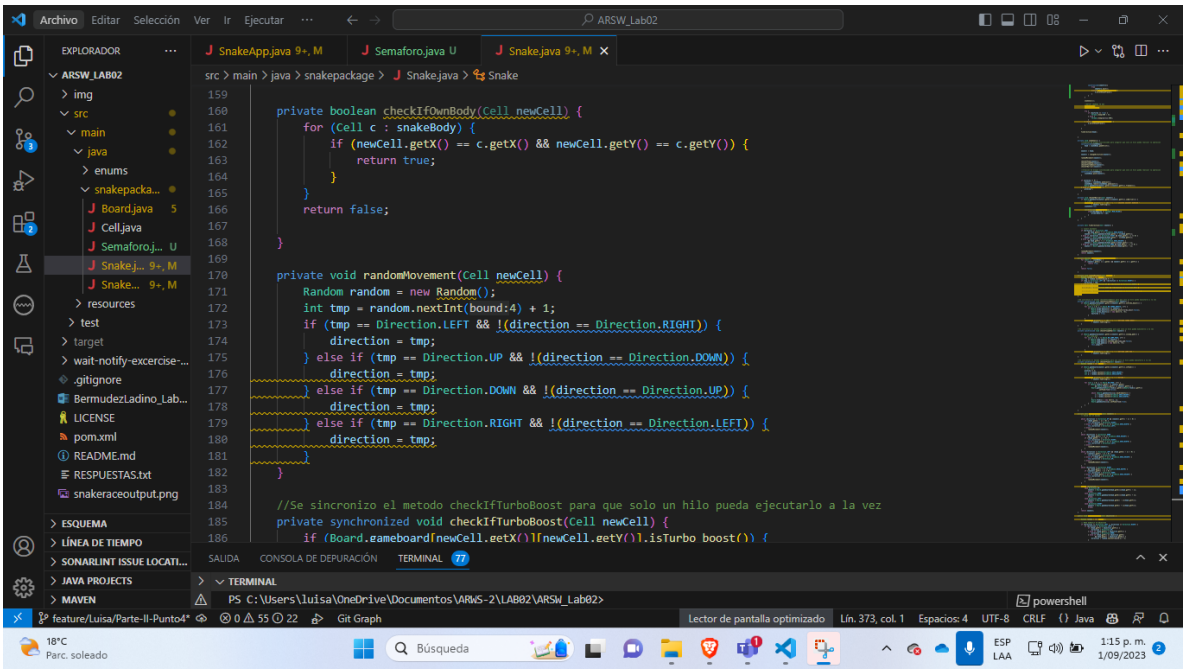

Archivo Editar Selección Ver Ir Ejecutar
ARSW_Lab02
EXPLORADOR
SnakeApp.java 9+, M
Semaforo.java U
Snake.java 9+, M
src > main > java > snakepackage > Snake.java > Snake
}
private void generateSnake(Cell head) {
start = head;
//Board.gameboard[head.getX()][head.getY()].reserveCell(jumps, idt);
//Creacion de bloque sincronizado para asegurar que solo un hilo pueda realizar la operacion
synchronized(snakeBody){
snakeBody.add(head);
}
growing = INIT_SIZE - 1;
}
@Override
public void run() {
while (!snakeEnd) {
if(!semaforo.getBandera()){
synchronized(semaforo){
try{
semaforo.wait();
}catch(InterruptedException e){
e.printStackTrace();
}
}
}
snakeCalc();
SALIDA
CONSOLA DE DEPURACIÓN
TERMINAL
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB02\ARSW_Lab02>
powershell
feature/Luisa/Parte-II-Punto4*
Git Graph
Lector de pantalla optimizado
Lín. 373, col. 1
Espacios: 4
UTF-8
CRLF
Java
18°C
Parc. soleado
Búsqueda
ESP
LAA
1:14 p. m.
1/09/2023

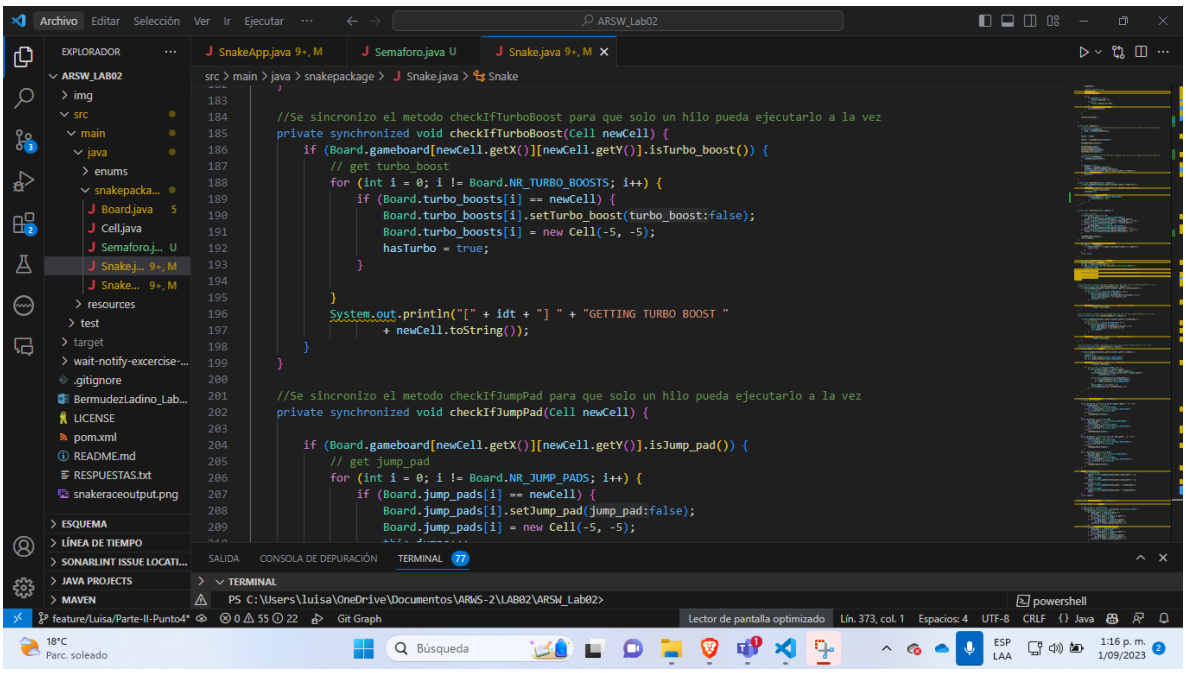

Archivo Editar Selección Ver Ir Ejecutar
ARSW_Lab02
EXPLORADOR
ARSW_LAB02
SnakeApp.java 9+, M
Semaforo.java U
Snake.java 9+, M
src > main > java > snakepackage > Snake.java > Snake
setChanged();
notifyObservers();
try {
if (hasTurbo == true) {
Thread.sleep(500 / 3);
} else {
Thread.sleep(millis:500);
}
} catch (InterruptedException e) {
e.printStackTrace();
}
}
fixDirection(head);
}
private void snakeCalc() {
//Creacion de bloque sincronizado para asegurar que solo un hilo pueda realizar la operacion
synchronized(snakeBody){
head = snakeBody.peekFirst();
}
newCell = head;
SALIDA
CONSOLA DE DEPURACIÓN
TERMINAL
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB02\ARSW_Lab02>
powershell
feature/Luisa/Parte-II-Punto4*
Git Graph
Lector de pantalla optimizado
Lín. 373, col. 1
Espacios: 4
UTF-8
CRLF
Java
18°C
Parc. soleado
Búsqueda
ESP
LAA
1:14 p. m.
1/09/2023

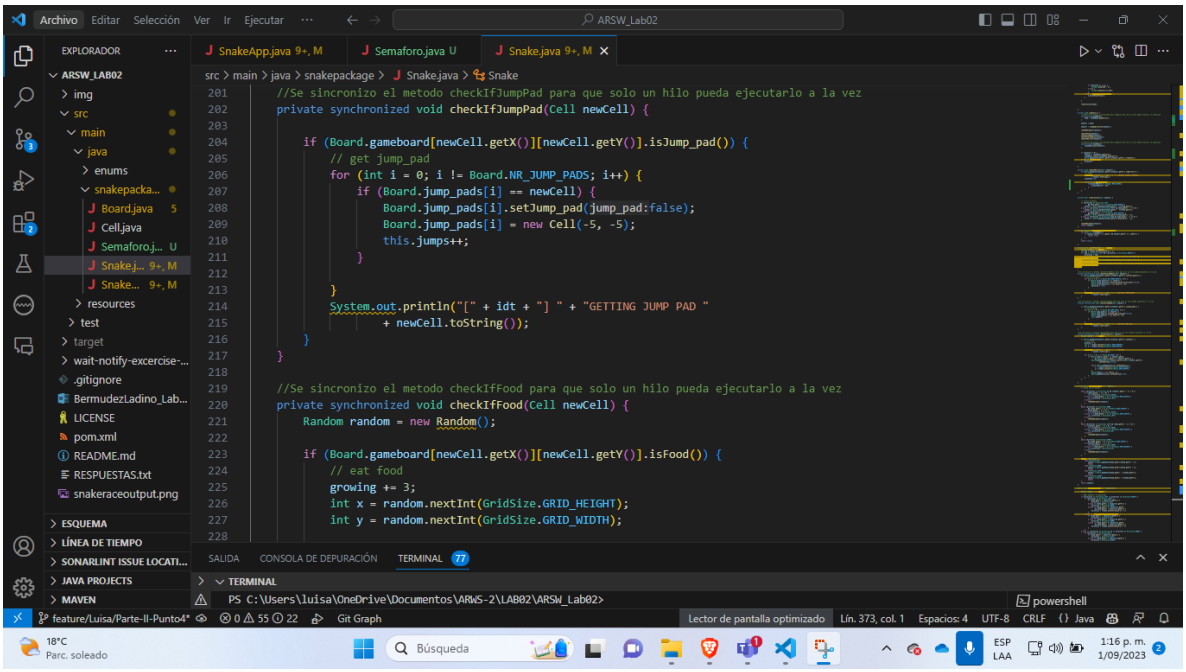

Archivo Editar Selección Ver Ir Ejecutar
ARSW_Lab02
EXPLORADOR
ARSW_LAB02
SnakeApp.java 9+, M
Semaforo.java U
Snake.java 9+, M
src > main > java > snakepackage > Snake.java > Snake
private void snakeCalc() {
//Creacion de bloque sincronizado para asegurar que solo un hilo pueda realizar la operacion
synchronized(snakeBody){
head = snakeBody.peekFirst();
}
newCell = head;
newCell = changeDirection(newCell);
randomMovement(newCell);
checkIfFood(newCell);
checkIfJumpPad(newCell);
checkIfTurboBoost(newCell);
checkIfBarrier(newCell);
//Creacion de bloque sincronizado para asegurar que solo un hilo pueda realizar la operacion
synchronized(snakeBody){
snakeBody.push(newCell);
}
if (growing <= 0) {
newCell = snakeBody.peekLast();
snakeBody.remove(snakeBody.peekLast());
SALIDA
CONSOLA DE DEPURACIÓN
TERMINAL
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB02\ARSW_Lab02>
powershell
feature/Luisa/Parte-II-Punto4*
Git Graph
Lector de pantalla optimizado
Lín. 373, col. 1
Espacios: 4
UTF-8
CRLF
Java
18°C
Parc. soleado
Búsqueda
ESP
LAA
1:15 p. m.
1/09/2023

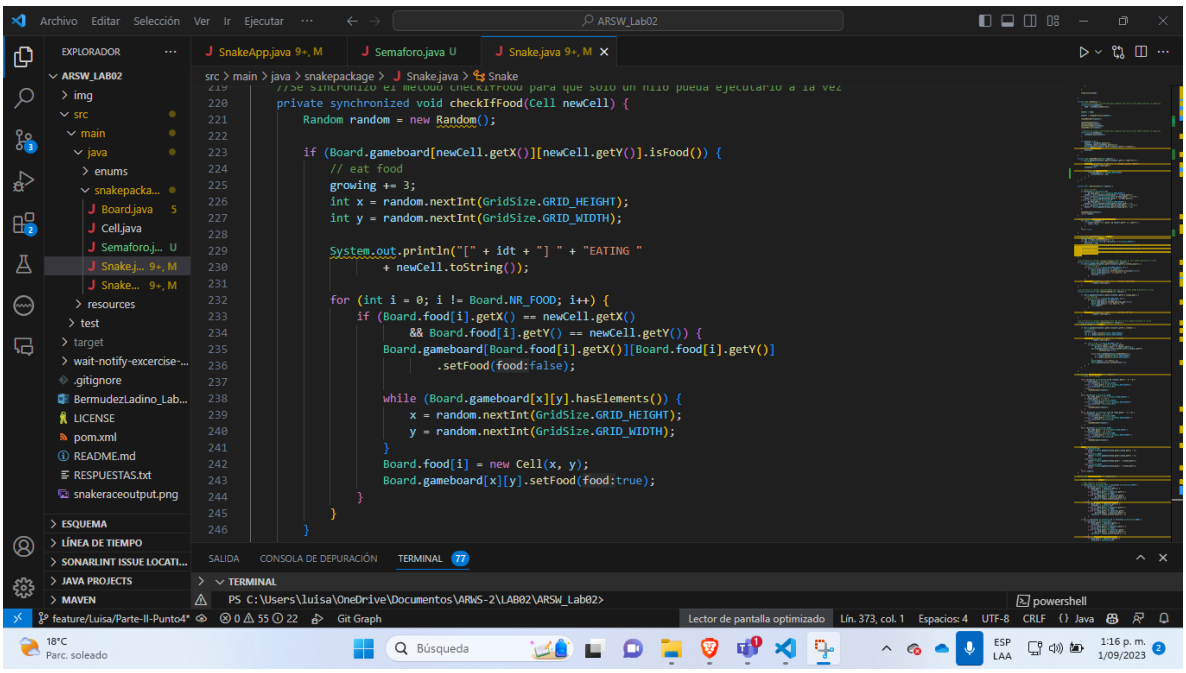

Archivo Editar Selección Ver Ir Ejecutar
ARSW_Lab02
EXPLORADOR
ARSW_LAB02
SnakeApp.java 9+, M
Semaforo.java U
Snake.java 9+, M
src > main > java > snakepackage > Snake.java > Snake
if (growing <= 0) {
newCell = snakeBody.peekLast();
snakeBody.remove(snakeBody.peekLast());
Board.gameboard[newCell.getX()][newCell.getY()].freeCell();
} else if (growing != 0) {
growing--;
}
}
private void checkIfBarrier(Cell newCell) {
if (Board.gameboard[newCell.getX()][newCell.getY()].isBarrier()) {
// crash
System.out.println("[" + idt + "] " + "CRASHED AGAINST BARRIER "
+ newCell.toString());
snakeEnd=true;
synchronized(primeraMuerte){
if(primeraMuerte == Integer.MIN_VALUE){
primeraMuerte = idt;
}
}
}
}
private Cell fixDirection(Cell newCell) {
SALIDA
CONSOLA DE DEPURACIÓN
TERMINAL 77
TERMINAL
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB02\ARSW_Lab02>
powershell
feature/Luisa/Parte-II-Punto4*
Git Graph
Lector de pantalla optimizado
Lín. 373, col. 1
Espacios: 4
UTF-8
CRLF
Java
18°C
Parc. soleado
Búsqueda
ESP
LAA
1:15 p. m.
1/09/2023

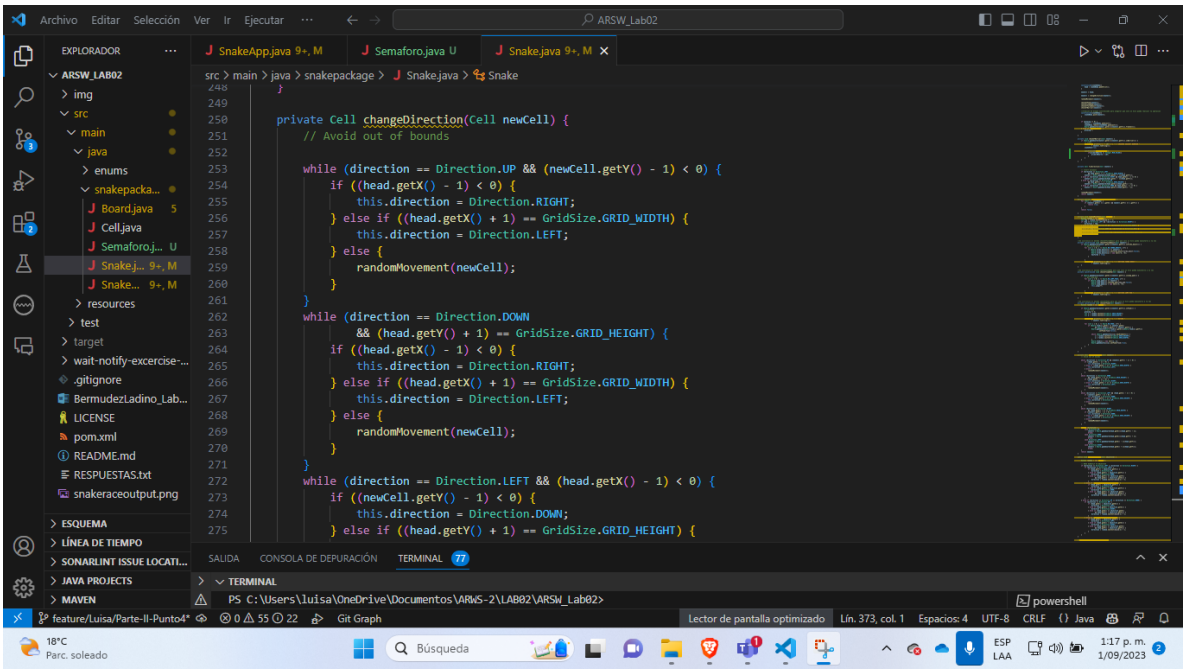

Archivo Editar Selección Ver Ir Ejecutar
ARSW_Lab02
EXPLORADOR
ARSW_LAB02
SnakeApp.java 9+, M
Semaforo.java U
Snake.java 9+, M
src > main > java > snakepackage > Snake.java > Snake
private Cell fixDirection(Cell newCell) {
// revert movement
if (direction == Direction.LEFT
&& head.getX() + 1 < GridSize.GRID_WIDTH) {
newCell = Board.gameboard[head.getX() + 1][head.getY()];
} else if (direction == Direction.RIGHT && head.getX() - 1 >= 0) {
newCell = Board.gameboard[head.getX() - 1][head.getY()];
} else if (direction == Direction.UP
&& head.getY() + 1 < GridSize.GRID_HEIGHT) {
newCell = Board.gameboard[head.getX()][head.getY() + 1];
} else if (direction == Direction.DOWN && head.getY() - 1 >= 0) {
newCell = Board.gameboard[head.getX()][head.getY() - 1];
}
randomMovement(newCell);
return newCell;
}
private boolean checkIfOwnBody(Cell newCell) {
for (Cell c : snakeBody) {
if (newCell.getX() == c.getX() && newCell.getY() == c.getY()) {
return true;
}
}
return false;
SALIDA
CONSOLA DE DEPURACIÓN
TERMINAL 77
TERMINAL
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB02\ARSW_Lab02>
powershell
feature/Luisa/Parte-II-Punto4*
Git Graph
Lector de pantalla optimizado
Lín. 373, col. 1
Espacios: 4
UTF-8
CRLF
Java
18°C
Parc. soleado
Búsqueda
ESP
LAA
1:15 p. m.
1/09/2023

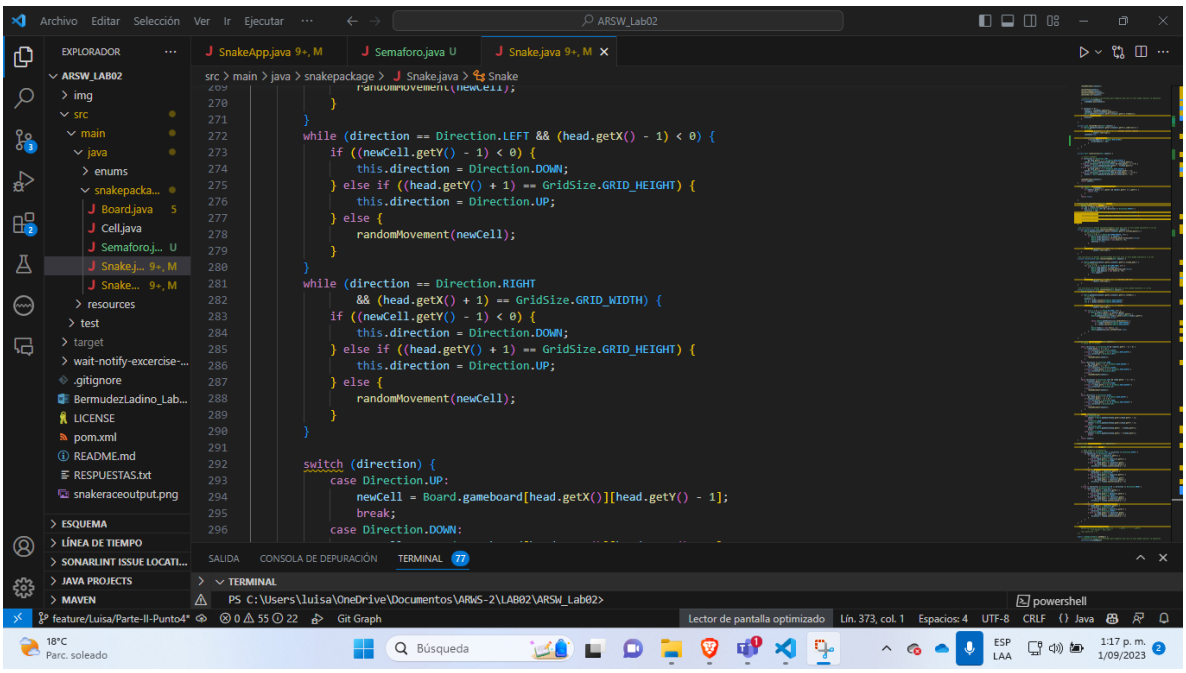

Archivo Editar Selección Ver Ir Ejecutar
ARSW_Lab02
EXPLORADOR
SnakeApp.java 9+, M
Semaforo.java U
Snake.java 9+, M
src > main > java > snakepackage > Snake.java > Snake
ARSW_LAB02
img
src
main
java
enums
snakepacka...
Board.java 5
Cell.java
Semaforo.j... U
Snake.j... 9+, M
Snake... 9+, M
resources
test
target
wait-notify-excercise-...
.gitignore
BermudezLadino_Lab...
LICENSE
pom.xml
README.md
RESPUESTAS.txt
snakeraceoutput.png
ESQUEMA
LÍNEA DE TIEMPO
SONARLINT ISSUE LOCATI...
JAVA PROJECTS
MAVEN
159 private boolean checkIfOwnBody(Cell newCell) {
160 for (Cell c : snakeBody) {
161 if (newCell.getX() == c.getX() && newCell.getY() == c.getY()) {
162 return true;
163 }
164 }
165 return false;
166 }
167
168 private void randomMovement(Cell newCell) {
169 Random random = new Random();
170 int tmp = random.nextInt(bound:4) + 1;
171 if (tmp == Direction.LEFT && !(direction == Direction.RIGHT)) {
172 direction = tmp;
173 } else if (tmp == Direction.UP && !(direction == Direction.DOWN)) {
174 direction = tmp;
175 } else if (tmp == Direction.DOWN && !(direction == Direction.UP)) {
176 direction = tmp;
177 } else if (tmp == Direction.RIGHT && !(direction == Direction.LEFT)) {
178 direction = tmp;
179 }
180 }
181
182 //Se sincronizo el metodo checkIfTurboBoost para que solo un hilo pueda ejecutarlo a la vez
183 private synchronized void checkIfTurboBoost(Cell newCell) {
184 if (Board.gameboard[newCell.getX()][newCell.getY()].isTurbo_boost()) {
SALIDA CONSOLA DE DEPURACIÓN TERMINAL 77
TERMINAL
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB02\ARSW_Lab02>
powershell
feature/Luisa/Parte-II-Punto4* 0 55 22 Git Graph
Lector de pantalla optimizado Lín. 373, col. 1 Espacios: 4 UTF-8 CRLF Java
18°C Parc. soleado
Búsqueda
ESP LAA
1:15 p. m. 1/09/2023

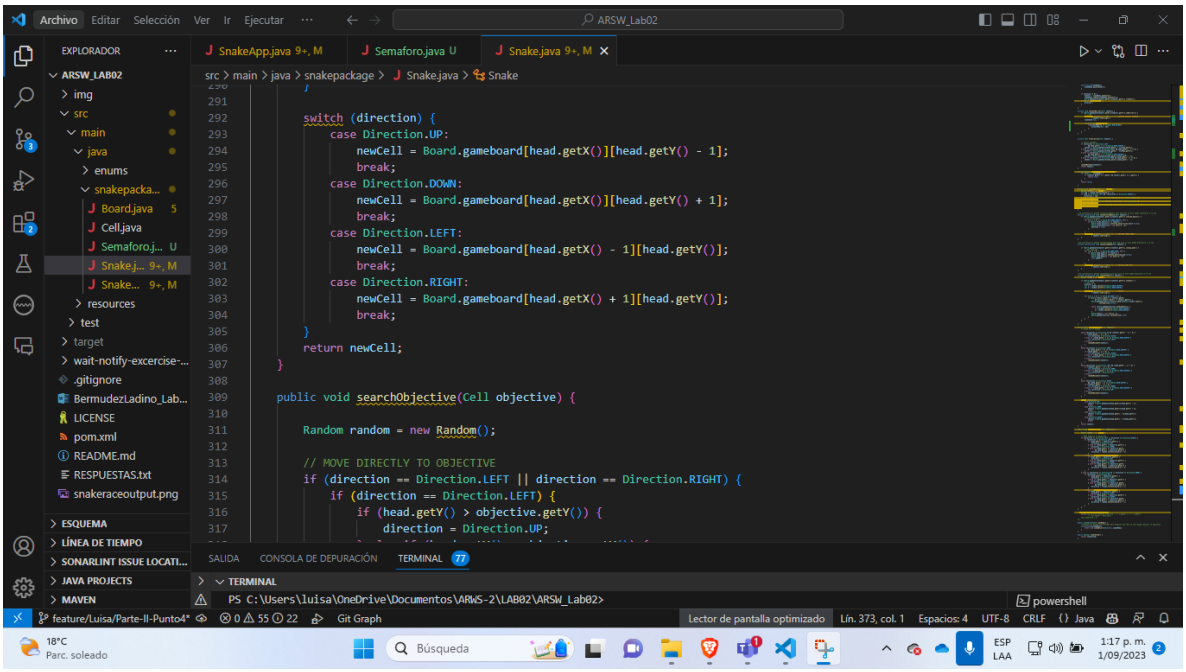

Archivo Editar Selección Ver Ir Ejecutar
ARSW_Lab02
EXPLORADOR
SnakeApp.java 9+, M
Semaforo.java U
Snake.java 9+, M
src > main > java > snakepackage > Snake.java > Snake
ARSW_LAB02
img
src
main
java
enums
snakepacka...
Board.java 5
Cell.java
Semaforo.j... U
Snake.j... 9+, M
Snake... 9+, M
resources
test
target
wait-notify-excercise-...
.gitignore
BermudezLadino_Lab...
LICENSE
pom.xml
README.md
RESPUESTAS.txt
snakeraceoutput.png
ESQUEMA
LÍNEA DE TIEMPO
SONARLINT ISSUE LOCATI...
JAVA PROJECTS
MAVEN
183
184 //Se sincronizo el metodo checkIfTurboBoost para que solo un hilo pueda ejecutarlo a la vez
185 private synchronized void checkIfTurboBoost(Cell newCell) {
186 if (Board.gameboard[newCell.getX()][newCell.getY()].isTurbo_boost()) {
187 // get turbo_boost
188 for (int i = 0; i != Board.NR_TURBO_BOOSTS; i++) {
189 if (Board.turbo_boosts[i] == newCell) {
190 Board.turbo_boosts[i].setTurbo_boost(turbo_boost:false);
191 Board.turbo_boosts[i] = new Cell(-5, -5);
192 hasTurbo = true;
193 }
194
195 }
196 System.out.println("[" + idt + "] " + "GETTING TURBO BOOST "
197 + newCell.toString());
198 }
199 }
200
201 //Se sincronizo el metodo checkIfJumpPad para que solo un hilo pueda ejecutarlo a la vez
202 private synchronized void checkIfJumpPad(Cell newCell) {
203
204 if (Board.gameboard[newCell.getX()][newCell.getY()].isJump_pad()) {
205 // get jump_pad
206 for (int i = 0; i != Board.NR_JUMP_PADS; i++) {
207 if (Board.jump_pads[i] == newCell) {
208 Board.jump_pads[i].setJump_pad(jump_pad:false);
209 Board.jump_pads[i] = new Cell(-5, -5);
SALIDA CONSOLA DE DEPURACIÓN TERMINAL 77
TERMINAL
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB02\ARSW_Lab02>
powershell
feature/Luisa/Parte-II-Punto4* 0 55 22 Git Graph
Lector de pantalla optimizado Lín. 373, col. 1 Espacios: 4 UTF-8 CRLF Java
18°C Parc. soleado
Búsqueda
ESP LAA
1:16 p. m. 1/09/2023

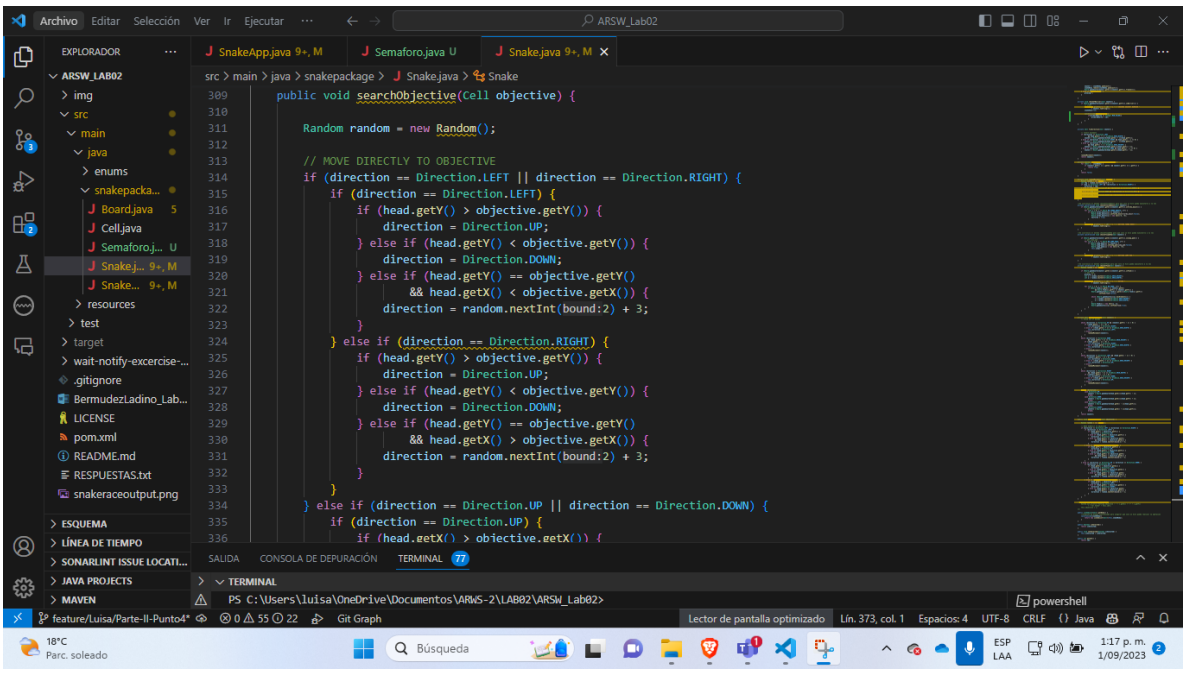

Archivo Editar Selección Ver Ir Ejecutar
ARSW_Lab02
EXPLORADOR
SnakeApp.java 9+, M
Semaforo.java U
Snake.java 9+, M
src > main > java > snakepackage > Snake.java > Snake
//Se sincronizo el metodo checkIfJumpPad para que solo un hilo pueda ejecutarlo a la vez
private synchronized void checkIfJumpPad(Cell newCell) {
if (Board.gameboard[newCell.getX()][newCell.getY()].isJump_pad()) {
// get jump_pad
for (int i = 0; i != Board.NR_JUMP_PADS; i++) {
if (Board.jump_pads[i] == newCell) {
Board.jump_pads[i].setJump_pad(jump_pad:false);
Board.jump_pads[i] = new Cell(-5, -5);
this.jumps++;
}
}
System.out.println("[" + idt + "] " + "GETTING JUMP PAD "
+ newCell.toString());
}
}
//Se sincronizo el metodo checkIfFood para que solo un hilo pueda ejecutarlo a la vez
private synchronized void checkIfFood(Cell newCell) {
Random random = new Random();
if (Board.gameboard[newCell.getX()][newCell.getY()].isFood()) {
// eat food
growing += 3;
int x = random.nextInt(GridSize.GRID_HEIGHT);
int y = random.nextInt(GridSize.GRID_WIDTH);
SALIDA CONSOLA DE DEPURACIÓN TERMINAL 77
TERMINAL
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB02\ARSW_Lab02>
powershell
feature/Luisa/Parte-II-Punto4*
Git Graph
Lector de pantalla optimizado
Lín. 373, col. 1
Espacios: 4
UTF-8
CRLF
Java
18°C
Parc. soleado
Búsqueda
ESP
LAA
1:16 p. m.
1/09/2023

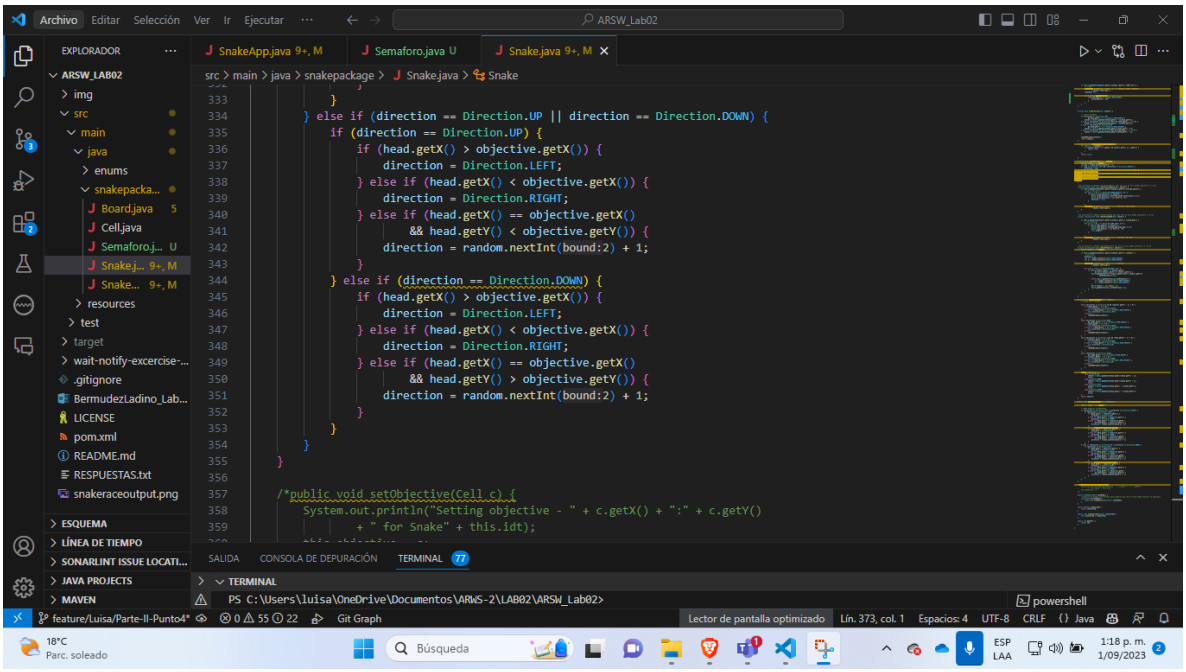

Archivo Editar Selección Ver Ir Ejecutar
ARSW_Lab02
EXPLORADOR
SnakeApp.java 9+, M
Semaforo.java U
Snake.java 9+, M
src > main > java > snakepackage > Snake.java > Snake
private synchronized void checkIfFood(Cell newCell) {
Random random = new Random();
if (Board.gameboard[newCell.getX()][newCell.getY()].isFood()) {
// eat food
growing += 3;
int x = random.nextInt(GridSize.GRID_HEIGHT);
int y = random.nextInt(GridSize.GRID_WIDTH);
System.out.println("[" + idt + "] " + "EATING "
+ newCell.toString());
for (int i = 0; i != Board.NR_FOOD; i++) {
if (Board.food[i].getX() == newCell.getX()
&& Board.food[i].getY() == newCell.getY()) {
Board.gameboard[Board.food[i].getX()][Board.food[i].getY()]
.setFood(food:false);
while (Board.gameboard[x][y].hasElements()) {
x = random.nextInt(GridSize.GRID_HEIGHT);
y = random.nextInt(GridSize.GRID_WIDTH);
}
Board.food[i] = new Cell(x, y);
Board.gameboard[x][y].setFood(food:true);
}
}
}
SALIDA CONSOLA DE DEPURACIÓN TERMINAL 77
TERMINAL
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB02\ARSW_Lab02>
powershell
feature/Luisa/Parte-II-Punto4*
Git Graph
Lector de pantalla optimizado
Lín. 373, col. 1
Espacios: 4
UTF-8
CRLF
Java
18°C
Parc. soleado
Búsqueda
ESP
LAA
1:16 p. m.
1/09/2023

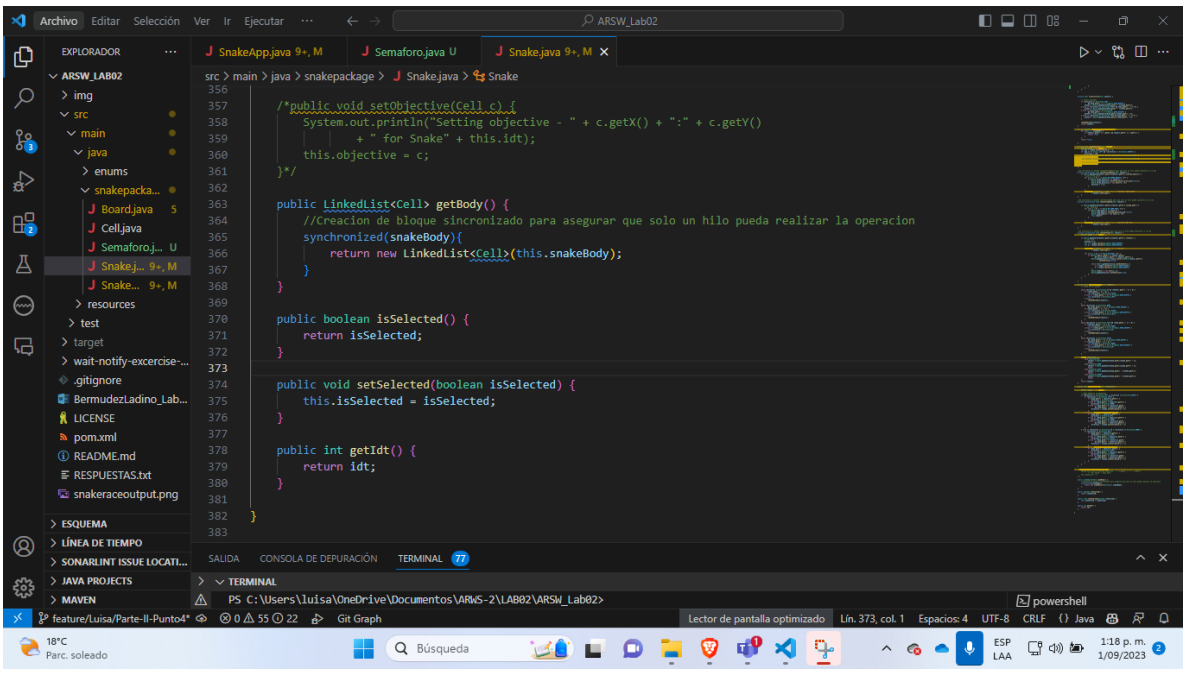

Archivo Editar Selección Ver Ir Ejecutar
ARSW_Lab02
EXPLORADOR
SnakeApp.java 9+, M
Semaforo.java U
Snake.java 9+, M
src > main > java > snakepackage > Snake.java > Snake
ARSW_LAB02
img
src
main
java
enums
snakepacka...
Board.java 5
Cell.java
Semaforo.j... U
Snake.j... 9+, M
Snake... 9+, M
resources
test
target
wait-notify-excercise-...
.gitignore
BermudezLadino_Lab...
LICENSE
pom.xml
README.md
RESPUESTAS.txt
snakeraceoutput.png
ESQUEMA
LÍNEA DE TIEMPO
SONARLINT ISSUE LOCATI...
JAVA PROJECTS
MAVEN
SALIDA
CONSOLA DE DEPURACIÓN
TERMINAL 77
TERMINAL
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB02\ARSW_Lab02>
powershell
feature/Luisa/Parte-II-Punto4*
Git Graph
Lector de pantalla optimizado
Lín. 373, col. 1
Espacios: 4
UTF-8
CRLF
Java
18°C
Parc. soleado
Búsqueda
ESP
LAA
1:17 p. m.
1/09/2023


Archivo Editar Selección Ver Ir Ejecutar
ARSW_Lab02
EXPLORADOR
SnakeApp.java 9+, M
Semaforo.java U
Snake.java 9+, M
src > main > java > snakepackage > Snake.java > Snake
ARSW_LAB02
img
src
main
java
enums
snakepacka...
Board.java 5
Cell.java
Semaforo.j... U
Snake.j... 9+, M
Snake... 9+, M
resources
test
target
wait-notify-excercise-...
.gitignore
BermudezLadino_Lab...
LICENSE
pom.xml
README.md
RESPUESTAS.txt
snakeraceoutput.png
ESQUEMA
LÍNEA DE TIEMPO
SONARLINT ISSUE LOCATI...
JAVA PROJECTS
MAVEN
SALIDA
CONSOLA DE DEPURACIÓN
TERMINAL 77
TERMINAL
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB02\ARSW_Lab02>
powershell
feature/Luisa/Parte-II-Punto4*
Git Graph
Lector de pantalla optimizado
Lín. 373, col. 1
Espacios: 4
UTF-8
CRLF
Java
18°C
Parc. soleado
Búsqueda
ESP
LAA
1:17 p. m.
1/09/2023

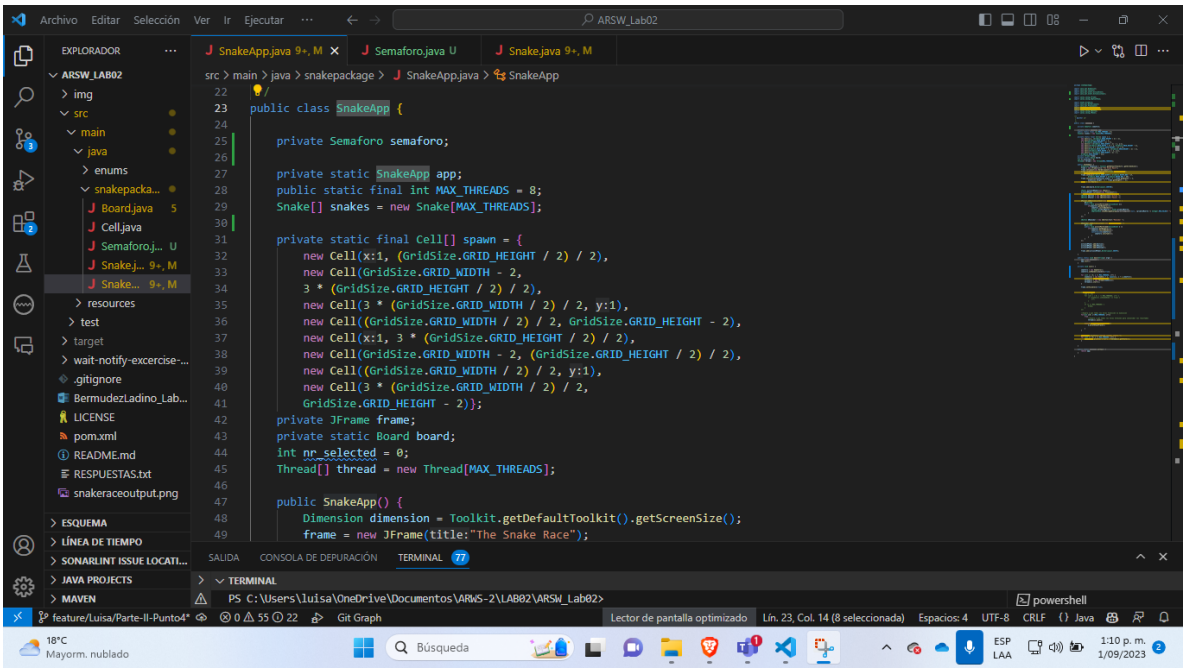

Archivo Editar Selección Ver Ir Ejecutar
ARSW_Lab02
EXPLORADOR
SnakeApp.java 9+, M
Semaforo.java U
Snake.java 9+, M
src > main > java > snakepackage > Snake.java > Snake
ARSW_LAB02
img
src
main
java
enums
snakepacka...
Board.java 5
Cell.java
Semaforo.j... U
Snake.j... 9+, M
Snake... 9+, M
resources
test
target
wait-notify-excercise-...
.gitignore
BermudezLadino_Lab...
LICENSE
pom.xml
README.md
RESPUESTAS.txt
snakeraceoutput.png
ESQUEMA
LÍNEA DE TIEMPO
SONARLINT ISSUE LOCATI...
JAVA PROJECTS
MAVEN
switch (direction) {
case Direction.UP:
newCell = Board.gameboard[head.getX()][head.getY() - 1];
break;
case Direction.DOWN:
newCell = Board.gameboard[head.getX()][head.getY() + 1];
break;
case Direction.LEFT:
newCell = Board.gameboard[head.getX() - 1][head.getY()];
break;
case Direction.RIGHT:
newCell = Board.gameboard[head.getX() + 1][head.getY()];
break;
}
return newCell;
}
public void searchObjective(Cell objective) {
Random random = new Random();
// MOVE DIRECTLY TO OBJECTIVE
if (direction == Direction.LEFT || direction == Direction.RIGHT) {
if (direction == Direction.LEFT) {
if (head.getY() > objective.getY()) {
direction = Direction.UP;
SALIDA
CONSOLA DE DEPURACIÓN
TERMINAL 77
TERMINAL
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB02\ARSW_Lab02>
powershell
feature/Luisa/Parte-II-Punto4*
Git Graph
Lector de pantalla optimizado
Lín. 373, col. 1
Espacios: 4
UTF-8
CRLF
Java
18°C
Parc. soleado
Búsqueda
ESP
LAA
1:17 p. m.
1/09/2023

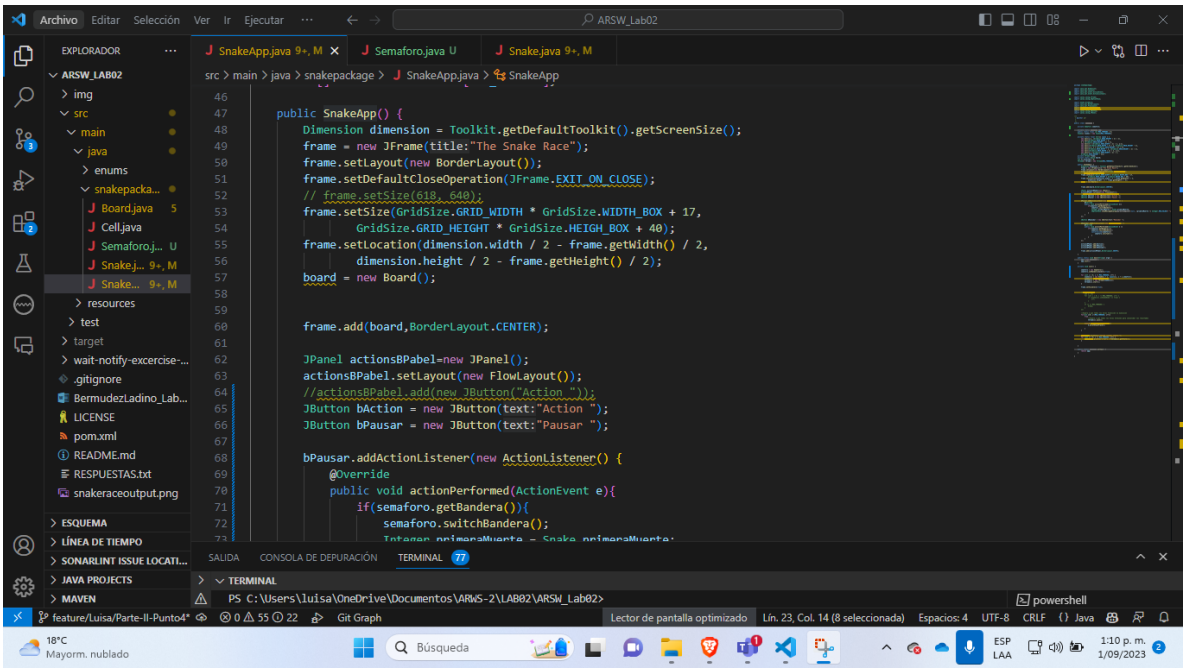

Archivo Editar Selección Ver Ir Ejecutar
ARSW_Lab02
EXPLORADOR
SnakeApp.java 9+, M
Semaforo.java U
Snake.java 9+, M
src > main > java > snakepackage > Snake.java > Snake
ARSW_LAB02
public void searchObjective(Cell objective) {
Random random = new Random();
// MOVE DIRECTLY TO OBJECTIVE
if (direction == Direction.LEFT || direction == Direction.RIGHT) {
if (direction == Direction.LEFT) {
if (head.getY() > objective.getY()) {
direction = Direction.UP;
} else if (head.getY() < objective.getY()) {
direction = Direction.DOWN;
} else if (head.getY() == objective.getY()
&& head.getX() < objective.getX()) {
direction = random.nextInt(bound:2) + 3;
}
} else if (direction == Direction.RIGHT) {
if (head.getY() > objective.getY()) {
direction = Direction.UP;
} else if (head.getY() < objective.getY()) {
direction = Direction.DOWN;
} else if (head.getY() == objective.getY()
&& head.getX() > objective.getX()) {
direction = random.nextInt(bound:2) + 3;
}
}
} else if (direction == Direction.UP || direction == Direction.DOWN) {
if (direction == Direction.UP) {
if (head.getX() > objective.getX()) {
SALIDA
CONSOLA DE DEPURACIÓN
TERMINAL 77
TERMINAL
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB02\ARSW_Lab02>
powershell
feature/Luisa/Parte-II-Punto4*
Git Graph
Lector de pantalla optimizado
Lín. 373, col. 1
Espacios: 4
UTF-8
CRLF
Java
18°C
Parc. soleado
Búsqueda
ESP
LAA
1:17 p. m.
1/09/2023

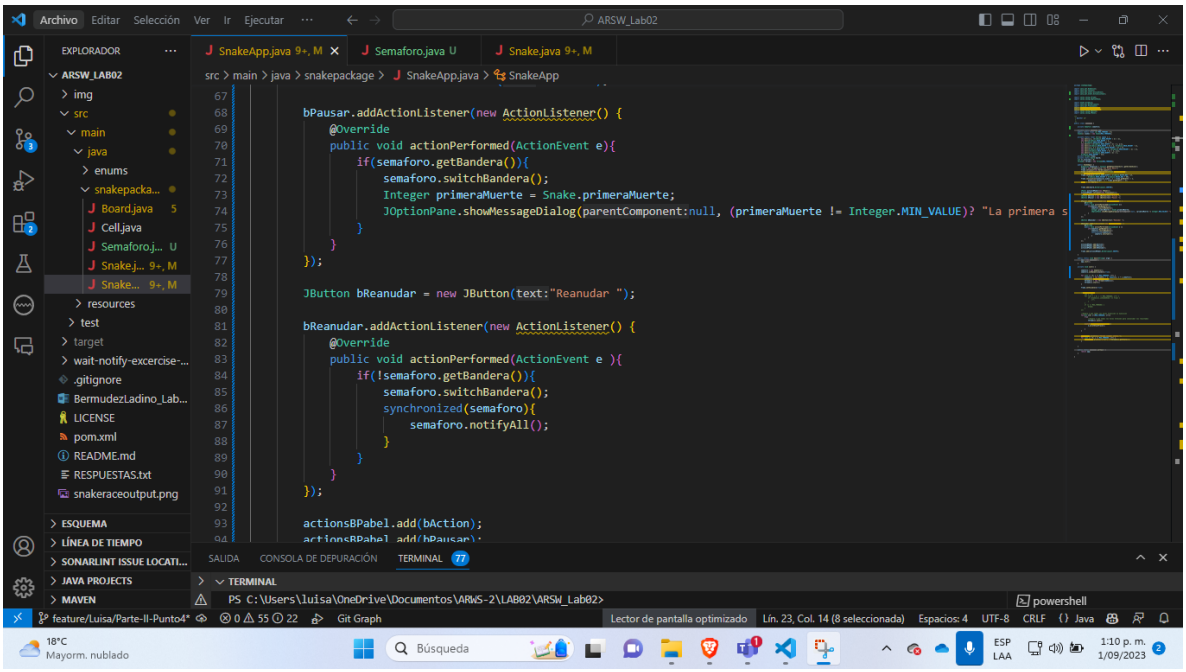

Archivo Editar Selección Ver Ir Ejecutar
ARSW_Lab02
EXPLORADOR
SnakeApp.java 9+, M
Semaforo.java U
Snake.java 9+, M
src > main > java > snakepackage > Snake.java > Snake
ARSW_LAB02
img
src
main
java
enums
snakepacka...
Board.java 5
Cell.java
Semaforo.j... U
Snake.j... 9+, M
Snake... 9+, M
resources
test
target
wait-notify-excercise-...
.gitignore
BermudezLadino_Lab...
LICENSE
pom.xml
README.md
RESPUESTAS.txt
snakeraceoutput.png
ESQUEMA
LÍNEA DE TIEMPO
SONARLINT ISSUE LOCATI...
JAVA PROJECTS
MAVEN
SALIDA
CONSOLA DE DEPURACIÓN
TERMINAL 77
TERMINAL
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB02\ARSW_Lab02>
powershell
feature/Luisa/Parte-II-Punto4*
Git Graph
Lector de pantalla optimizado
Lín. 373, col. 1
Espacios: 4
UTF-8
CRLF
Java
18°C
Parc. soleado
Búsqueda
ESP
LAA
1:18 p. m.
1/09/2023

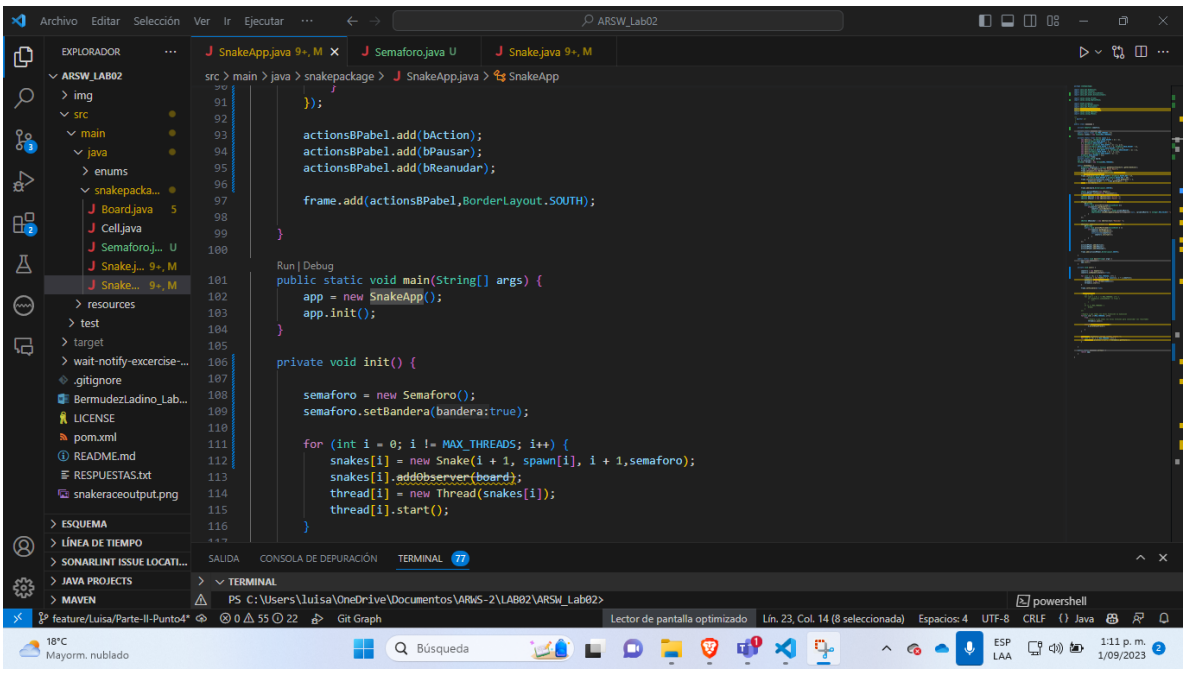

Archivo Editar Selección Ver Ir Ejecutar
ARSW_Lab02
EXPLORADOR
SnakeApp.java 9+, M
Semaforo.java U
Snake.java 9+, M
src > main > java > snakepackage > Snake.java > Snake
ARSW_LAB02
img
src
main
java
enums
snakepacka...
Board.java 5
Cell.java
Semaforo.j... U
Snake.j... 9+, M
Snake... 9+, M
resources
test
target
wait-notify-excercise-...
.gitignore
BermudezLadino_Lab...
LICENSE
pom.xml
README.md
RESPUESTAS.txt
snakeraceoutput.png
ESQUEMA
LÍNEA DE TIEMPO
SONARLINT ISSUE LOCATI...
JAVA PROJECTS
MAVEN
SALIDA
CONSOLA DE DEPURACIÓN
TERMINAL 77
TERMINAL
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB02\ARSW_Lab02>
powershell
feature/Luisa/Parte-II-Punto4*
Git Graph
Lector de pantalla optimizado
Lín. 373, col. 1
Espacios: 4
UTF-8
CRLF
Java
18°C
Parc. soleado
Búsqueda
ESP
LAA
1:18 p. m.
1/09/2023

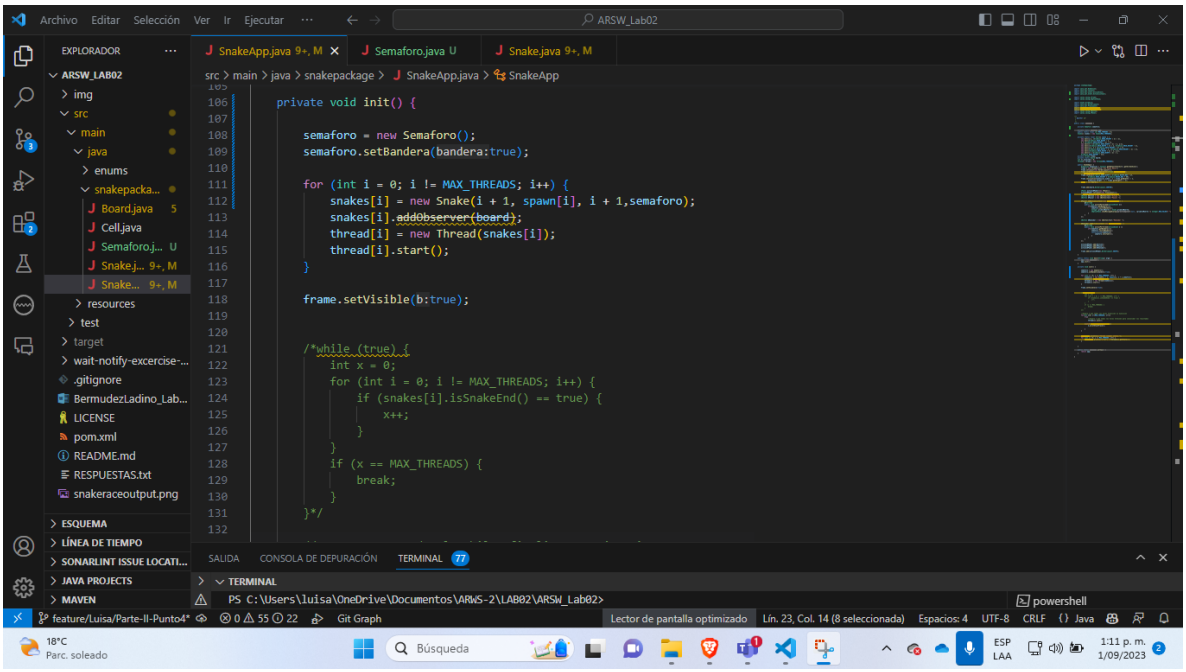
- ✓ **Modificación de la clase SnakeApp para mostrar la serpiente que primero murió**
  Se agregaron los botones Iniciar y Reanudar a la interfaz con sus respectivas acciones teniendo en cuenta las instrucciones del enunciado.
  Adicional a esto se agregó el botón Pausar para que cuando el juego se suspenda muestre la peor serpiente, es decir, la primera en morir.

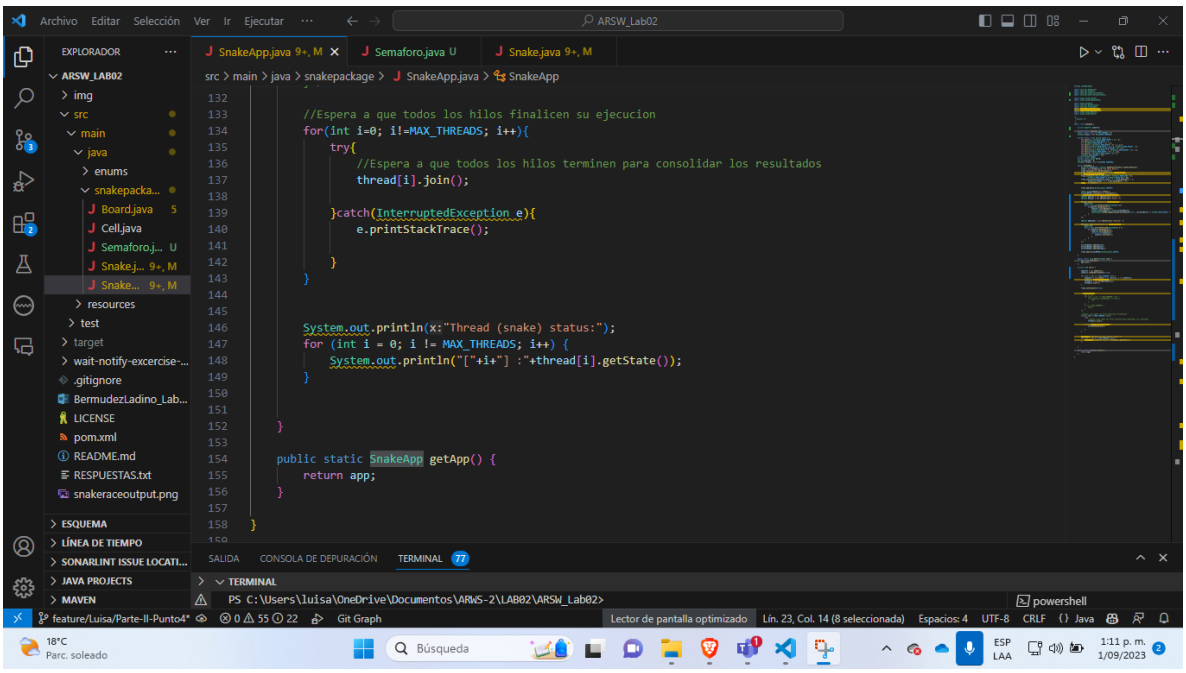




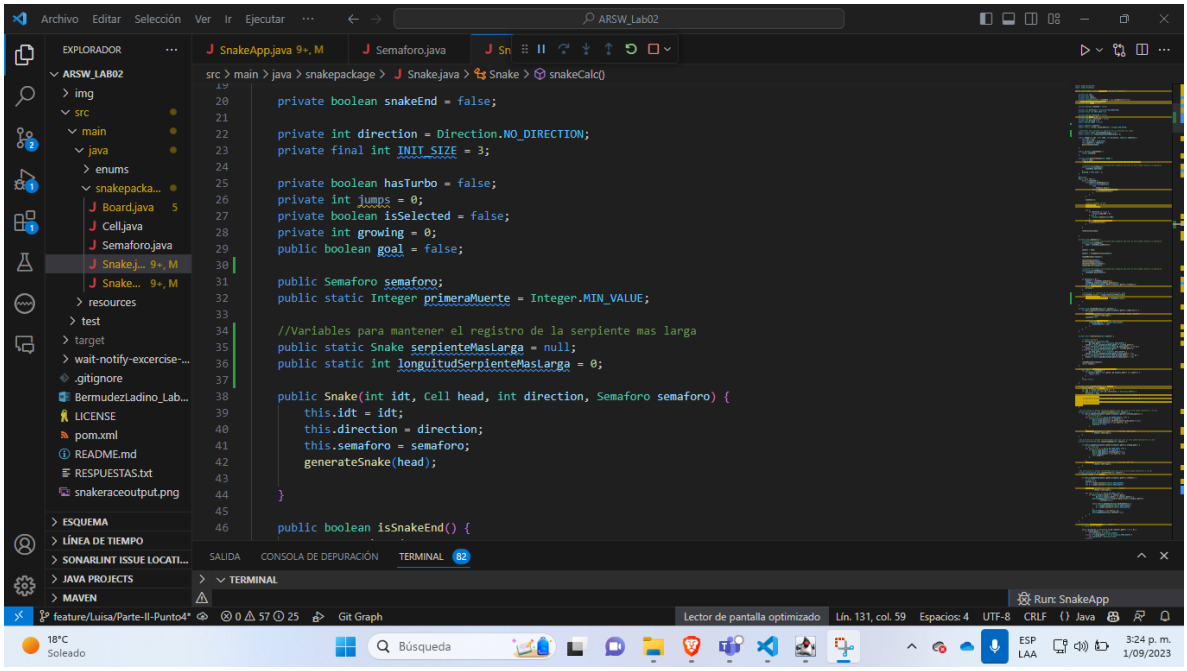

Archivo Editar Selección Ver Ir Ejecutar
ARSW_Lab02
EXPLORADOR
ARSW_LAB02
SnakeApp.java 9+, M
Semaforo.java U
Snake.java 9+, M
src > main > java > snakepackage > SnakeApp.java > SnakeApp
bPausar.addActionListener(new ActionListener() {
@Override
public void actionPerformed(ActionEvent e){
if(semaforo.getBandera()){
semaforo.switchBandera();
Integer primeraMuerte = Snake.primeraMuerte;
JOptionPane.showMessageDialog(parentComponent:null, (primeraMuerte != Integer.MIN_VALUE)? "La primera s
}
}
});
JButton bReanudar = new JButton(text:"Reanudar ");
bReanudar.addActionListener(new ActionListener() {
@Override
public void actionPerformed(ActionEvent e){
if(!semaforo.getBandera()){
semaforo.switchBandera();
synchronized(semaforo){
semaforo.notifyAll();
}
}
}
});
actionsBPabel.add(bAction);
SALIDA
CONSOLA DE DEPURACIÓN
TERMINAL 77
TERMINAL
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB02\ARSW_Lab02>
powershell
feature/Luisa/Parte-II-Punto4*
Git Graph
Lector de pantalla optimizado
Lín. 23, Col. 14 (8 seleccionada)
Espacios: 4
UTF-8
CRLF
Java
18°C
Mayorm. nublado
Búsqueda
ESP
LAA
1:10 p. m.
1/09/2023


Archivo Editar Selección Ver Ir Ejecutar
ARSW_Lab02
EXPLORADOR
ARSW_LAB02
SnakeApp.java 9+, M
Semaforo.java U
Snake.java 9+, M
src > main > java > snakepackage > SnakeApp.java > SnakeApp
});
actionsBPabel.add(bAction);
actionsBPabel.add(bPausar);
actionsBPabel.add(bReanudar);
frame.add(actionsBPabel,BorderLayout.SOUTH);
}
Run | Debug
public static void main(String[] args) {
app = new SnakeApp();
app.init();
}
private void init() {
semaforo = new Semaforo();
semaforo.setBandera(bandera:true);
for (int i = 0; i != MAX_THREADS; i++) {
snakes[i] = new Snake(i + 1, spawn[i], i + 1,semaforo);
snakes[i].addObserver(board);
thread[i] = new Thread(snakes[i]);
thread[i].start();
}
SALIDA
CONSOLA DE DEPURACIÓN
TERMINAL 77
TERMINAL
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB02\ARSW_Lab02>
powershell
feature/Luisa/Parte-II-Punto4*
Git Graph
Lector de pantalla optimizado
Lín. 23, Col. 14 (8 seleccionada)
Espacios: 4
UTF-8
CRLF
Java
18°C
Mayorm. nublado
Búsqueda
ESP
LAA
1:11 p. m.
1/09/2023

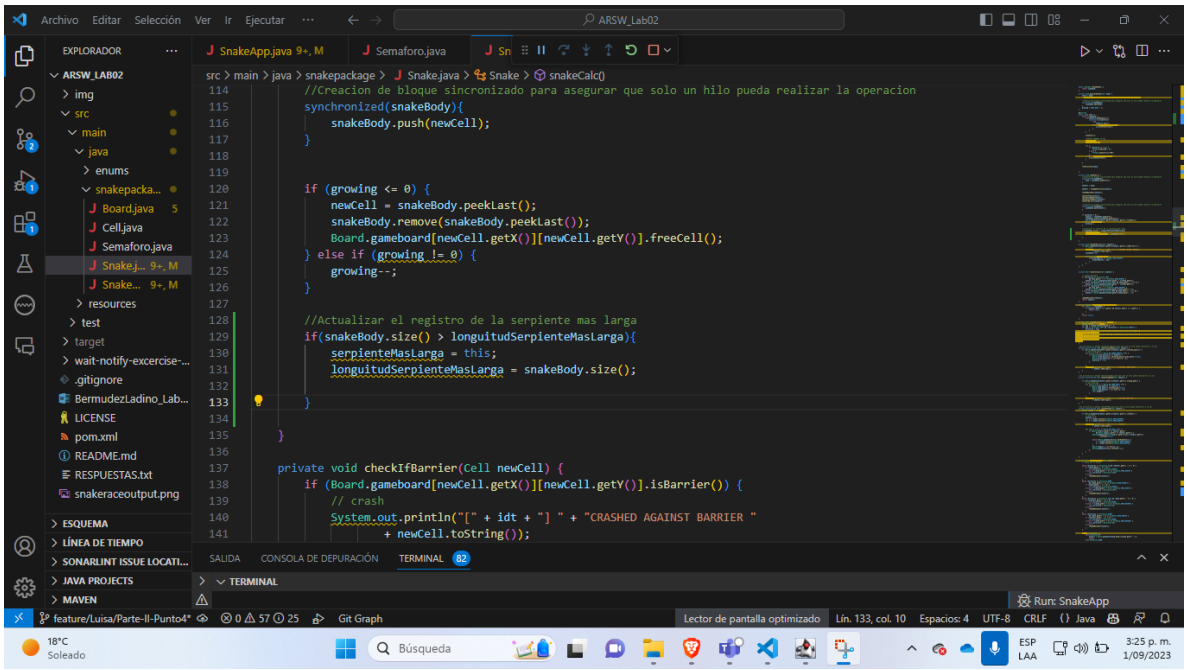

Archivo Editar Selección Ver Ir Ejecutar
ARSW_Lab02
EXPLORADOR
ARSW_LAB02
SnakeApp.java 9+, M
Semaforo.java U
Snake.java 9+, M
src > main > java > snakepackage > SnakeApp.java > SnakeApp
private void init() {
semaforo = new Semaforo();
semaforo.setBandera(bandera:true);
for (int i = 0; i != MAX_THREADS; i++) {
snakes[i] = new Snake(i + 1, spawn[i], i + 1,semaforo);
snakes[i].addObserver(board);
thread[i] = new Thread(snakes[i]);
thread[i].start();
}
frame.setVisible(b:true);
/*while (true) {
int x = 0;
for (int i = 0; i != MAX_THREADS; i++) {
if (snakes[i].isSnakeEnd() == true) {
x++;
}
}
if (x == MAX_THREADS) {
break;
}
}*/
SALIDA
CONSOLA DE DEPURACIÓN
TERMINAL
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB02\ARSW_Lab02>
powershell
feature/Luisa/Parte-II-Punto4*
Git Graph
Lector de pantalla optimizado
Lín. 23, Col. 14 (8 seleccionada)
Espacios: 4
UTF-8
CRLF
Java
18°C
Parc. soleado
Búsqueda
ESP
LAA
1:11 p. m.
1/09/2023


Archivo Editar Selección Ver Ir Ejecutar
ARSW_Lab02
EXPLORADOR
ARSW_LAB02
SnakeApp.java 9+, M
Semaforo.java U
Snake.java 9+, M
src > main > java > snakepackage > SnakeApp.java > SnakeApp
//Espera a que todos los hilos finalicen su ejecucion
for(int i=0; i!=MAX_THREADS; i++){
try{
//Espera a que todos los hilos terminen para consolidar los resultados
thread[i].join();
}catch(InterruptedException e){
e.printStackTrace();
}
}
System.out.println(x:"Thread (snake) status:");
for (int i = 0; i != MAX_THREADS; i++) {
System.out.println("["+i+"] :"+thread[i].getState());
}
}
public static SnakeApp getApp() {
return app;
}
}
SALIDA
CONSOLA DE DEPURACIÓN
TERMINAL
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB02\ARSW_Lab02>
powershell
feature/Luisa/Parte-II-Punto4*
Git Graph
Lector de pantalla optimizado
Lín. 23, Col. 14 (8 seleccionada)
Espacios: 4
UTF-8
CRLF
Java
18°C
Parc. soleado
Búsqueda
ESP
LAA
1:11 p. m.
1/09/2023

✓ **Modificación de la clase Snake para mostrar la serpiente más larga**
Vamos a mantener un registro de la serpiente más larga.

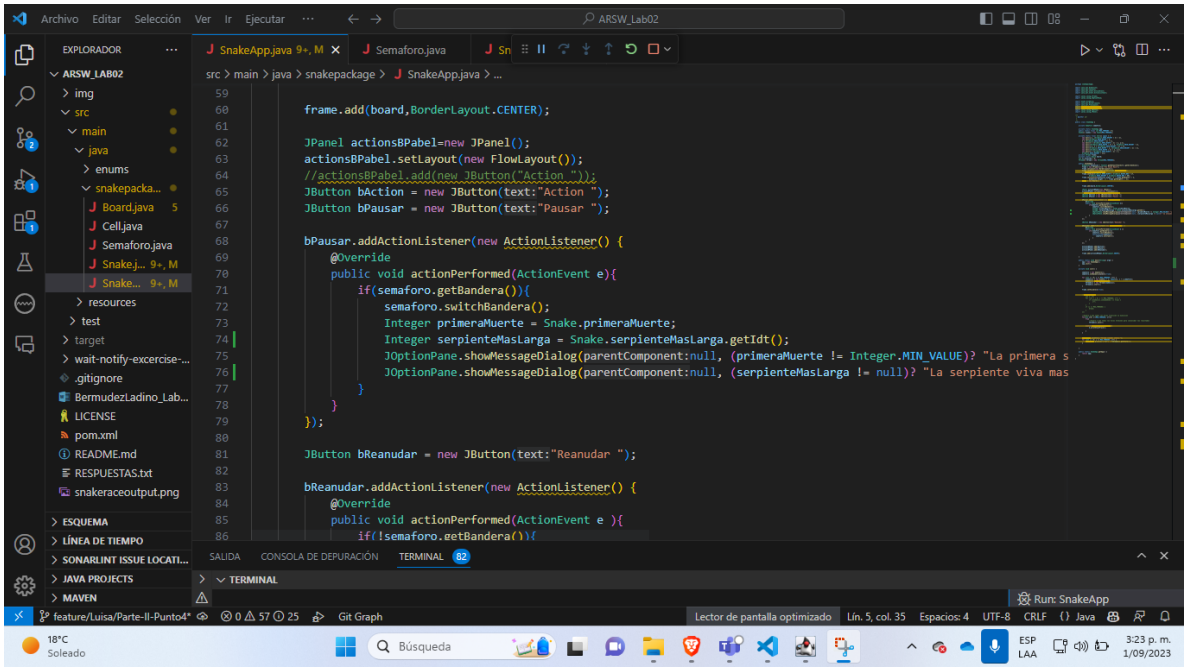


```java
    private boolean snakeEnd = false;

    private int direction = Direction.NO_DIRECTION;
    private final int INIT_SIZE = 3;

    private boolean hasTurbo = false;
    private int jumps = 0;
    private boolean isSelected = false;
    private int growing = 0;
    public boolean goal = false;

    public Semaforo semaforo;
    public static Integer primeraMuerte = Integer.MIN_VALUE;

    //Variables para mantener el registro de la serpiente mas larga
    public static Snake serpienteMasLarga = null;
    public static int longuitudSerpienteMasLarga = 0;

    public Snake(int idt, Cell head, int direction, Semaforo semaforo) {
        this.idt = idt;
        this.direction = direction;
        this.semaforo = semaforo;
        generateSnake(head);

    }

    public boolean isSnakeEnd() {
```



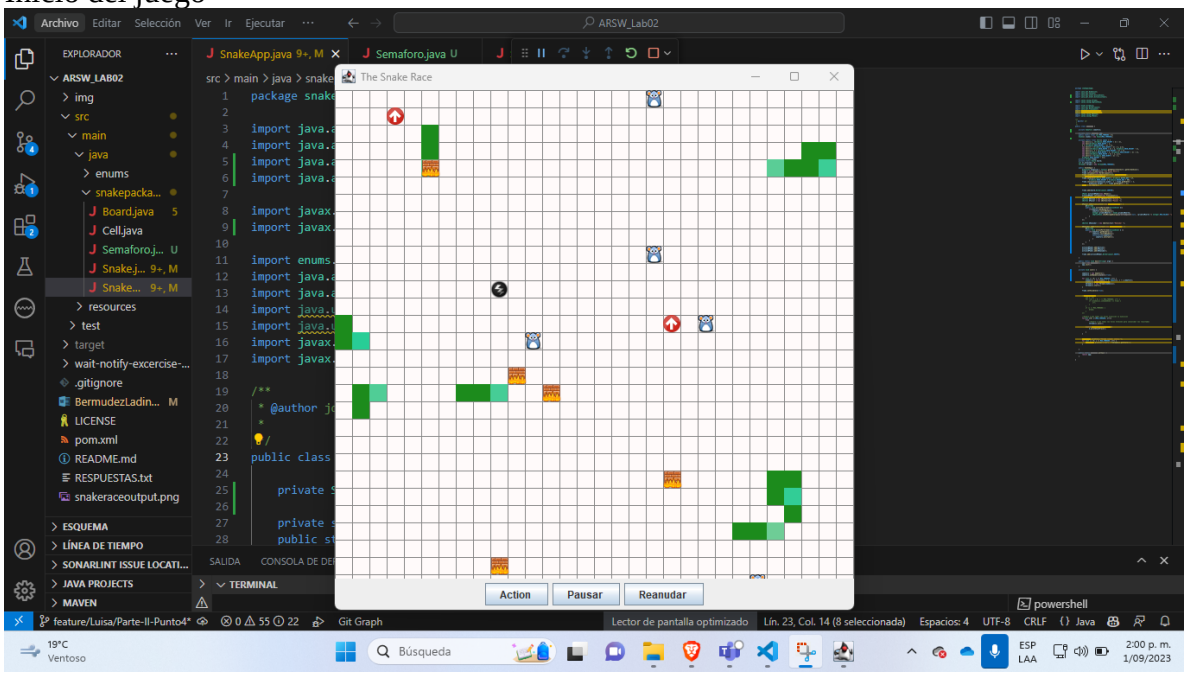
```java
        //Creacion de bloque sincronizado para asegurar que solo un hilo pueda realizar la operacion
        synchronized(snakeBody){
            snakeBody.push(newCell);
        }


        if (growing <= 0) {
            newCell = snakeBody.peekLast();
            snakeBody.remove(snakeBody.peekLast());
            Board.gameboard[newCell.getX()][newCell.getY()].freeCell();
        } else if (growing != 0) {
            growing--;
        }

        //Actualizar el registro de la serpiente mas larga
        if(snakeBody.size() > longuitudSerpienteMasLarga){
            serpienteMasLarga = this;
            longuitudSerpienteMasLarga = snakeBody.size();

        }

    }

    private void checkIfBarrier(Cell newCell) {
        if (Board.gameboard[newCell.getX()][newCell.getY()].isBarrier()) {
            // crash
            System.out.println("[" + idt + "] " + "CRASHED AGAINST BARRIER "
                    + newCell.toString());
```

✓ **Modificación de la clase SnakeApp para mostrar la serpiente más larga**
Se agregó un mensaje para qué momento de dar click en el botón Pausar muestre cual es la serpiente mas larga y su longitud.



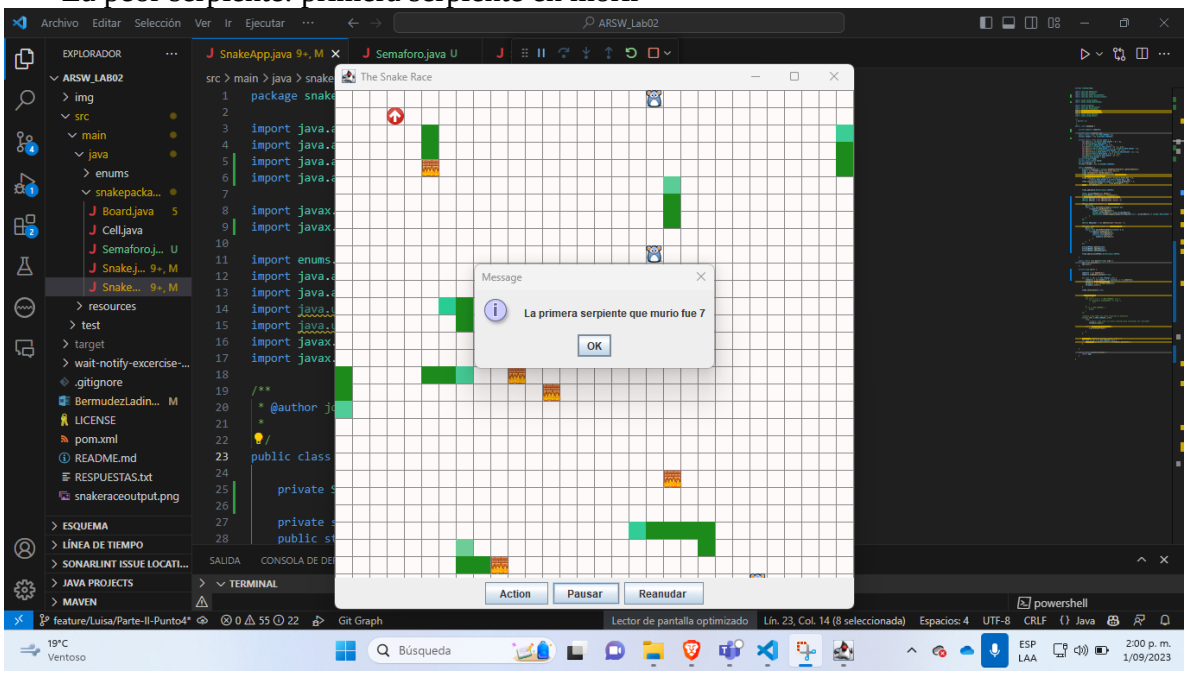
➢ Resultados Ejecución:

Inicio del juego

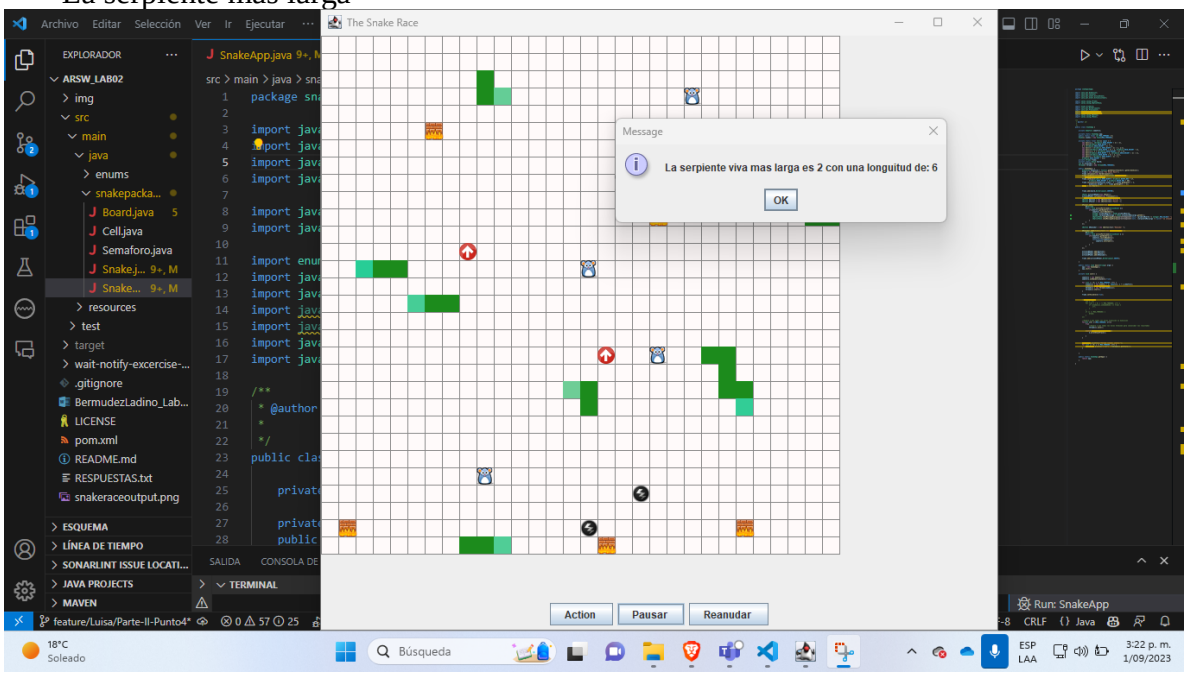


Pausar el juego

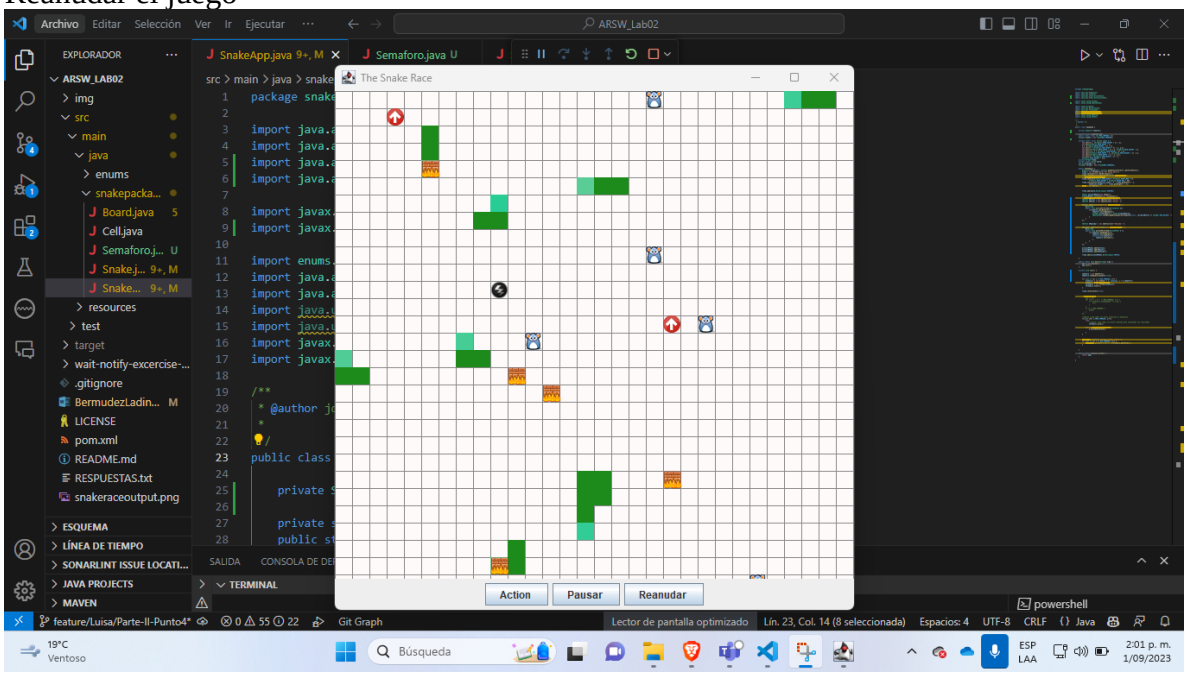
- La peor serpiente: primera serpiente en morir



- La serpiente más larga



Reanudar el juego



III. Conclusiones

- Durante el desarrollo de este laboratorio, hemos explorado los conceptos fundamentales de programación concurrente. Hemos aprendido cómo manejar múltiples hilos de ejecución de manera eficiente y segura en entornos paralelos.

- Hemos aplicado los mecanismos de sincronización proporcionados por el lenguaje Java, como wait, notify y notifyAll, para controlar el comportamiento de los hilos. Esto nos ha permitido pausar y reanudar hilos de manera sincronizada.

- Identificamos posibles condiciones de carrera en el código y con el fin de tomar acciones para evitarlas. Esto incluye el uso de bloqueos synchronized para asegurar el acceso exclusivo a secciones críticas del código.

url repositorio:https://github.com/ARSW2023-2/ARSW_Lab02.git