

### Laboratorio 3

Ejercicio – programación concurrente, condiciones de carrera y sincronización de hilos.

url repositorio:

[https://github.com/ARSW2023-2/ARSW\\_Lab03.git](https://github.com/ARSW2023-2/ARSW_Lab03.git)

Integrantes:

Luisa Fernanda Bermúdez Girón

Karol Daniela Ladino Ladino

Squad:

Inside Out

Profesor:

Javier Iván Toquica Barrera

Curso:

ARSW – 1

Fecha De Entrega:

08-09-2023

# I. Introducción

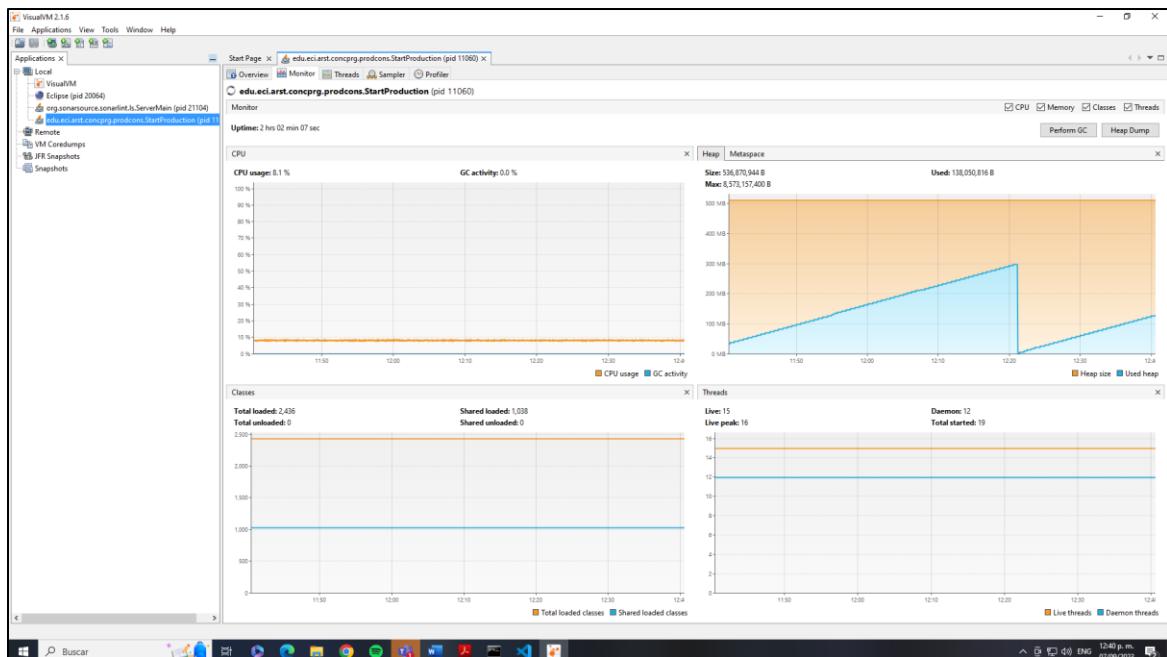
En el ámbito de la programación concurrente y la sincronización de hilos, se presentan desafíos cruciales para el diseño de aplicaciones robustas y eficientes. Este laboratorio se centra en explorar conceptos y técnicas relacionadas con la gestión de hilos, la prevención de condiciones de carrera y la optimización del rendimiento en aplicaciones Java.

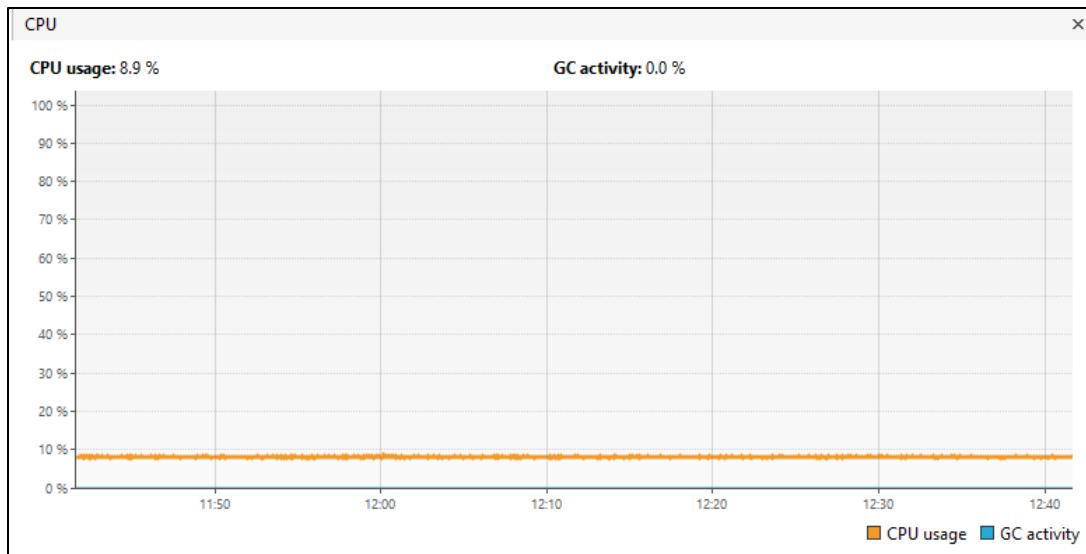
## II. Desarrollo del laboratorio

### Parte I

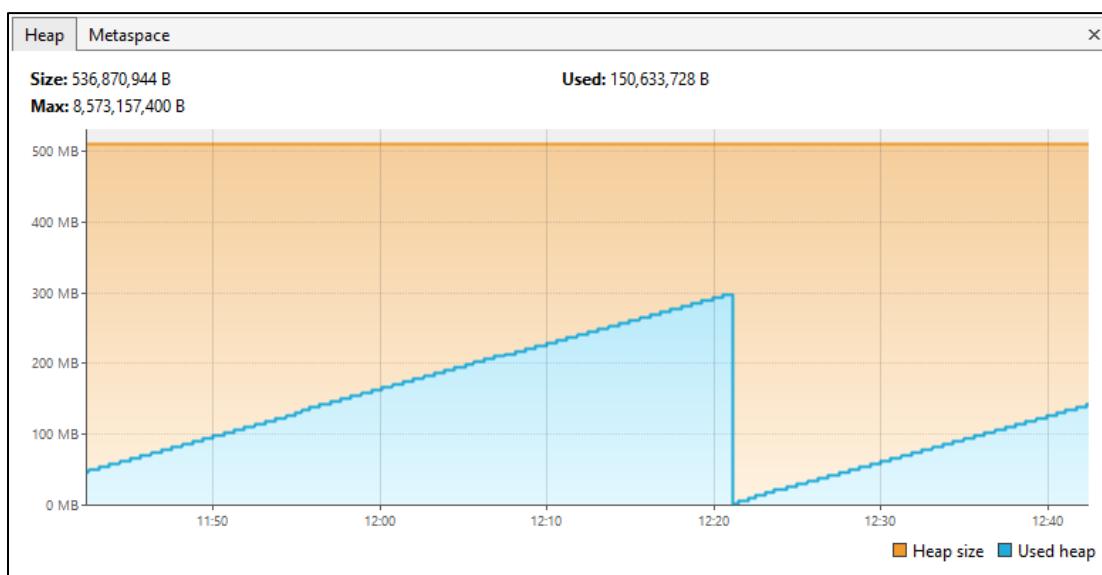
Control de hilos con wait/notify. Productor/consumidor.

1. Revise el funcionamiento del programa y ejecútelo. Mientras esto ocurren, ejecute jVisualVM y revise el consumo de CPU del proceso correspondiente. A qué se debe este consumo?, cual es la clase responsable?





Como podemos observar el pico de la CPU se mantiene por debajo del 10% debido a que la aplicación no consume una gran cantidad de recursos.



Como podemos observar en la gráfica de la memoria hubo un pico a las 12:20, lo cual indica que hubo un consumo mayor de memoria, así mismo en este punto el recolector de basura actúa liberando los objetos que él pueda.

- ¿A qué se debe este consumo?  
La memoria Heap se va llenando debido a que se está produciendo más de lo que se está consumiendo.
- ¿Cuál es la clase responsable?  
La clase responsable es Producer.

2. Haga los ajustes necesarios para que la solución use más eficientemente la CPU, teniendo en cuenta que -por ahora- la producción es lenta y el consumo es rápido. Verifique con JVisualVM que el consumo de CPU se reduzca.

```

src > main > java > edu > eci > arst > conceprg > prodcons > J_Producer.java > Producer.java > run()
25     public Producer<Queue<Integer>> queue,Long stockLimit) {
26         this.queue = queue;
27         rand = new Random(System.currentTimeMillis());
28         this.stockLimit=stockLimit;
29     }
30
31     @Override
32     public void run() {
33         while (true) {
34             dataSeed = dataSeed + rand.nextInt(bound:100);
35             System.out.println("Producer added " + dataSeed);
36             queue.add(dataSeed);
37
38             try {
39                 Thread.sleep(1000);
40             } catch (InterruptedException ex) {
41                 Logger.getLogger(Producer.class.getName()).log(Level.SEVERE, null, ex);
42             }
43         }
44     }
45 }
46
47 }

```

unknown, hace 6 días \* Fuentes base ...

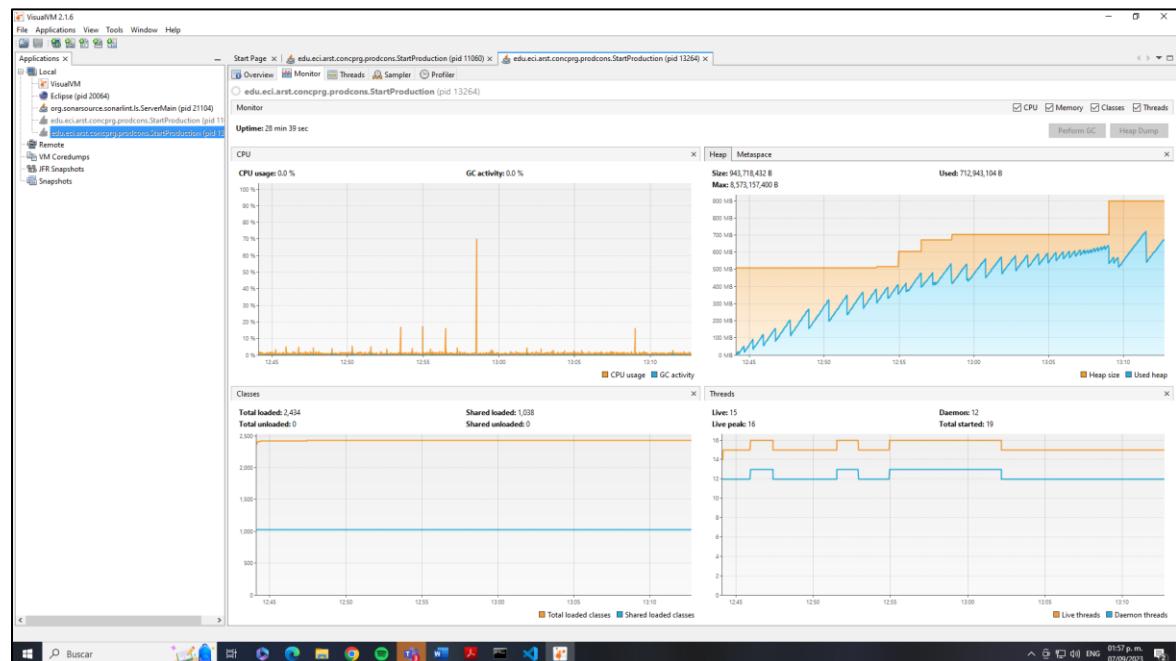
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS GITLENS SQL CONSOLE COMENTARIOS

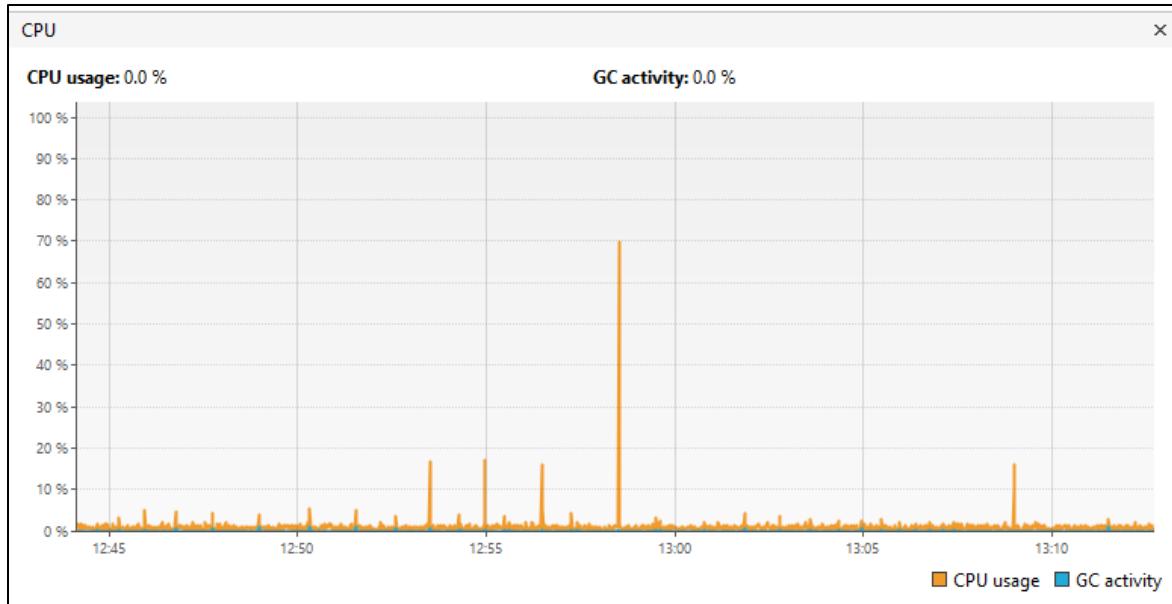
Advertencia: PowerShell detectó que es posible que estés usando un lector de pantalla y que hayas deshabilitado PSReadline con fines de compatibilidad. Si quieres volver a habilitarlo, ejecuta 'Import-Module PSReadline'.

PS C:\Users\ala de dios\Documents\ARSW\ARSW\_Lab03> git checkout -b feature/Daniela/Partel  
Switched to a new branch 'feature/Daniela/Partel'  
PS C:\Users\ala de dios\Documents\ARSW\ARSW\_Lab03> [ ]

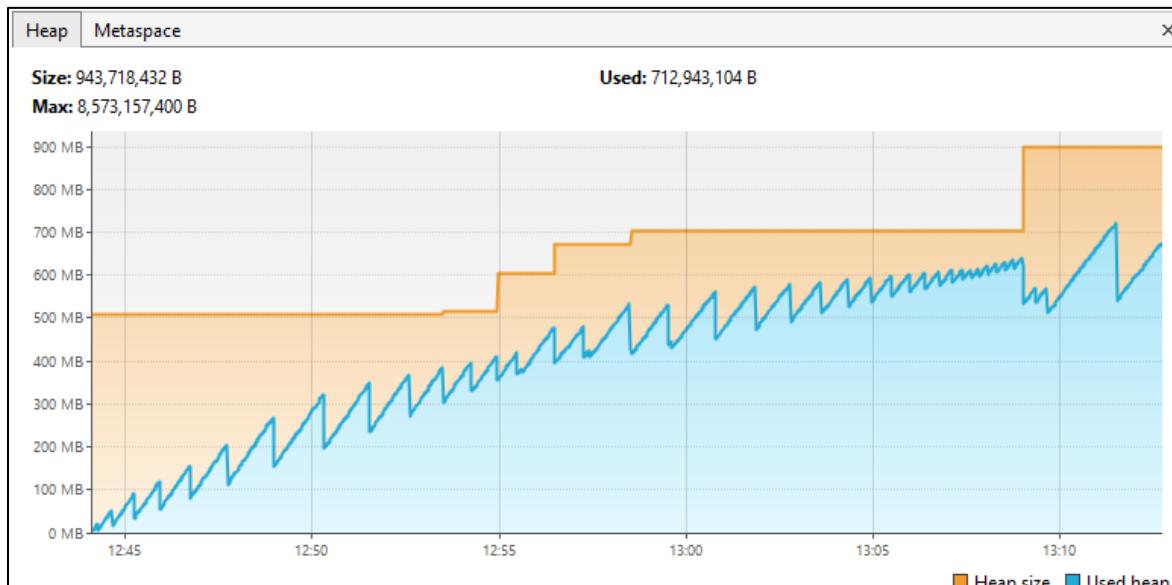
> ESQUEMA > LÍNEA DE TIEMPO > OBJETOS OCULTOS > SONARQURE ISSUE LOCATOR... > JAVA PROJECTS > MAVEN

feature/Daniela/Partel\* Sign in to Bitbucket 0 14 0 7 0 Connect Git Graph unknown, hace 6 días Lin. 46, col. 6 Espacio: 4 UTF-8 CR/LF ( Java 11/30 a. m. 08/09/2023





Como podemos observar la CPU está trabajando más eficientemente debido a que ahora el productor no está esperando y los recursos se están liberando más rápido, es por esto que se presentan picos en un menor tiempo.

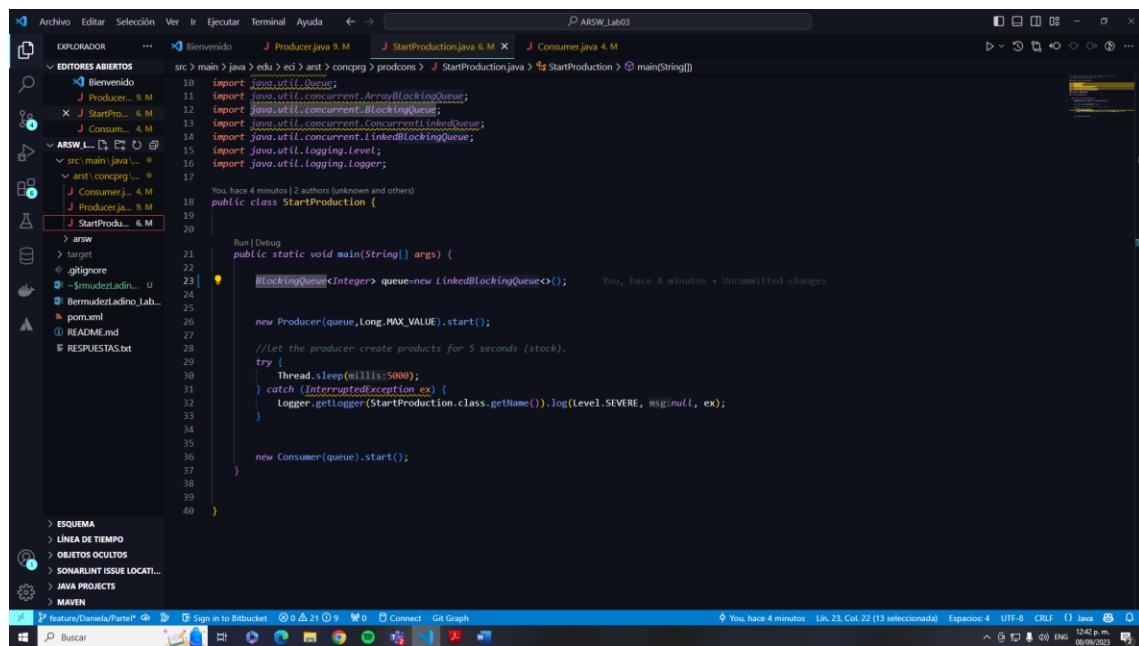


Como podemos observar la memoria registra los picos en un lapso de tiempo más corto y así mismo el recolector de basura libera los objetos de una forma más rápida.

Por otro lado, se puede visualizar que el tamaño del Heap aumenta entre las 12:55 y la 1:00 y nuevamente aumenta a la 1:10 debido a que en el momento en que java considere que debe aumentar este lo va a hacer.

3. Haga que ahora el productor produzca muy rápido, y el consumidor consuma lento. Teniendo en cuenta que el productor conoce un límite de Stock (cuantos elementos debería tener, a lo sumo en la cola), haga que dicho límite se respete. Revise el API de la colección usada como cola para ver cómo garantizar que dicho límite no se supere. Verifique que, al poner un límite pequeño para el 'stock', no haya consumo alto de CPU ni errores.

➤ Modificaciones Clase StartProduction:

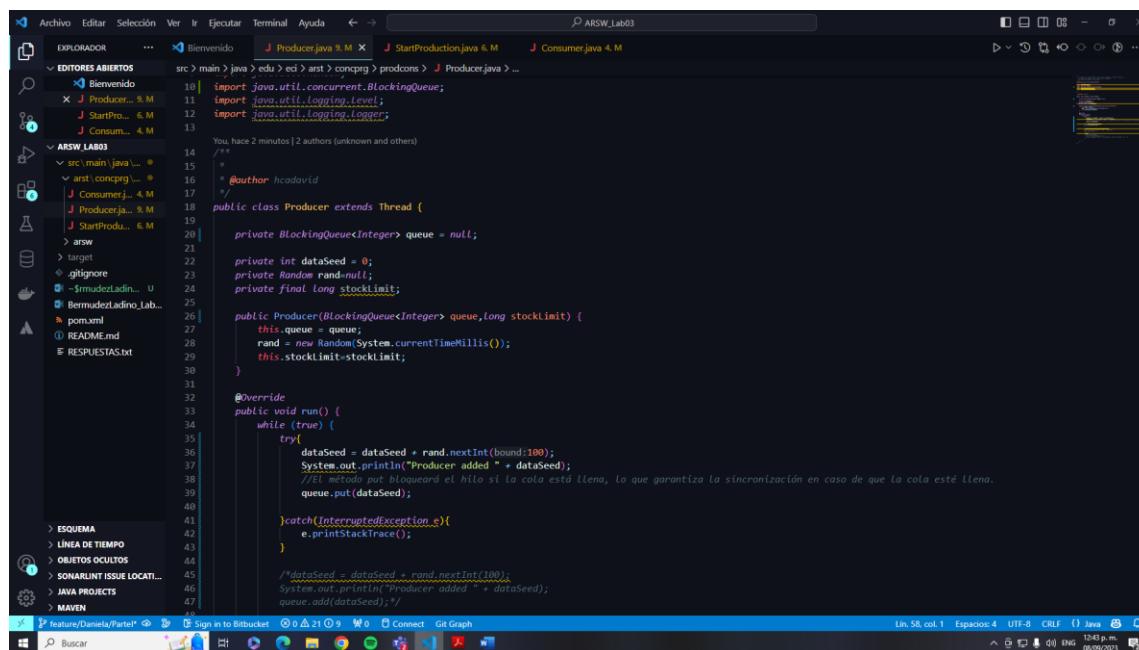


```

10 import java.util.Queue;
11 import java.util.concurrent.ArrayBlockingQueue;
12 import java.util.concurrent.BlockingQueue;
13 import java.util.concurrent.ConcurrentLinkedQueue;
14 import java.util.concurrent.LinkedBlockingQueue;
15 import java.util.logging.Level;
16 import java.util.logging.Logger;
17
18 You have 4 minutes | 2 authors (unknown and others)
19 public class StartProduction {
20
21     public static void main(String[] args) {
22
23         BlockingQueue<Integer> queue=new LinkedBlockingQueue<>();
24
25         new Producer(queue,Long.MAX_VALUE).start();
26
27         //Let the producer create products for 5 seconds (stock).
28         try {
29             Thread.sleep(5000);
30         } catch (InterruptedException ex) {
31             Logger.getLogger(StartProduction.class.getName()).log(Level.SEVERE, null, ex);
32         }
33
34
35         new Consumer(queue).start();
36
37     }
38
39 }
40

```

➤ Modificaciones Clase Producer:

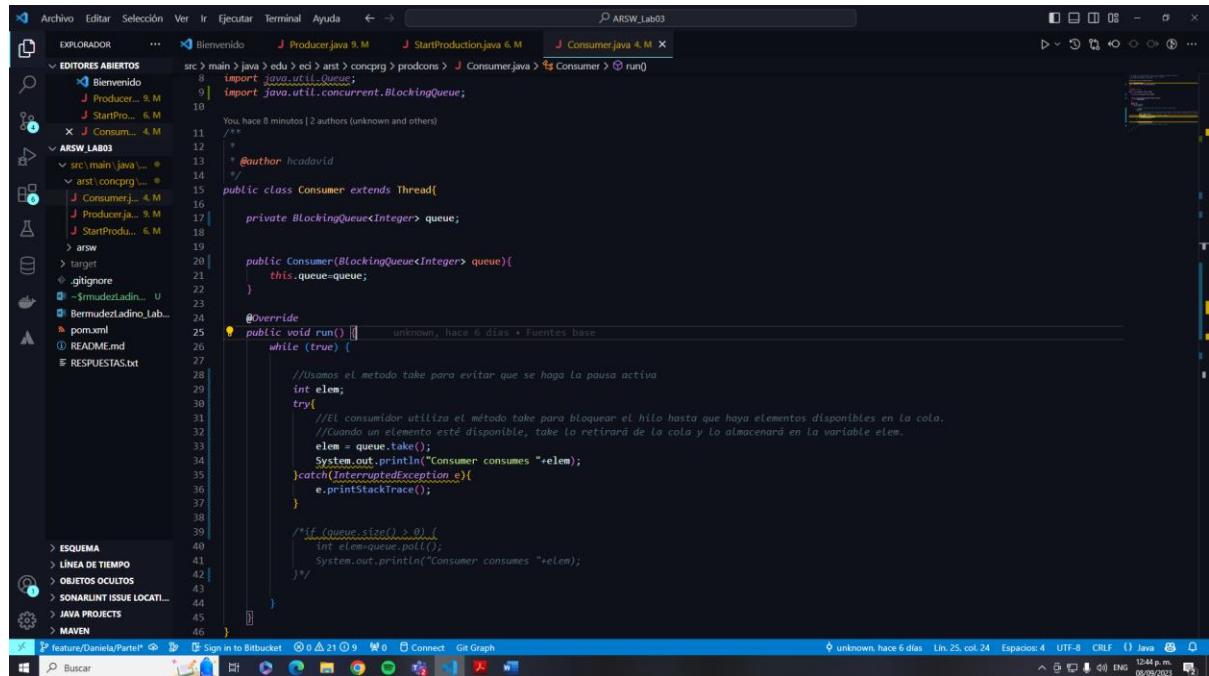


```

10 import java.util.concurrent.BlockingQueue;
11 import java.util.logging.Level;
12 import java.util.logging.Logger;
13
14 /**
15 * 
16 * @author hadavid
17 */
18 public class Producer extends Thread {
19
20     private BlockingQueue<Integer> queue = null;
21
22     private int dataSeed = 0;
23     private Random rand=null;
24     private final long stockLimit;
25
26     public Producer(BlockingQueue<Integer> queue,long stockLimit) {
27         this.queue = queue;
28         rand = new Random(System.currentTimeMillis());
29         this.stockLimit=stockLimit;
30     }
31
32     @Override
33     public void run() {
34         while (true) {
35             try{
36                 dataSeed = dataSeed + rand.nextInt(100);
37                 System.out.println("Producer added " + dataSeed);
38                 //El método put bloqueará el hilo si la cola está llena, lo que garantiza la sincronización en caso de que la cola esté llena.
39                 queue.put(dataSeed);
40             }catch(InterruptedException e){
41                 e.printStackTrace();
42             }
43
44             /*dataSeed = dataSeed + rand.nextInt(100);
45             System.out.println("Producer added " + dataSeed);
46             queue.add(dataSeed);*/
47         }
48     }
49
50 }
51

```

## ➤ Modificaciones Clase Consumer:



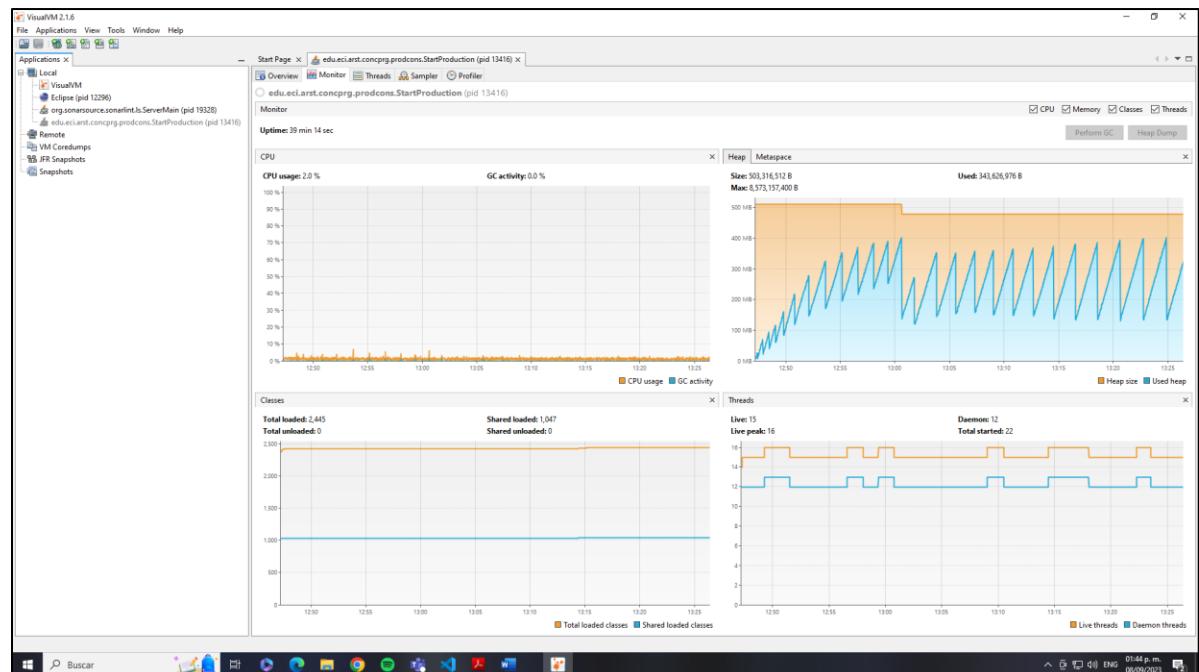
```

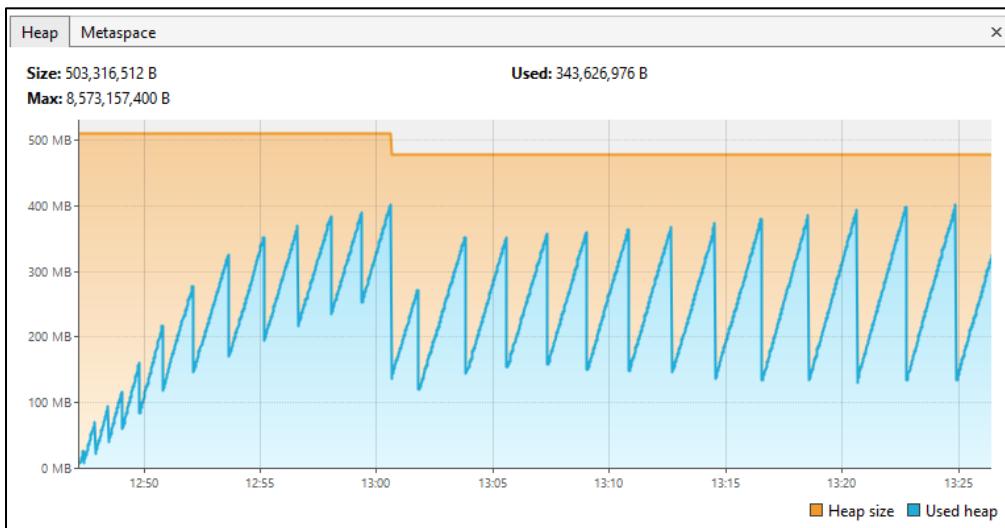
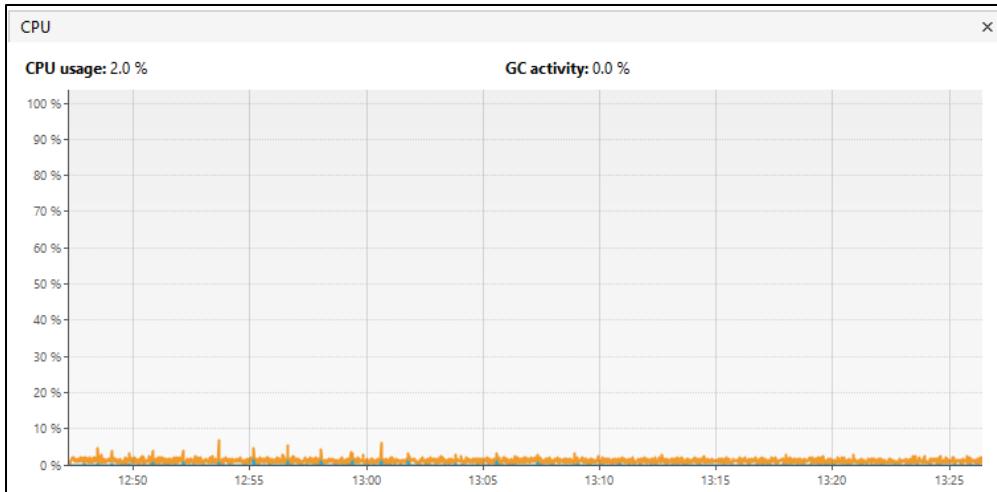
src/main/java/edu/eci/arsl/concprg/prodcons/Consumer.java
import java.util.concurrent.BlockingQueue;
import java.util.concurrent.TimeUnit;

public class Consumer extends Thread{
    private BlockingQueue<Integer> queue;
    public Consumer(BlockingQueue<Integer> queue){
        this.queue=queue;
    }
    @Override
    public void run(){
        while (true){
            try{
                //Usamos el método take para evitar que se haga la pausa activa
                int elem;
                //El consumidor utiliza el método take para bloquear el hilo hasta que haya elementos disponibles en la cola.
                //Cuando un elemento esté disponible, take lo retirará de la cola y lo almacenará en la variable elem.
                elem = queue.take();
                System.out.println("Consumer consumes "+elem);
            }catch(InterruptedException e){
                e.printStackTrace();
            }
            /*if(queue.size()>0){
                int elem=queue.poll();
                System.out.println("Consumer consumes "+elem);
            }*/
        }
    }
}

```

## ➤ Resultados ejecución:





## Parte II

Teniendo en cuenta los conceptos vistos de condición de carrera y sincronización, haga una nueva versión -más eficiente- del ejercicio anterior (el buscador de listas negras). En la versión actual, cada hilo se encarga de revisar el host en la totalidad del subconjunto de servidores que le corresponde, de manera que en conjunto se están explorando la totalidad de servidores. Teniendo esto en cuenta, haga que:

- La búsqueda distribuida se detenga (deje de buscar en las listas negras restantes) y retorne la respuesta apenas, en su conjunto, los hilos hayan detectado el número de ocurrencias requerido que determina si un host es confiable o no (*BLACK\_LIST\_ALARM\_COUNT*).
- Lo anterior, garantizando que no se den condiciones de carrera.

➤ Creamos la clase Contador

The screenshot shows the Eclipse IDE interface with the ARSW\_Lab03 project open. The left sidebar displays the project structure under 'EXPLORADOR'. In the center, the code editor shows the 'Contador.java' file. The code defines a class 'Contador' with private fields 'bandera' and 'conteoActual', and a constructor that initializes 'conteoFinal'. It includes synchronized methods for getting and setting the 'bandera' value, and for incrementing the 'conteoActual' counter.

```
src > main > java > edu > eci > arsw > blacklistvalidator > J Contador.java X
1 package edu.eci.arsw.blacklistvalidator;
2
3
4 public class Contador {
5     private boolean bandera;
6     private int conteoActual;
7     private int conteoFinal;
8
9     public Contador(boolean bandera, int conteoFinal){
10         this.conteoActual = 0;
11         this.conteoFinal = conteoFinal;
12         this.bandera = bandera;
13     }
14
15     public synchronized boolean getBandera(){
16         return bandera;
17     }
18
19     private synchronized void setBandera(boolean bandera){
20         this.bandera = bandera;
21     }
22
23     public synchronized void aumentarConteo(){
24         conteoActual++;
25         if(conteoActual == conteoFinal){
26             setBandera(false);
27         }
28     }
29
30 }
31
32 }
```

➤ Modificación de la clase HostBlackListValidator

The screenshot shows the Eclipse IDE interface with the ARSW\_Lab03 project open. The left sidebar displays the project structure under 'EXPLORADOR'. In the center, the code editor shows the 'HostBlackListValidator.java' file. The code defines a class 'HostBlackListValidator' with a static final integer 'BLACK\_LIST\_ALARM\_COUNT' set to 5. It contains a detailed multi-line comment explaining the purpose of the class, which is to check a host's IP address against multiple blacklists and report its trustworthiness.

```
J HostBlackListValidator.java 2, U
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.blacklistvalidator;
7
8 import edu.eci.arsw.spamkeywordsdatasource.HostBlacklistsDataSourceFacade;
9 import java.util.LinkedList;
10 import java.util.List;
11 import java.util.logging.Level;
12 import java.util.logging.Logger;
13
14 /**
15  * @author hcadavid
16  */
17
18 public class HostBlackListValidator {
19
20     private static final int BLACK_LIST_ALARM_COUNT=5;
21
22
23     /**
24      * Check the given host's IP address in all the available black lists,
25      * and report it as NOT Trustworthy when such IP was reported in at least
26      * BLACK_LIST_ALARM_COUNT lists, or as Trustworthy in any other case.
27      * The search is not exhaustive: When the number of occurrences is equal to
28      * BLACK_LIST_ALARM_COUNT, the search is finished, the host reported as
29      * NOT Trustworthy, and the list of the five blacklists returned.
30      * @param ipAddress suspicious host's IP address.
31      * @return Blacklists numbers where the given host's IP address was found.
32     */
33 }
```

Explorador de Proyecto: ARSW\_Lab03

```

src > main > java > edu > eci > arsw > blacklistvalidator > J HostBlackListsValidator.java > HostBlackListsValidator
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64

```

HostBlackListsValidator.java (Linea 18, Col. 14)

```

public List<Integer> checkHost(String ipaddress,int numHilos){
    ListaServidoresThread [] hilos = new ListaServidoresThread[numHilos];
    HostBlacklistsDataSourceFacade skds=HostBlacklistsDataSourceFacade.getInstance();

    //tamano seccion = tamano region / numero de hilos
    int tamanoSeccion = (int) (Math.ceil((skds.getRegisteredServersCount())/numHilos));
    Contador contador = new Contador(bandera:true,BLACK_LIST_ALARM_COUNT);
    for (int i = 0; i < numHilos; i++) {
        hilos[i] = new ListaServidoresThread(
            i*tamanoSeccion,
            (i*numHilos-1) ? (i+1)*tamanoSeccion :skds.getRegisteredServersCount(),
            ipaddress, contador);
        hilos[i].start();
    }

    int occurrencesCount=0;
    int checkedListsCount=0;
    LinkedList<Integer> blackListOcurrences=new LinkedList<>();

    for(ListaServidoresThread h: hilos){
        try {
            h.join();
            occurrencesCount += h.getCanOcurrences();
            checkedListsCount += h.getCanServidores();
            blackListOcurrences.addAll(h.getBlackListOcurrences());
        } catch (InterruptedException ex) {
            Logger.getLogger(HostBlacklistsValidator.class.getName()).log(Level.SEVERE, null, ex);
            h.interrupt();
        }
    }
}

```

Lector de pantalla optimizado | Lin. 18, Col. 14 (23 seleccionada) | Espacios: 4 | UTF-8 | CRLF | Java | Git Graph | 11:11 p.m. 8/09/2023

Explorador de Proyecto: ARSW\_Lab03

```

src > main > java > edu > eci > arsw > blacklistvalidator > J HostBlackListsValidator.java > HostBlackListsValidator
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94

```

HostBlackListsValidator.java (Linea 18, Col. 14)

```

/* LinkedList<Integer> blackListOcurrences=new LinkedList<>();

int occurrencesCount=0;

HostBlacklistsDataSourceFacade skds=HostBlacklistsDataSourceFacade.getInstance();

int checkedListsCount=0;

for (int i=0;i<skds.getRegisteredServersCount() && occurrencesCount<BLACK_LIST_ALARM_COUNT;i++){

    checkedListsCount++;

    if (skds.isInBlackListServer(i, ipaddress)){

        blackListOcurrences.add(i);

        occurrencesCount++;
    }
}

if (occurrencesCount>=BLACK_LIST_ALARM_COUNT){

    skds.reportAsNotTrustworthy(ipaddress);
}
else{
    skds.reportAsTrustworthy(ipaddress);
}

LOG.log(Level.INFO, msg:"Checked Black Lists:{0} of {1}", new Object[]{checkedListsCount, skds.getRegisteredServersCount()});
return blackListOcurrences;
}

```

Lector de pantalla optimizado | Lin. 18, Col. 14 (23 seleccionada) | Espacios: 4 | UTF-8 | CRLF | Java | Git Graph | 11:12 p.m. 8/09/2023

The screenshot shows the Eclipse IDE interface with the ARSW\_Lab03 project open. The left pane displays the project structure under the EXPLORADOR view, showing packages like ARSW\_LAB03, PARTE II, and src, containing classes such as Contador.java, HostBlackListsValidator.java, Main.java, etc. The right pane shows the code editor with HostBlackListsValidator.java. The code implements a Validator interface, checking if an IP address has exceeded a threshold of occurrences in blacklists. It uses a static logger and a Skd's instance to report trustworthiness.

```

    package main.java.edu.eci.arsw.blacklistvalidator;
    import java.util.LinkedList;
    import edu.eci.arsw.spamkeywordsdatasource.HostBlacklistsDataSourceFacade;
    public class HostBlackListsValidator extends Validator {
        private static final Logger LOG = Logger.getLogger(HostBlackListsValidator.class.getName());
        private Skd's skds;
        private static final int BLACK_LIST_ALARM_COUNT = 10;
        public void validate(String ipAddress) {
            int occurrencesCount = 0;
            for (String checkedList : checkedLists) {
                occurrencesCount++;
            }
            if (occurrencesCount >= BLACK_LIST_ALARM_COUNT) {
                skds.reportAsNotTrustworthy(ipAddress);
            } else {
                skds.reportAsTrustworthy(ipAddress);
            }
            LOG.log(Level.INFO, msg: "Checked Black Lists:{0} of {1}", new Object[]{checkedListsCount, skds.getRegistered});
            return blackListOccurrences;
        }
        private static final Logger LOG = Logger.getLogger(HostBlackListsValidator.class.getName());
    }

```

## ➤ Modificación de la clase ListaServidoresThread

The screenshot shows the Eclipse IDE interface with the ARSW\_Lab03 project open. The left pane displays the project structure under the EXPLORADOR view, showing packages like ARSW\_LAB03, PARTE II, and src, containing classes such as Contador.java, HostBlackListsValidator.java, Main.java, etc. The right pane shows the code editor with ListaServidoresThread.java. The code defines a Thread class that implements the ListaServidoresThread interface. It tracks the start and end of a section, the IP address being checked, and the count of occurrences. It also maintains a linked list of integers representing the occurrence counts of each server in the blacklist.

```

    package main.java.edu.eci.arsw.blacklistvalidator;
    import java.lang.Thread;
    import java.util.LinkedList;
    import edu.eci.arsw.spamkeywordsdatasource.HostBlacklistsDataSourceFacade;
    public class ListaServidoresThread extends Thread {
        private int inicioSeccion;
        private int finSeccion;
        //cantidad de ocurrencias de la ip en la lista negra
        private int cantOcurrencias;
        private String ipAddress;
        //Cantidad de servidores que se encuentran en la lista negra
        private int cantServidores;
        private LinkedList<Integer> blackListOccurrences;
        private Contador contador;
        public ListaServidoresThread(int inicioSeccion, int finSeccion, String ipAddress, Contador contador) {
            super();
            this.inicioSeccion = inicioSeccion;
            this.finSeccion = finSeccion;
            this.ipAddress = ipAddress;
            this.blackListOccurrences = new LinkedList<>();
            this.contador = contador;
        }
        public int getCanOcurrencias() {
            return cantOcurrencias;
        }
    }

```

```
public int getCanOcurrencias() {
    return cantOcurrencias;
}

public int getCantServidores() {
    return cantServidores;
}

public LinkedList<Integer> getBlackListOcurrences() {
    return blackListOcurrences;
}

@Override
public void run(){
    int i = inicioSeccion;
    while(contador.getBandera() && i<finSeccion){
        if ((HostBlacklistsDataSourceFacade.getInstance().isInBlackListServer(i, ipaddress)) {
            blackListOcurrences.add(i);
            cantOcurrencias++;
            contador.aumentarConteo();
        }
        cantServidores++;
        i++;
    }
}
```

#### ➤ Resultados al ejecutar:

The screenshot shows the Eclipse IDE interface with the ARSW\_Lab03 project open. The code editor displays Main.java, which contains the following code:

```
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.blacklistvalidator;
7
8 import java.util.List;
9
10 /**
11 *
12 * @author hcadavid
13 */
14 public class Main {
15
16     Run | Debug
17     public static void main(String a[]) {
18
19     }
20 }
```

The terminal window shows the output of the application:

```
powerShell
Run: Main
dator>Main
sep. 08, 2023 11:09:34 P.M. edu.eci.arsw.spankeywordsdatasource.HostBlacklistsDataSourceFacade reportAsNotTrustworthy

INFO: HOST 202.24.34.55 Reported as NOT trustworthy
sep. 08, 2023 11:09:34 P.M. edu.eci.arsw.blacklistvalidator.HostBlacklistsValidator checkHost
INFO: Checked Black Lists:[60,863 of 80,000
The host was found in the following blacklists:[29, 10034, 20200, 31000, 70500]
PS C:\Users\luisa\OneDrive\Documentos\ARWIS-2\LAB03\ARSW_Lab03>
```

## Parte III

1. Revise el programa “highlander-simulator”, dispuesto en el paquete `edu.eci.arsw.highlandersim`. Este es un juego en el que:
  - Se tienen N jugadores inmortales.
  - Cada jugador conoce a los N-1 jugadores restantes.
  - Cada jugador, permanentemente, ataca a algún otro inmortal. El que primero ataca le resta M puntos de vida a su contrincante, y aumenta en esta misma cantidad sus propios puntos de vida.
  - El juego podría nunca tener un único ganador. Lo más probable es que al final sólo queden dos, peleando indefinidamente quitando y sumando puntos de vida.

➤ **ControlFrame:**

Este código crea una interfaz gráfica simple que permite controlar una simulación de inmortales que luchan entre sí. Los inmortales generan informes sobre su estado y actividades, que se muestran en un área de texto en la GUI. Los botones permiten iniciar, pausar, reanudar y detener la simulación, así como verificar estadísticas durante la simulación.

➤ **ImmortalUpdateReportCallback:**

Es una interfaz que se utiliza para definir un contrato que las clases pueden implementar si desean proporcionar una forma de procesar y reportar información sobre acciones o eventos relacionados con objetos inmortales.

➤ **Immortal:**

Modela un ser inmortal que lucha contra otros inmortales en una simulación. Cada inmortal tiene un nombre, salud, valor de daño predeterminado y puede comunicar sus acciones a través del objeto `ImmortalUpdateReportCallback`. El método `run` simula la participación del inmortal en una lucha continua con otros inmortales.

2. Revise el código e identifique cómo se implementó la funcionalidad antes indicada. Dada la intención del juego, un invariante debería ser que la sumatoria de los puntos de vida de todos los jugadores siempre sea el mismo (claro está, en un instante de tiempo en el que no esté en proceso una operación de incremento/reducción de tiempo). Para este caso, para N jugadores, cual debería ser este valor?

**Respuesta:** para N jugadores el valor debería ser `N * DEFAULT_INMORTAL_HEALTH` que es la variable estática que indica la cantidad de vida con la que inicia cada jugador. Por lo tanto, el invariante de la suma es cero que es el valor mínimo de vida que pueden tener los jugadores, mientras que el máximo dependerá de la cantidad de jugadores que se tengan.

3. Ejecute la aplicación y verifique cómo funcionan la opción ‘pause and check’. Se cumple el invariante?

```

1: import java.awt.event.ActionListener;
2: import java.awt.event.ActionEvent;
3:
4: import javax.swing.JLabel;
5: import javax.swing.JTextField;
6: import java.awt.Color;
7: import javax.swing.JScrollPane;
8:
9: public class ControlFrame extends JFrame {
10:
11    private static final int DEFAULT_IMMORTAL_HEALTH = 100;
12    private static final int DEFAULT_DAMAGE_VALUE = 10;
13
14    private JPanel contentPane;
15
16    private List<Immortal> immortals;
17
18    private JTextField output;
19    private JLabel statisticLabel;
20
21    public ControlFrame() {
22        immortals = new ArrayList<Immortal>();
23
24        contentPane = new JPanel();
25
26        contentPane.setLayout(new BorderLayout());
27
28        contentPane.add(statisticLabel, "North");
29
30        contentPane.add(output, "Center");
31
32        setContentPane(contentPane);
33
34        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
35        setBounds(100, 100, 450, 300);
36    }
37
38    public void addImmortal(Immortal immortal) {
39        immortals.add(immortal);
40
41        updateStatisticLabel();
42    }
43
44    public void removeImmortal(Immortal immortal) {
45        immortals.remove(immortal);
46
47        updateStatisticLabel();
48    }
49
50    public void attackImmortal(Immortal target) {
51        target.setHealth(target.getHealth() - DEFAULT_DAMAGE_VALUE);
52
53        updateStatisticLabel();
54    }
55
56    public void healImmortal(Immortal target) {
57        target.setHealth(Math.min(DEFAULT_IMMORTAL_HEALTH, target.getHealth() + DEFAULT_DAMAGE_VALUE));
58
59        updateStatisticLabel();
60    }
61
62    private void updateStatisticLabel() {
63        statisticLabel.setText("Immortals total health: " + immortals.stream().map(Immortal::getHealth).reduce(Integer::sum).orElse(0));
64    }
65
66    public static void main(String[] args) {
67        SwingUtilities.invokeLater(() -> {
68            new ControlFrame().setVisible(true);
69        });
70    }
71}

```

```

1: import java.awt.event.ActionListener;
2: import java.awt.event.ActionEvent;
3:
4: import javax.swing.JLabel;
5: import javax.swing.JTextField;
6: import java.awt.Color;
7: import javax.swing.JScrollPane;
8:
9: public class ControlFrame extends JFrame {
10:
11    private static final int DEFAULT_IMMORTAL_HEALTH = 100;
12    private static final int DEFAULT_DAMAGE_VALUE = 10;
13
14    private JPanel contentPane;
15
16    private List<Immortal> immortals;
17
18    private JTextField output;
19    private JLabel statisticLabel;
20
21    public ControlFrame() {
22        immortals = new ArrayList<Immortal>();
23
24        contentPane = new JPanel();
25
26        contentPane.setLayout(new BorderLayout());
27
28        contentPane.add(statisticLabel, "North");
29
30        contentPane.add(output, "Center");
31
32        setContentPane(contentPane);
33
34        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
35        setBounds(100, 100, 450, 300);
36    }
37
38    public void addImmortal(Immortal immortal) {
39        immortals.add(immortal);
40
41        updateStatisticLabel();
42    }
43
44    public void removeImmortal(Immortal immortal) {
45        immortals.remove(immortal);
46
47        updateStatisticLabel();
48    }
49
50    public void attackImmortal(Immortal target) {
51        target.setHealth(target.getHealth() - DEFAULT_DAMAGE_VALUE);
52
53        updateStatisticLabel();
54    }
55
56    public void healImmortal(Immortal target) {
57        target.setHealth(Math.min(DEFAULT_IMMORTAL_HEALTH, target.getHealth() + DEFAULT_DAMAGE_VALUE));
58
59        updateStatisticLabel();
60    }
61
62    private void updateStatisticLabel() {
63        statisticLabel.setText("Immortals total health: " + immortals.stream().map(Immortal::getHealth).reduce(Integer::sum).orElse(0));
64    }
65
66    public static void main(String[] args) {
67        SwingUtilities.invokeLater(() -> {
68            new ControlFrame().setVisible(true);
69        });
70    }
71}

```

A screenshot of Microsoft Visual Studio Code (VS Code) interface. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and ControlFrame.java - ARSW\_Lab03 - Visual Studio Code. The left sidebar shows the Explorer (with ARSW\_Lab03 selected), Search, and Maven sections. The main editor area displays 'ControlFrame.java' with code related to Java Swing and immortal health. A terminal window at the bottom shows PowerShell commands being run, including navigating to the project directory and executing Java code. A status bar at the bottom indicates the file is 66 columns wide, 8 spaces, and 1 page long, with a Java icon and a date/time stamp.

**Respuesta:** Al ejecutar la aplicación se puede observar que no se está cumpliendo el invariante porque el resultado de la sumatoria no corresponde a la cantidad de inmortales debido a que no hay una sincronización cuando se le quita y se le pone salud a un inmortal.

4. Una primera hipótesis para que se presente la condición de carrera para dicha función (`pause and check`), es que el programa consulta la lista cuyos valores va a imprimir, a la vez que otros hilos modifican sus valores. Para corregir esto, haga lo que sea necesario para que efectivamente, antes de imprimir los resultados actuales, se pausen todos los demás hilos. Adicionalmente, implemente la opción ‘resume’.

➤ Creamos la clase Semaforo la cual nos permitirá conocer en qué estado se encuentra el hilo.

Creemos la clase semáforo la cual nos permitirá conocer en qué estado se encuentra el hilo.

The screenshot shows the Eclipse IDE interface. The title bar says "ControlFrame.java 9+, M". The left sidebar shows project files like "src", "main", "java", "arst", "arw", and "highlan...". The central editor area contains the following Java code:

```
src > main > java > edu > edc > arsw > highlandersim > Semaforo.java ...  
1 package edu.edc.arsw.highlandersim;  
2  
3 public class Semaforo {  
4  
5     private boolean bandera;  
6  
7     public synchronized boolean getBandera(){  
8         return bandera;  
9     }  
10  
11    public synchronized void setBandera(boolean bandera){  
12        this.bandera = bandera;  
13    }  
14  
15 }  
16
```

The bottom status bar shows "Lector de pantalla optimizado" and "Lin. 16, col. 1".

➤ Modificación de la clase Immortal.

```

    package edu.eci.arsw.highlandersim;

    import java.util.List;
    import java.util.Random;

    public class Immortal extends Thread {

        private ImmortalUpdateReportCallback updateCallback=null;

        private int health;

        private int defaultDamageValue;

        private final List<Immortal> immortalsPopulation;

        private final String name;

        private final Random r = new Random(System.currentTimeMillis());

        private Semaforo semaforo;

        public Immortal(String name, List<Immortal> immortalsPopulation, int health, int defaultDamageValue, ImmortalUpdateRepo
super(name);
this.updateCallback=ucb;
this.name = name;
this.immortalsPopulation = immortalsPopulation;
this.health = health;
this.defaultDamageValue=defaultDamageValue;
this.semaforo = semaforo;
    }

```

```

    public void run() {
        while(true) {
            Immortal im;

            if(!semaforo.getBandera()){
                synchronized(semaforo){
                    try {
                        semaforo.wait();
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }

            int myIndex = immortalsPopulation.indexOf(this);

            int nextFighterIndex = r.nextInt(immortalsPopulation.size());

            //avoid self-fight
            if (nextFighterIndex == myIndex) {
                nextFighterIndex = ((nextFighterIndex + 1) % immortalsPopulation.size());
            }

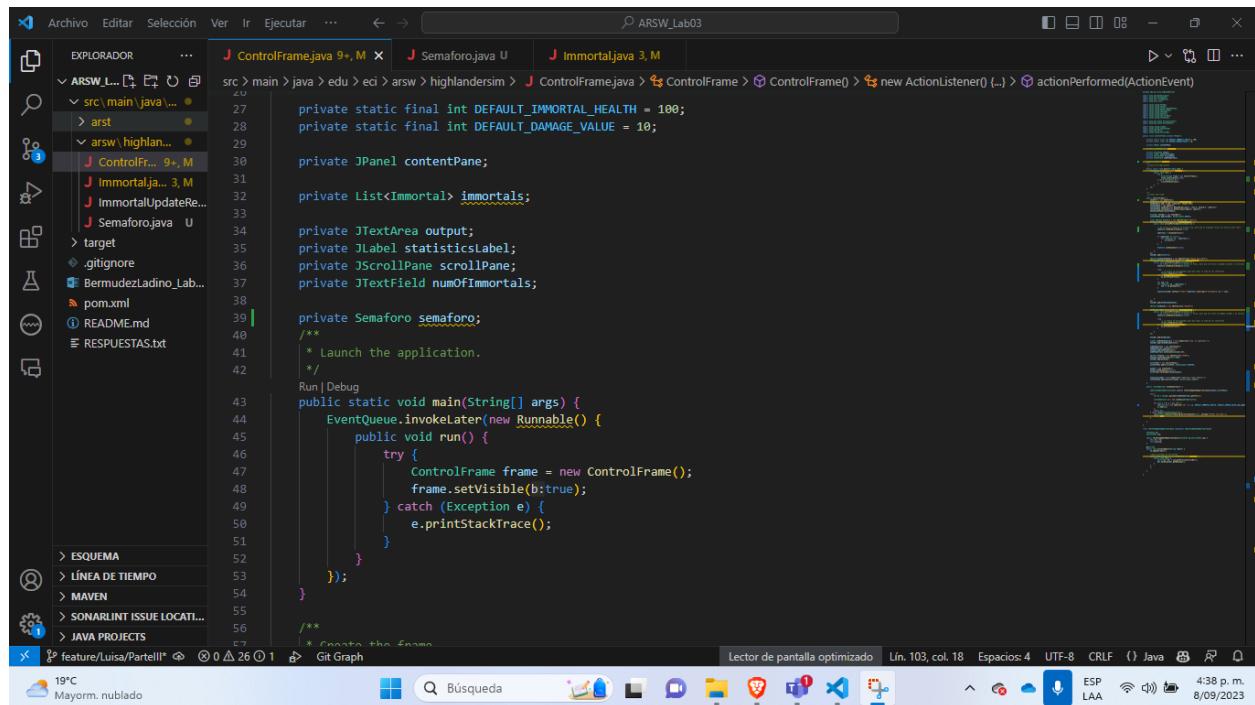
            im = immortalsPopulation.get(nextFighterIndex);

            this.fight(im);

            try {
                Thread.sleep(millis:1);
            } catch (InterruptedException e) {

```

➤ Modificación de la clase ControlFrame



```

src/main/java/edu/eci/arsw/highlandersim/ControlFrame.java
private static final int DEFAULT_IMMORTAL_HEALTH = 100;
private static final int DEFAULT_DAMAGE_VALUE = 10;

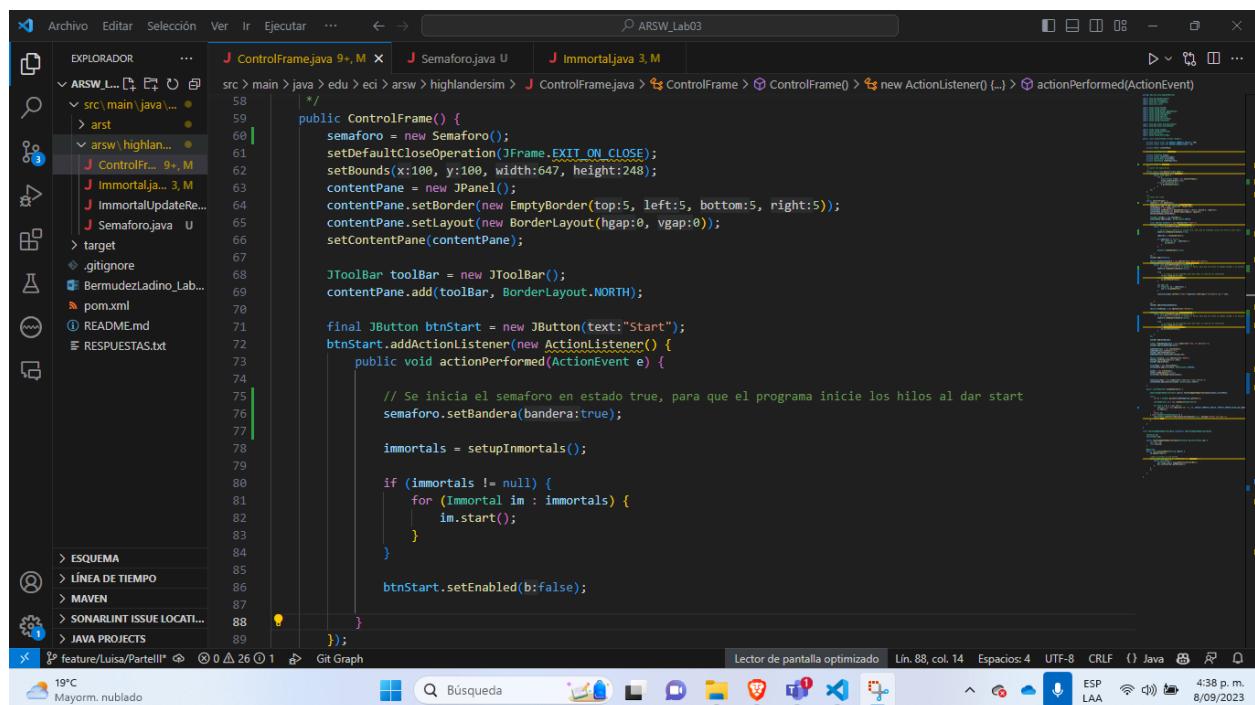
private JPanel contentPane;

private List<Immortal> immortals;

private JTextArea output;
private JLabel statisticsLabel;
private JScrollPane scrollPane;
private JTextField numOfImmortals;

private Semaforo semaforo;
/*
 * Launch the application.
 */
public static void main(String[] args) {
    EventQueue.invokeLater(new Runnable() {
        public void run() {
            try {
                ControlFrame frame = new ControlFrame();
                frame.setVisible(true);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });
}
/*
 * Create the frame.
 */

```



```

src/main/java/edu/eci/arsw/highlandersim/ControlFrame.java
/*
 * Launch the application.
 */
public static void main(String[] args) {
    EventQueue.invokeLater(new Runnable() {
        public void run() {
            try {
                ControlFrame frame = new ControlFrame();
                frame.setVisible(true);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });
}

/*
 * Create the frame.
 */
public ControlFrame() {
    semaforo = new Semaforo();
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100, 100, 647, 248);
    contentPane = new JPanel();
    contentPane.setLayout(new EmptyBorder(5, 5, 5, 5));
    contentPane.setBorder(new BorderLayout());
    contentPane.setLayout(new BorderLayout());
    contentPane.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    contentPane.setLayout(new BorderLayout());
    contentPane.setContentPane(contentPane);

    JToolBar toolBar = new JToolBar();
    contentPane.add(toolBar, BorderLayout.NORTH);

    final JButton btnStart = new JButton(text:"Start");
    btnStart.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            // Se inicia el semáforo en estado true, para que el programa inicie los hilos al dar start
            semaforo.setBandera(bandera:true);

            immortals = setupImmortals();

            if (immortals != null) {
                for (Immortal im : immortals) {
                    im.start();
                }
            }

            btnStart.setEnabled(b:false);
        }
    });
}
```

ARSW\_Lab03

```
src > main > java > edu > eci > arsw > highlandersim > J ControlFrame.java > ControlFrame > ControlFrame()
```

```
86     btnStart.setEnabled(false);
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }
101 }
102 }
103 }
104 }
105 }
106 }
107 }
108 }
109 }
110 }
111 }
112 }
113 }
114 }
115 }});
116 });
117 JButton btnStart = new JButton("Start");
118 btnStart.addActionListener(new ActionListener() {
119     public void actionPerformed(ActionEvent e) {
120         // Se cambia el estado de la bandera a true
121         synchronized(semáforo){
122             semáforo.setBandera(true);
123             // Se despiertan todos los hilos
124             semáforo.notifyAll();
125         }
126     }
127 }
128 }
129 }
130 }
131 }
132 }
133 }
134 }
135 }
136 });
137 }
138 }
139 }
140 }
141 }
142 }
143 }
```

Lector de pantalla optimizado Lín. 116, col. 39 Espacios 4 UTF-8 CRLF () Java 🌐 4:39 p. m. 8/09/2023

ARSW\_Lab03

```
Mensaje (Ctrl+Enter...) ✓ Confirmación | ▾
```

```
src > main > java > edu > eci > arsw > highlandersim > J ControlFrame.java > ControlFrame > new ActionListener() {} > actionPerformed(ActionEvent)
```

```
117 JButton btnResume = new JButton("Resume");
118 btnResume.addActionListener(new ActionListener() {
119     public void actionPerformed(ActionEvent e) {
120         // se accede al monitor de semáforo sincronizando los hilos, se cambia el estado de la bandera a true
121         synchronized(semáforo){
122             semáforo.setBandera(true);
123             // Se despiertan todos los hilos
124             semáforo.notifyAll();
125         }
126     }
127 }
128 }
129 }
130 }
131 }
132 }
133 }
134 }
135 }
136 });
137 }
138 }
139 }
140 }
141 }
142 }
143 }
```

SALIDA CONSOLA DE DÉPURAÇÃO TERMINAL 29

```
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARSW_Lab03>
```

Lector de pantalla optimizado Lín. 136, col. 14 Espacios 4 UTF-8 CRLF () Java 🌐 9:39 p. m. 8/09/2023

```

src > main > java > edu > eci > arsw > highlandersim > J ControlFrame.java > ControlFrame > ControlFrame > setupImmortals()

scrollPane = new JScrollPane();
contentPane.add(scrollPane, BorderLayout.CENTER);

output = new JTextArea();
output.setEditable(false);
scrollPane.setViewportView(output);

statisticsLabel = new JLabel(text:"Immortals total health:");
contentPane.add(statisticsLabel, BorderLayout.SOUTH);

}

public List<Immortal> setupImmortals() {
    ImmortalUpdateReportCallback ucb=new TextAreaUpdateReportCallback(output,scrollPane);

    try {
        int ni = Integer.parseInt(numOfImmortals.getText());
        List<Immortal> il = new LinkedList<Immortal>();

        for (int i = 0; i < ni; i++) {
            Immortal ii = new Immortal("im" + i, il, DEFAULT_IMMORTAL_HEALTH, DEFAULT_DAMAGE_VALUE, ucb, semaforo);
            il.add(ii);
        }
        return il;
    } catch (NumberFormatException e) {
        JOptionPane.showConfirmDialog(parentComponent,null, message:"Número inválido.");
        return null;
    }
}

```

19°C Mayorm. nublado Lector de pantalla optimizado Lín. 178, col. 25 Espacios: 4 UTF-8 CRLF Java 4:40 p. m. 8/09/2023

- Verifique nuevamente el funcionamiento (haga clic muchas veces en el botón). Se cumple o no el invariante?

```

src > main > java > edu > eci > arsw > highlandersim > J ControlFrame.java > ControlFrame > ControlFrame > new ActionListener() {} > actionPerformed(ActionEvent)
    sumar la vida de los inmortales
    num. of immortals:4
    STOP
    Start Pause and check
    Fight im2[870]vs im0[270]
    Fight im3[10]vs im2[860]
    Fight im1[200]vs im3[0]
    Fight im0[270]vs im1[190]
    Fight im3[10]vs im0[260]
    Fight im2[880]vs im0[260]
    Bern
    [im0[270],im1[190],im2[880],im3[0]]
    pom Health sum:1340
    toString()+"<br>Health sum:"+ sum);
    ...
    JButton btnResume = new JButton(text:"Resume");

```

SALIDA CONSOLA DE DEPURACIÓN TERMINAL 27

Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. <https://aka.ms/PSWindows>

PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARSM\_Lab03> & 'C:\Program Files\Java\jdk-17.0.2\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARSM\_Lab03\target\classes' 'edu.eci.arsw.highlandersim.ControlFrame'

powershell Run: ControlFrame

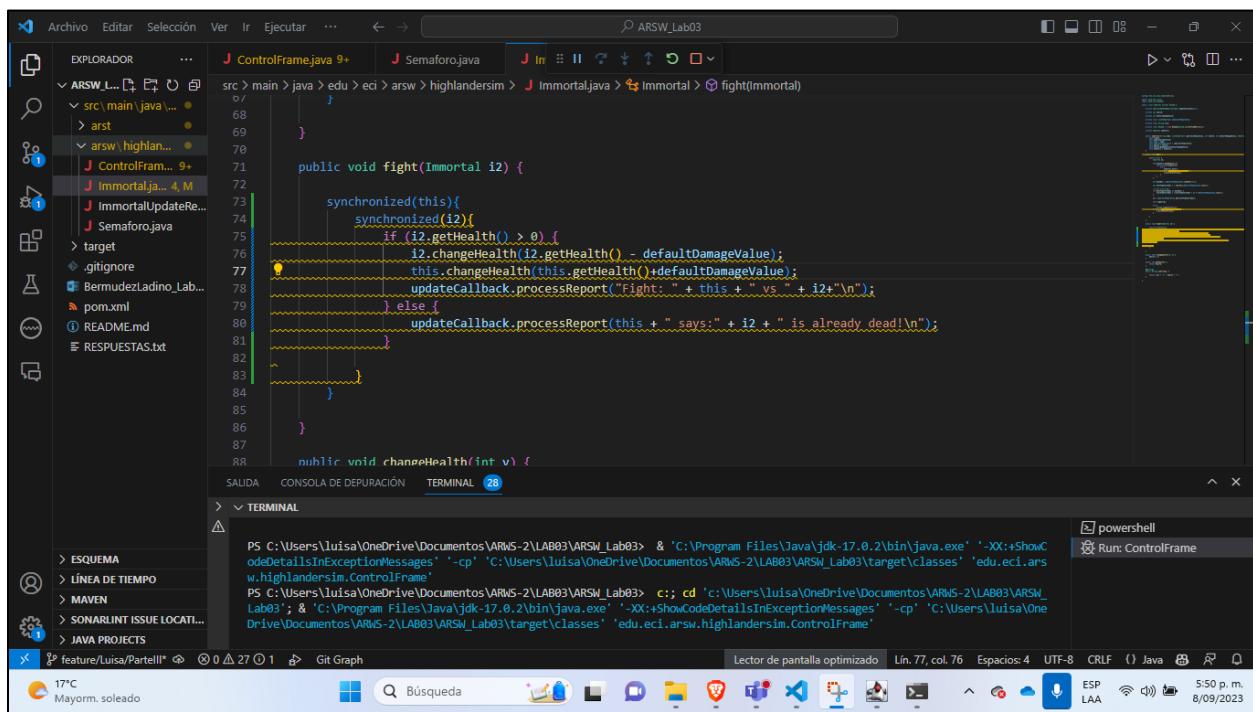
19°C Mayorm. nublado Lector de pantalla optimizado Lín. 103, col. 18 Espacios: 4 UTF-8 CRLF Java 4:33 p. m. 8/09/2023

**Respuesta:** No cumple con la invariante porque se sincronizaron los hilos mas no el acceso a la variable compartida.

6. Identifique posibles regiones críticas en lo que respecta a la pelea de los inmortales. Implemente una estrategia de bloqueo que evite las condiciones de carrera. Recuerde que, si usted requiere usar dos o más ‘locks’ simultáneamente, puede usar bloques sincronizados anidados:

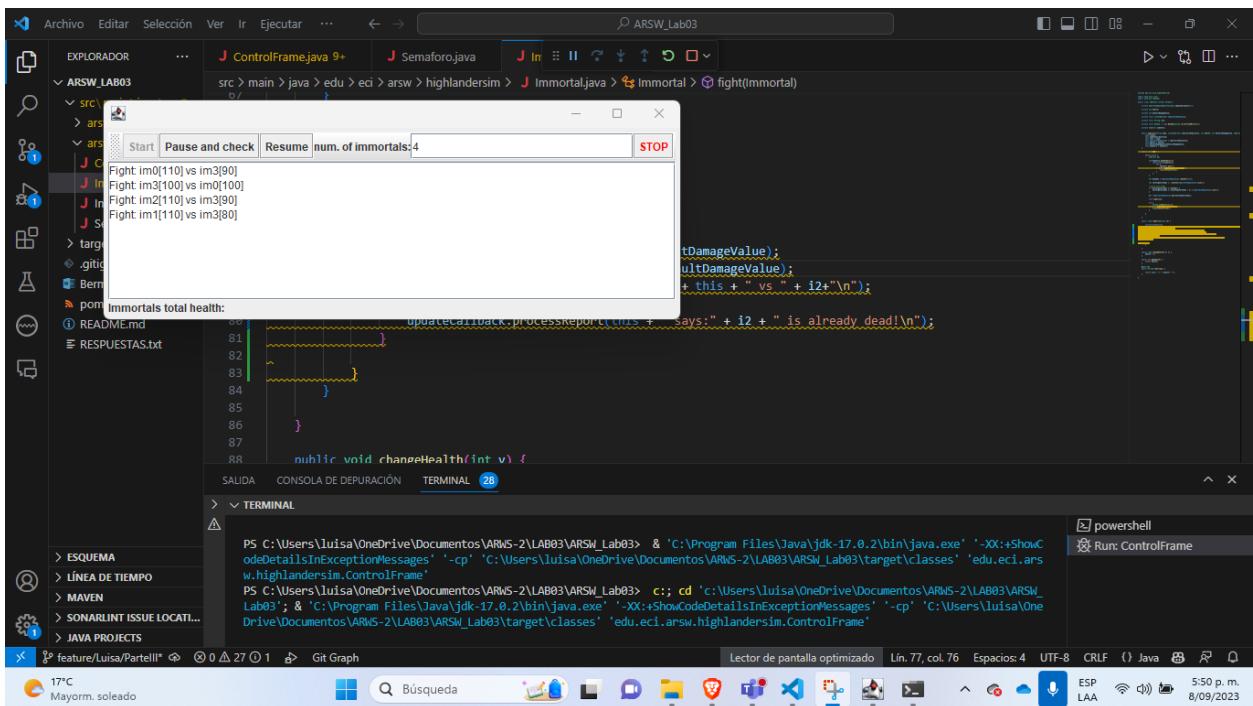
```
synchronized(locka){
    synchronized(lockb){
        ...
    }
}
```

## ➤ Modificación de la clase Immortal



7. Tras implementar su estrategia, ponga a correr su programa, y ponga atención a si éste se llega a detener. Si es así, use los programas jps y jstack para identificar por qué el programa se detuvo.

**Respuesta:** Como podemos observar en la siguiente imagen al ejecutar el programa este se detuvo



- **jps:** es una herramienta de línea de comandos que se incluye en la distribución de Java y se utiliza para listar los procesos Java en ejecución en un sistema. Esta herramienta es útil para obtener información sobre los procesos de Java que se están ejecutando en una máquina, incluyendo sus identificadores de proceso (PID) y los nombres de las clases principales asociadas.
- **jstack:** es una herramienta de línea de comandos que se incluye en el Kit de Desarrollo de Java (JDK) y se utiliza para obtener trazas de hilos (stack traces) de una máquina virtual de Java (JVM). Estas trazas de hilos son útiles para el diagnóstico y la resolución de problemas relacionados con la concurrencia, bloqueos, cuellos de botella y problemas de rendimiento en aplicaciones Java.
  - ✓ **jstack -l <PID>:** se utiliza para obtener una traza de hilos detallada de una máquina virtual de Java (JVM) en ejecución.
  - ✓ **-l:** indica que se debe incluir información adicional, como bloqueos y propietarios de los monitores bloqueados.

La salida de este comando mostrará información detallada sobre los hilos en ejecución en esa JVM, incluyendo la pila de llamadas de cada hilo y cualquier información relacionada con bloqueos y monitores.

Esta información es útil para diagnosticar problemas de bloqueo, condiciones de carrera y problemas de rendimiento en una aplicación Java que se esté ejecutando en esa JVM específica.

```

C:\Users\luisa>jps
17008 org.eclipse.equinox.launcher_1.6.500.v20230717-2134.jar
14900 sonarlint-ls.jar
20148 ControlFrame
17516 Jps

C:\Users\luisa>jstack -l 20148
2023-09-08 17:47:56
Full thread dump OpenJDK 64-Bit Server VM (17.0.2+8-86 mixed mode, sharing):

Threads class SMR info:

    ...


"Reference Handler" #2 daemon prio=10 os_prio=2 cpu=0.00ms elapsed=1489.13s tid=0x000001ab871e9b60 nid=0x3f3c waiting on
condition [0x00000a95ad4ff000]
    java.lang.Thread.State: RUNNABLE
        at java.lang.ref.Reference.waitForReferencePendingList(java.base@17.0.2/Native Method)
        at java.lang.ref.Reference$ReferencePendingList.run(java.base@17.0.2/Reference.java:253)
        at java.lang.ref.Reference$ReferenceHandler.run(java.base@17.0.2/Reference.java:215)

Locked ownable synchronizers:
    - None

8 elementos

```

Como podemos observar en la siguiente imagen obtuvimos información de los deadlock y los hilos.

```

JNI global refs: 101, weak refs: 14

Found one Java-level deadlock:
=====
"im0":
    waiting to lock monitor 0x000001ab8a811750 (object 0x00000000911ba640, a edu.eci.arsw.hIGHLANDERSIM.Immortal),
    which is held by "im3"

"im3":
    waiting to lock monitor 0x000001ab87e2f3e0 (object 0x00000000911b9248, a edu.eci.arsw.hIGHLANDERSIM.Immortal),
    which is held by "im0"

Java stack information for the threads listed above:
=====
"im0":
    at edu.eci.arsw.hIGHLANDERSIM.Immortal.fight(Immortal.java:75)
    - waiting to lock <0x00000000911ba640> (a edu.eci.arsw.hIGHLANDERSIM.Immortal)
    - locked <0x00000000911b9248> (a edu.eci.arsw.hIGHLANDERSIM.Immortal)
    at edu.eci.arsw.hIGHLANDERSIM.Immortal.run(Immortal.java:59)

"im3":
    at edu.eci.arsw.hIGHLANDERSIM.Immortal.fight(Immortal.java:75)
    - waiting to lock <0x00000000911b9248> (a edu.eci.arsw.hIGHLANDERSIM.Immortal)
    - locked <0x00000000911ba640> (a edu.eci.arsw.hIGHLANDERSIM.Immortal)
    at edu.eci.arsw.hIGHLANDERSIM.Immortal.run(Immortal.java:59)

Found 1 deadlock.

C:\Users\luisa>

```

8. Plantee una estrategia para corregir el problema antes identificado (puede revisar de nuevo las páginas 206 y 207 de *Java Concurrency in Practice*).

### ➤ Modificación de la clase Immortal

```

public class Immortal extends Thread {
    private ImmortalUpdateReportCallback updateCallback=null;
    private AtomicInteger health;
    private int defaultDamageValue;
    private final List<Immortal> immortalsPopulation;
    private final String name;
    private final Random r = new Random(System.currentTimeMillis());
    private Semaforo semaforo;

    public Immortal(String name, List<Immortal> immortalsPopulation, int health, int defaultDamageValue, ImmortalUpdateReportCallback ucb ,Semaforo semaforo) {
        super(name);
        this.updateCallback=ucb;
        this.name = name;
        this.immortalsPopulation = immortalsPopulation;
        this.health = new AtomicInteger(health);
        this.defaultDamageValue=defaultDamageValue;
        this.semaforo = semaforo;
    }

    public void run() {
        while (true) {
            Immortal im;
            if(!semaforo.getBandera()){
                synchronized(semaforo){
                    try {
                        semaforo.wait();
                    } catch (InterruptedException e) {
                        continue;
                    }
                }
            }
            synchronized(im){
                if(i2.getHealth() > 0) {
                    i2.changeHealth(-defaultDamageValue);
                    this.changeHealth(defaultDamageValue);
                    updateCallback.processReport("Fight: " + this + " vs " + i2+"\n");
                } else {
                    updateCallback.processReport(this + " says:" + i2 + " is already dead!\n");
                }
            }
        }
    }

    public void fight(Immortal i2) {
        synchronized(this){
            synchronized(i2){
                if (i2.getHealth() > 0) {
                    i2.changeHealth(-defaultDamageValue);
                    this.changeHealth(defaultDamageValue);
                    updateCallback.processReport("Fight: " + this + " vs " + i2+"\n");
                } else {
                    updateCallback.processReport(this + " says:" + i2 + " is already dead!\n");
                }
            }
        }
    }

    public void changeHealth(int v) {
        //Obtiene el valor actual en memoria (variable), lo setea y lo vuelve a agregar
        health.addAndGet(v);
    }

    public int getHealth() {
        //Permite retornar el AtomicInteger como un entero
        return health.intValue();
    }

    @Override
    public String toString() {
        return name + "[" + health + "]";
    }
}

```

```

public class Immortal extends Thread {
    private ImmortalUpdateReportCallback updateCallback=null;
    private AtomicInteger health;
    private int defaultDamageValue;
    private final List<Immortal> immortalsPopulation;
    private final String name;
    private final Random r = new Random(System.currentTimeMillis());
    private Semaforo semaforo;

    public Immortal(String name, List<Immortal> immortalsPopulation, int health, int defaultDamageValue, ImmortalUpdateReportCallback ucb ,Semaforo semaforo) {
        super(name);
        this.updateCallback=ucb;
        this.name = name;
        this.immortalsPopulation = immortalsPopulation;
        this.health = new AtomicInteger(health);
        this.defaultDamageValue=defaultDamageValue;
        this.semaforo = semaforo;
    }

    public void run() {
        while (true) {
            Immortal im;
            if(!semaforo.getBandera()){
                synchronized(semaforo){
                    try {
                        semaforo.wait();
                    } catch (InterruptedException e) {
                        continue;
                    }
                }
            }
            if (i2.getHealth() > 0) {
                i2.changeHealth(-defaultDamageValue);
                this.changeHealth(defaultDamageValue);
                updateCallback.processReport("Fight: " + this + " vs " + i2+"\n");
            } else {
                updateCallback.processReport(this + " says:" + i2 + " is already dead!\n");
            }
        }
    }

    public void fight(Immortal i2) {
        synchronized(this){
            if (i2.getHealth() > 0) {
                i2.changeHealth(-defaultDamageValue);
                this.changeHealth(defaultDamageValue);
                updateCallback.processReport("Fight: " + this + " vs " + i2+"\n");
            } else {
                updateCallback.processReport(this + " says:" + i2 + " is already dead!\n");
            }
        }
    }

    public void changeHealth(int v) {
        //Obtiene el valor actual en memoria (variable), lo setea y lo vuelve a agregar
        health.addAndGet(v);
    }

    public int getHealth() {
        //Permite retornar el AtomicInteger como un entero
        return health.intValue();
    }

    @Override
    public String toString() {
        return name + "[" + health + "]";
    }
}

```

9. Una vez corregido el problema, rectifique que el programa siga funcionando de manera consistente cuando se ejecutan 100, 1000 o 10000 inmortales. Si en estos casos grandes se empieza a incumplir de nuevo el invariante, debe analizar lo realizado en el paso 4.

## ➤ 100 inmortales

```

    Bienvenido J Immortal.java 5.M J ControlFrame.java 9+ x
src > main > java > edu > eci > arsw > highlanders > J ControlFrame.java > ControlFrame
btmTop.setSetForeground(Color.RED);
toolBar.add(bttnStop);

public class ControlFrame extends JFrame {
    ...
    public void run() {
        ...
        while(true) {
            ...
            if(num_of_inmortals == 100) {
                ...
            }
        }
    }
}
  
```

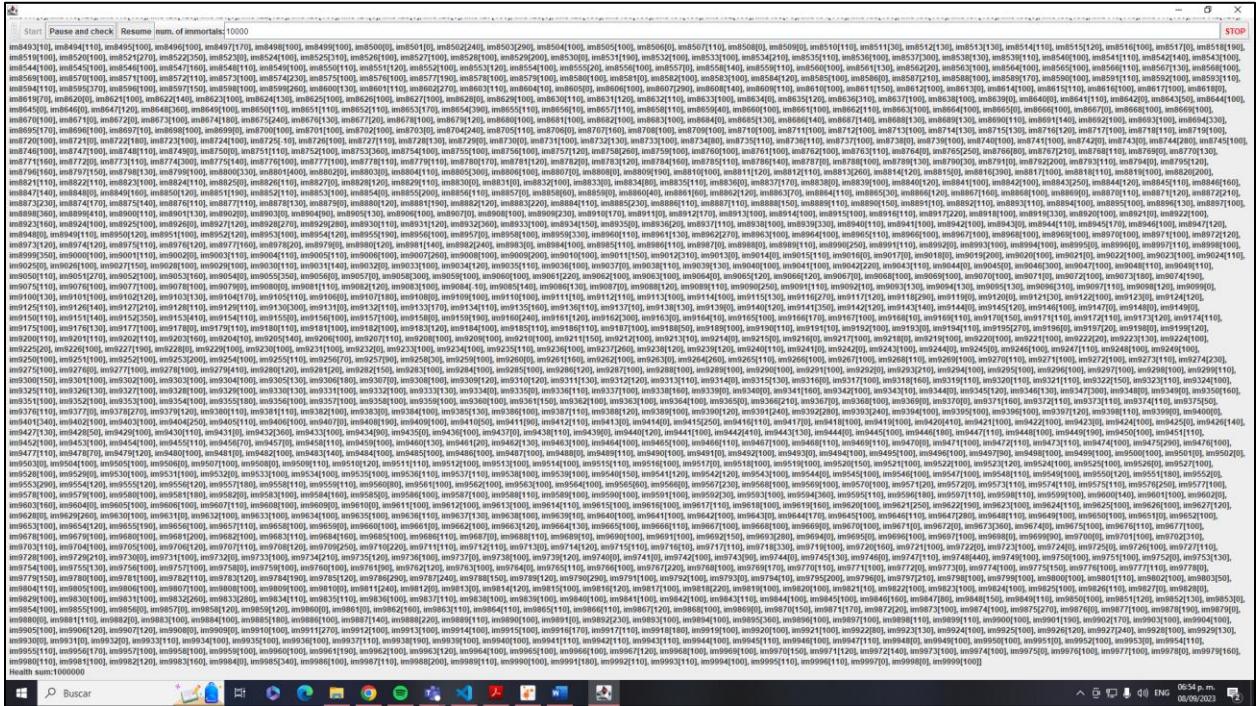
## ➤ 1000 inmortales

```

    Bienvenido J Immortal.java 5.M J ControlFrame.java 9+ x
src > main > java > edu > eci > arsw > highlanders > J ControlFrame.java > ControlFrame
btmTop.setSetForeground(Color.RED);
toolBar.add(bttnStop);

public class ControlFrame extends JFrame {
    ...
    public void run() {
        ...
        while(true) {
            ...
            if(num_of_inmortals == 1000) {
                ...
            }
        }
    }
}
  
```

## ➤ 10000 inmortales



10. Un elemento molesto para la simulación es que en cierto punto de la misma hay pocos 'inmortales' vivos realizando peleas fallidas con 'inmortales' ya muertos. Es necesario ir suprimiendo los inmortales muertos de la simulación a medida que van muriendo. Para esto:

- Analizando el esquema de funcionamiento de la simulación, esto podría crear una condición de carrera? Implemente la funcionalidad, ejecute la simulación y observe qué problema se presenta cuando hay muchos 'inmortales' en la misma. Escriba sus conclusiones al respecto en el archivo RESPUESTAS.txt.

**Respuesta:** Si se crea una condición carrera.

Cuando alguien pide un inmortal de la lista este podrá obtenerlo antes de que este sea removido, pero también puede existir el caso en el que soliciten el inmortal y este ya haya sido eliminado de la lista, por lo tanto, no lo alcanzaría a obtener.

- Corrija el problema anterior **SIN hacer uso de sincronización**, pues volver secuencial el acceso a la lista compartida de inmortales haría extremadamente lenta la simulación.

## ➤ Modificación de la clase Immortal

```

src > main > java > edu > edc > arsw > highlandersim > J Immortal.java 5 M x J ControlFrame.java 9+ J Semaforo.java J ImmortalUpdateReportCallback.java

public class Immortal extends Thread {
    private ImmortalUpdateReportCallback updateCallback=null;
    private AtomicInteger health;
    private int defaultDamageValue;
    private final List<Immortal> immortalsPopulation;
    private final String name;
    private final Random r = new Random(System.currentTimeMillis());
    private Semaphore semaforo;
    private boolean estoyVivo;

    public Immortal(String name, List<Immortal> immortalsPopulation, int health, int defaultDamageValue, ImmortalUpdateReportCallback ucb ,Semaphore semaforo) {
        super(name);
        this.updateCallback=ucb;
        this.name = name;
        this.immortalsPopulation = immortalsPopulation;
        this.health = new AtomicInteger(health);
        this.defaultDamageValue=defaultDamageValue;
        this.semaforo = semaforo;
        estoyVivo = true;
    }

    public void run() {
        while (estoyVivo) {
            Immortal im;
            if(!semaforo.getBandera()){
                synchronized(semaforo){
                    try {
                        semaforo.wait();
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }
            int myIndex = -1;
            int nextFighterIndex = 0;
            synchronized(immortalsPopulation){
                myIndex = immortalsPopulation.indexOf(this);
                if(immortalsPopulation.isEmpty()){
                    nextFighterIndex = r.nextInt(immortalsPopulation.size());
                }
            }
            //avoid self-fight
            if (nextFighterIndex == myIndex) {
                nextFighterIndex = (nextFighterIndex + 1) % immortalsPopulation.size();
            }
        }
        if(nextFighterIndex != myIndex){
            im = immortalsPopulation.get(nextFighterIndex);
            this.Fight(im);
            try {
                Thread.sleep(millis:1);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

```

```

src > main > java > edu > edc > arsw > highlandersim > J Immortal.java 5 M x J ControlFrame.java 9+ J Semaforo.java J ImmortalUpdateReportCallback.java

public void run() {
    while (estoyVivo) {
        Immortal im;
        if(!semaforo.getBandera()){
            synchronized(semaforo){
                try {
                    semaforo.wait();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
        int myIndex = -1;
        int nextFighterIndex = 0;
        synchronized(immortalsPopulation){
            myIndex = immortalsPopulation.indexOf(this);
            if(immortalsPopulation.isEmpty()){
                nextFighterIndex = r.nextInt(immortalsPopulation.size());
            }
        }
        //avoid self-fight
        if (nextFighterIndex == myIndex) {
            nextFighterIndex = (nextFighterIndex + 1) % immortalsPopulation.size();
        }
    }
    if(nextFighterIndex != myIndex){
        im = immortalsPopulation.get(nextFighterIndex);
        this.Fight(im);
        try {
            Thread.sleep(millis:1);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}

```

```
src > main > java > edu > eci > arsw > highlandersim > J Immortal.java 5. M x J ControlFrame.java 9+ J Semaforo.java J ImmortalUpdateReportCallback.java

if(nextFighterIndex != myIndex){

    im = immortalsPopulation.get(nextFighterIndex);

    this.Fight(im);

    try {
        Thread.sleep(millis);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}

public void fight(Immortal i2) {

    synchronized(immortalsPopulation){

        if (i2.getHealth() > 0 && estoyVivo) {
            i2.changeHealth(-defaultDamageValue);
            this.changeHealth(defaultDamageValue);
            updateCallback.processReport("Fight: " + this + " vs " + i2+"\n");
        } else if (estoyVivo){
            updateCallback.processReport(this + " says:" + i2 + " is already dead!\n");
            if(immortalsPopulation.size() > 1){
                //Se elimina el immortal muerto de la lista
                immortalsPopulation.remove(i2);
            }
        }else{
            if(immortalsPopulation.size() > 1){
                //Se elimina el immortal muerto de la lista
                immortalsPopulation.remove(this);
            }
        }
    }
}
```

```
src > main > java > edu > eci > arsw > highlandersim > J Immortal.java 5. M x J ControlFrame.java 9+ J Semaforo.java J ImmortalUpdateReportCallback.java

this.changeHealth(defaultDamageValue);
updateCallback.processReport("Fight: " + this + " vs " + i2+"\n");
} else if (estoyVivo){
    updateCallback.processReport(this + " says:" + i2 + " is already dead!\n");
    if(immortalsPopulation.size() > 1){
        //Se elimina el immortal muerto de la lista
        immortalsPopulation.remove(i2);
    }
} else{
    if(immortalsPopulation.size() > 1){
        //Se elimina el immortal muerto de la lista
        immortalsPopulation.remove(this);
    }
}

unknown, hace 7 días * Fuentes base ...

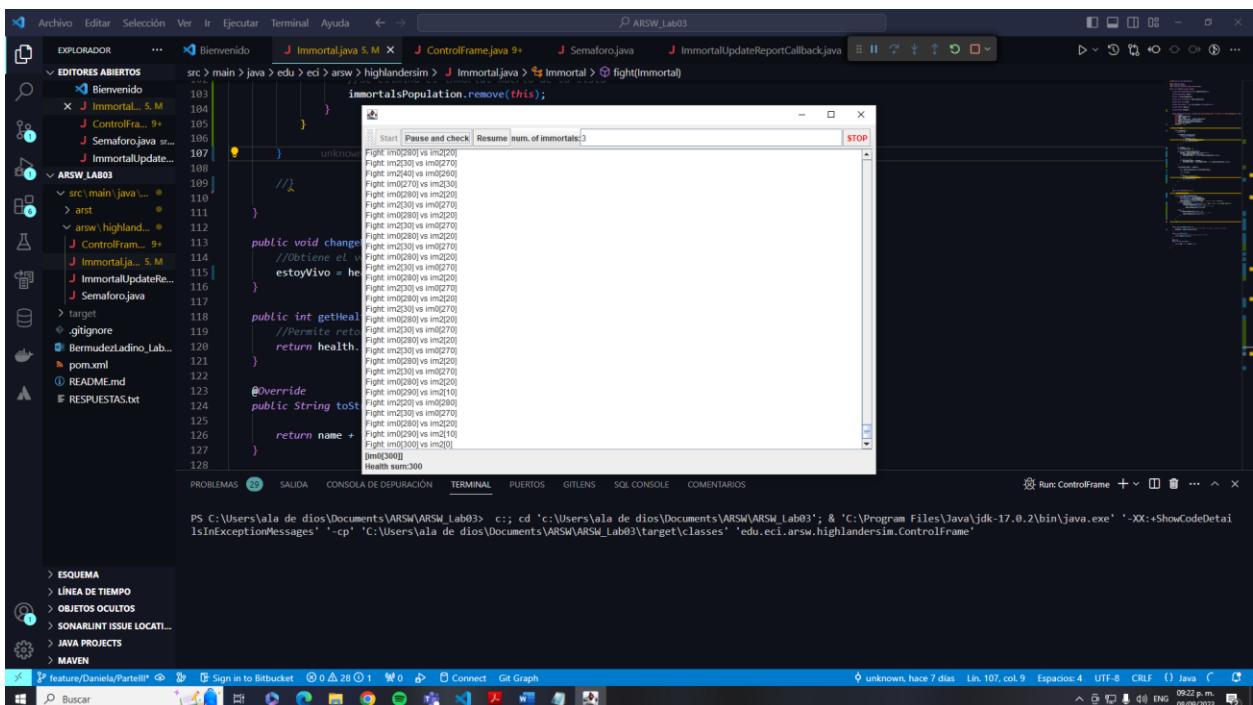
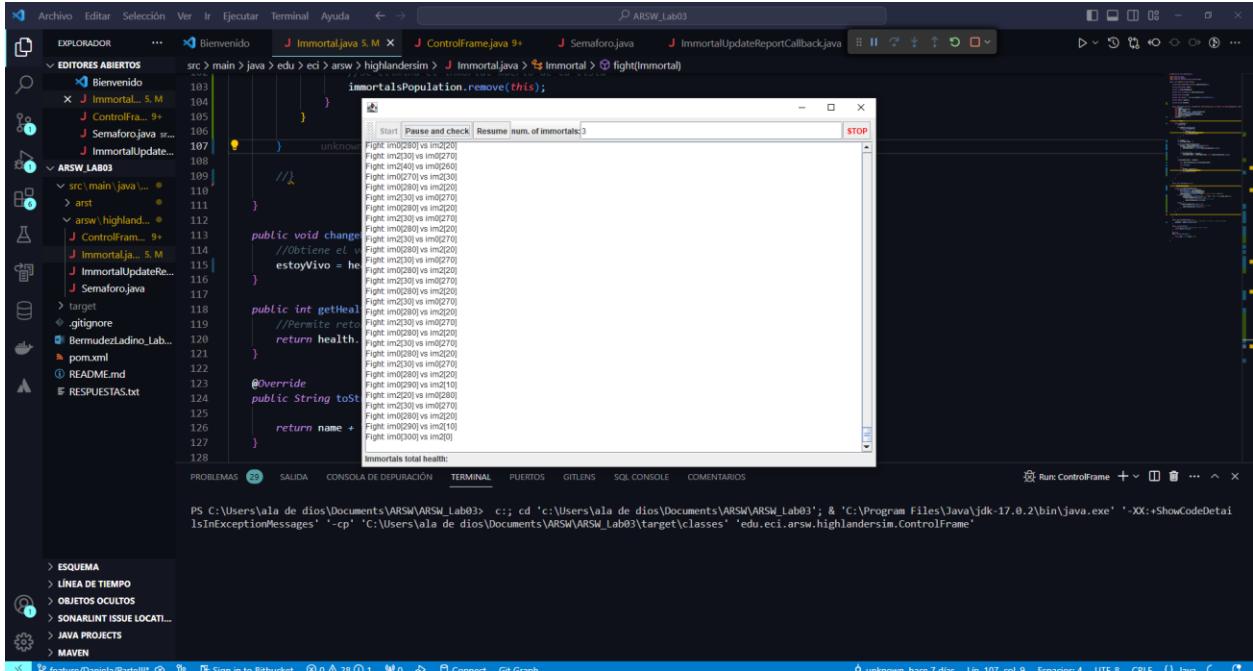
//]

public void changeHealth(int v) {
    //Obtiene el valor actual en memoria (variable), lo setea y lo vuelve a agregar
    estoyVivo = health.addAndGet(v)>0;
}

public int getHealth() {
    //Permite retornar el AtomicInteger como un entero
    return health.intValue();
}

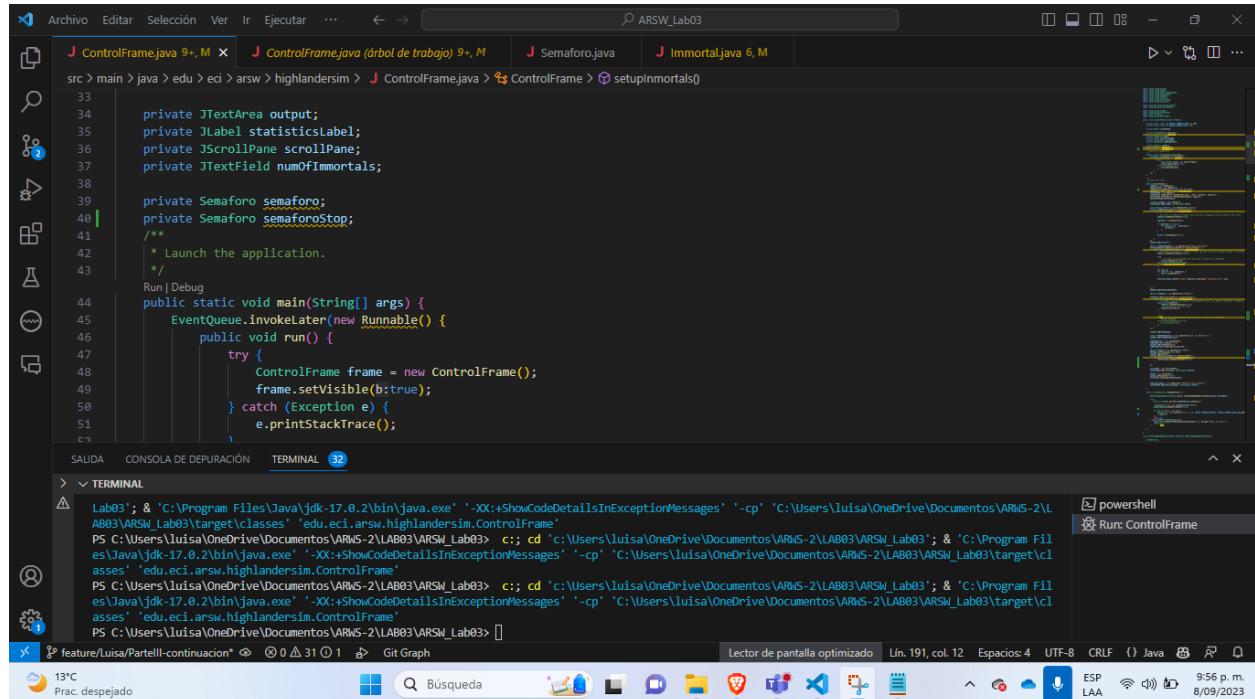
@Override
public String toString() {
    return name + "[" + health + "]";
}
```

## ➤ Resultados al ejecutar:



## 11. Para finalizar, implemente la opción STOP.

### ➤ Modificación clase ControlFrame



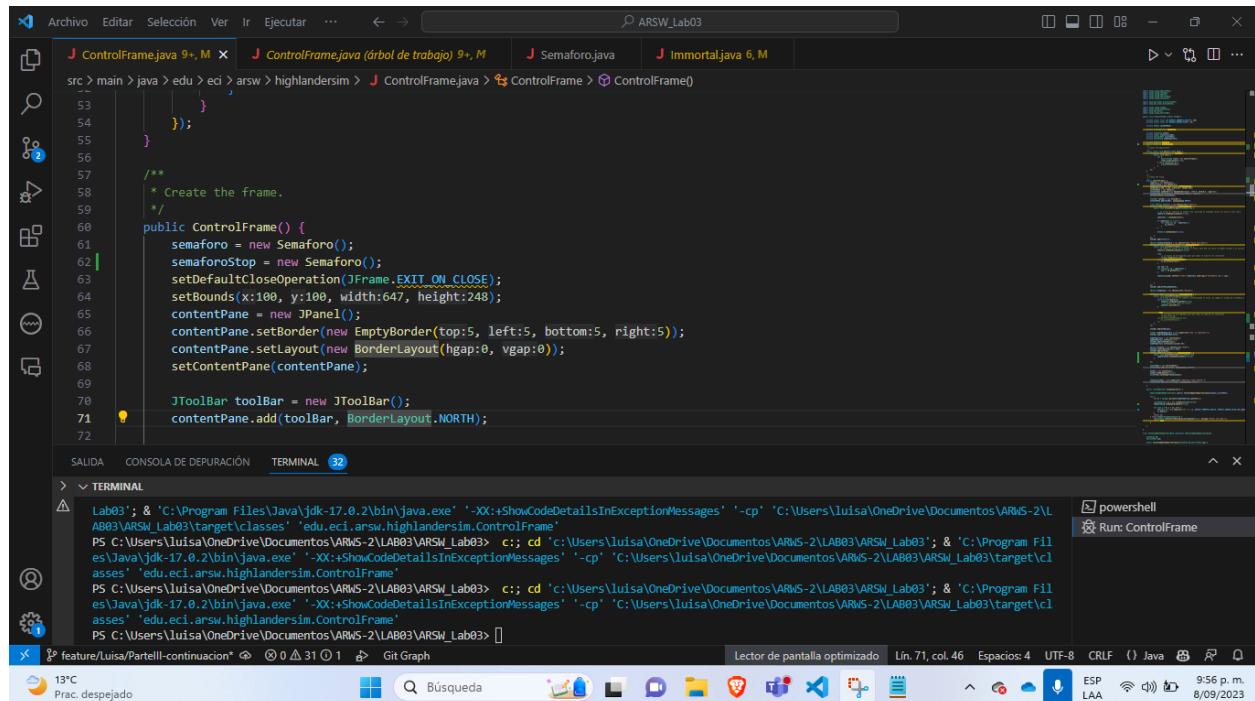
```
private JTextArea output;
private JLabel statisticsLabel;
private JScrollPane scrollPane;
private JTextField numOfImmortals;

private Semaforo semaforo;
private Semaforo semaforoStop;
/**
 * Launch the application.
 */
Run|Debug
public static void main(String[] args) {
    EventQueue.invokeLater(new Runnable() {
        public void run() {
            try {
                ControlFrame frame = new ControlFrame();
                frame.setVisible(true);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });
}
```

SALIDA CONSOLA DE DEPURACIÓN TERMINAL 

```
Lab03 & 'C:\Program Files\Java\jdk-17.0.2\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARWS_Lab03\target\classes' 'edu.eci.arsw.highlandersim.ControlFrame'
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARWS_Lab03> c:; cd 'c:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARWS_Lab03'; & 'C:\Program Files\Java\jdk-17.0.2\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARWS_Lab03\target\classes' 'edu.eci.arsw.highlandersim.ControlFrame'
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARWS_Lab03> c:; cd 'c:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARWS_Lab03'; & 'C:\Program Files\Java\jdk-17.0.2\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARWS_Lab03\target\classes' 'edu.eci.arsw.highlandersim.ControlFrame'
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARWS_Lab03> [REDACTED]
```

feature/Luisa/Partell-continuacion ⌂ 0 △ 31 ⓘ 1 ⌂ Git Graph Lector de pantalla optimizado Lín. 191, col. 12 Espacios: 4 UTF-8 CRLF ⓘ Java ⓘ 🔍 ⓘ ESP LAA ⓘ 9:56 p. m. 8/09/2023



```
    }
}

/**
 * Create the frame.
 */
public ControlFrame() {
    semaforo = new Semaforo();
    semaforoStop = new Semaforo();
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(x:100, y:100, width:647, height:248);
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(top:5, left:5, bottom:5, right:5));
    contentPane.setLayout(new BorderLayout(hgap:0, vgap:0));
    setContentPane(contentPane);

    JToolBar toolbar = new JToolBar();
    contentPane.add(toolbar, BorderLayout.NORTH);
```

SALIDA CONSOLA DE DEPURACIÓN TERMINAL 

```
Lab03 & 'C:\Program Files\Java\jdk-17.0.2\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARWS_Lab03\target\classes' 'edu.eci.arsw.highlandersim.ControlFrame'
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARWS_Lab03> c:; cd 'c:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARWS_Lab03'; & 'C:\Program Files\Java\jdk-17.0.2\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARWS_Lab03\target\classes' 'edu.eci.arsw.highlandersim.ControlFrame'
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARWS_Lab03> c:; cd 'c:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARWS_Lab03'; & 'C:\Program Files\Java\jdk-17.0.2\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARWS_Lab03\target\classes' 'edu.eci.arsw.highlandersim.ControlFrame'
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARWS_Lab03> [REDACTED]
```

feature/Luisa/Partell-continuacion ⌂ 0 △ 31 ⓘ 1 ⌂ Git Graph Lector de pantalla optimizado Lín. 71, col. 46 Espacios: 4 UTF-8 CRLF ⓘ Java ⓘ 🔍 ⓘ ESP LAA ⓘ 9:56 p. m. 8/09/2023

```
146 numOfImmortals = new JTextField();
147 numOfImmortals.setText("3");
148 toolbar.add(numOfImmortals);
149 numOfImmortals.setColumns(10);
150
151 JButton btnStop = new JButton(text:"STOP");
152 btnStop.setForeground(Color.RED);
153 toolbar.add(btnStop);
154 btnStop.addActionListener(new ActionListener(){
155     public void actionPerformed (ActionEvent e){
156         semaforoStop.setBandera(bandera:false);
157     }
158 });
159
160 scrollPane = new JScrollPane();
161 contentPane.add(scrollPane, BorderLayout.CENTER);
162
163 output = new JTextArea();
164 output.setEditable(b:false);
165
```

SALIDA CONSOLA DE DEPURACIÓN TERMINAL

```
Lab03' & 'C:\Program Files\Java\jdk-17.0.2\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARSW_Lab03\target\classes' 'edu.eci.arsw.hightlandersim.ControlFrame'
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARSW_Lab03> c:: cd 'c:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARSW_Lab03'; & 'C:\Program Files\Java\jdk-17.0.2\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARSW_Lab03\target\classes' 'edu.eci.arsw.hightlandersim.ControlFrame'
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARSW_Lab03> c:: cd 'c:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARSW_Lab03'; & 'C:\Program Files\Java\jdk-17.0.2\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARSW_Lab03\target\classes' 'edu.eci.arsw.hightlandersim.ControlFrame'
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARSW_Lab03> [REDACTED]
```

feature/luisa/Partell-continuacion\* ⊗ 0 △ 31 ⊕ 1 ⌛ Git Graph

13°C Prac. despejado

Búsqueda Lector de pantalla optimizado Lín. 164, col. 23 Espacios 4 UTF-8 CR LF ESP LAA 9:55 p. m. 8/09/2023

```
172 }
173
174 public List<Immortal> setupImmortals() {
175
176     ImmortalUpdateReportCallback ucb=new TextAreaUpdateReportCallback(output,scrollPane);
177
178     try {
179         int ni = Integer.parseInt(numOfImmortals.getText());
180
181         List<Immortal> il = new LinkedList<Immortal>();
182         semaforoStop.setBandera(bandera:true);
183
184         for (int i = 0; i < ni; i++) {
185             Immortal ii = new Immortal("im" + i, il, DEFAULT_IMMORTAL_HEALTH, DEFAULT_DAMAGE_VALUE, ucb,semaforo,semaforoStop);
186             il.add(ii);
187         }
188         return il;
189     } catch (NumberFormatException e) {
190         JOptionPane.showConfirmDialog(parentComponent:null, message:"Número inválido.");
191         return null;
192     }
193 }
```

SALIDA CONSOLA DE DEPURACIÓN TERMINAL

```
Lab03' & 'C:\Program Files\Java\jdk-17.0.2\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARSW_Lab03\target\classes' 'edu.eci.arsw.hightlandersim.ControlFrame'
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARSW_Lab03> c:: cd 'c:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARSW_Lab03'; & 'C:\Program Files\Java\jdk-17.0.2\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARSW_Lab03\target\classes' 'edu.eci.arsw.hightlandersim.ControlFrame'
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARSW_Lab03> c:: cd 'c:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARSW_Lab03'; & 'C:\Program Files\Java\jdk-17.0.2\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARSW_Lab03\target\classes' 'edu.eci.arsw.hightlandersim.ControlFrame'
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARSW_Lab03> [REDACTED]
```

feature/luisa/Partell-continuacion\* ⊗ 0 △ 31 ⊕ 1 ⌛ Git Graph

13°C Prac. despejado

Búsqueda Lector de pantalla optimizado Lín. 190, col. 37 Espacios 4 UTF-8 CR LF ESP LAA 9:57 p. m. 8/09/2023

## ➤ Modificación clase Immortal

The screenshot shows an IDE interface with several tabs open: ControlFrame.java, ControlFrame.java (árbol de trabajo), Semaforo.java, and Immortal.java. The Immortal.java tab is active, displaying the following code:

```

10
11     private final String name;
12
13     private final Random r = new Random(System.currentTimeMillis());
14
15     private Semaforo semaforo;
16
17     private Semaforo semaforoStop;
18
19     private boolean estoyVivo;
20
21
22     public Immortal(String name, List<Immortal> immortalsPopulation, int health, int defaultDamageValue, ImmortalUpdateReportCallback ucb ,Semafo
23
24
25
26
27
28
29
30
31
32
33
34
35
36

```

Below the code editor is a terminal window showing the command-line steps to build and run the application:

```

Lab03'; & 'C:\Program Files\Java\jdk-17.0.2\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARSW_Lab03\target\classes' 'edu.eci.arsw.highlandersim.ControlFrame'
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARSW_Lab03> c:; cd 'c:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARSW_Lab03'; & 'C:\Program Files\Java\jdk-17.0.2\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARSW_Lab03\target\cl
asses' 'edu.eci.arsw.highlandersim.ControlFrame'
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARSW_Lab03> c:; cd 'c:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARSW_Lab03'; & 'C:\Program Files\Java\jdk-17.0.2\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARSW_Lab03\target\cl
asses' 'edu.eci.arsw.highlandersim.ControlFrame'
PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARSW_Lab03> [REDACTED]

```

The terminal also shows the current weather and date.

This screenshot is nearly identical to the one above, showing the same code editor and terminal output. The main difference is in the terminal window, which now displays the output of the application execution:

```

ortal> immortalsPopulation, int health, int defaultDamageValue, ImmortalUpdateReportCallback ucb ,Semaforo semaforo, Semaforo semaforoStop) {
29
30
31
32     alsPopulation;
33     ealht);
34     mageValue;
35
36

```

The terminal also shows the current weather and date.

```
src > main > java > edu > eci > arsw > highlandersim > J ImmortalJava 6, M > Immortal > fight(Immortal)
37     estoyVivo = true;
38 }
39
40 public void run() {
41
42     while (estoyVivo && semaforoStop.getBandera()) {
43
44         Immortal im;
45
46         if(!semaforo.getBandera()){
47             synchronized(semaforo){
48                 try {
49                     semaforo.wait();
50                 } catch (InterruptedException e) {
51                     e.printStackTrace();
52                 }
53             }
54         }
55
56         int myIndex = -1;
57         int nextFighterIndex = 0;
58
59         synchronized(immortalsPopulation){
60             myIndex = immortalsPopulation.indexOf(this);
61             if(immortalsPopulation.isEmpty()){
62                 nextFighterIndex = r.nextInt(immortalsPopulation.size());
63             }
64         }
65     }
66 }
```

SALIDA CONSOLA DE DEPURACIÓN TERMINAL 32

TERMINAL

PS C:\Users\luisa\OneDrive\Documentos\ARWS-2\LAB03\ARSW\_Lab03>

Lector de pantalla optimizado Lín. 107, col. 54 Espacios: 4 UTF-8 CRLF () Java powershell

13°C Prac. despejado Búsqueda

### III. Conclusiones

- La sincronización es crucial en la programación concurrente para evitar condiciones de carrera y garantizar la consistencia de los datos compartidos.
- La eficiencia en la concurrencia puede mejorarse mediante una planificación adecuada y el uso de estructuras de datos y técnicas de sincronización apropiadas.
- La detección y resolución de problemas de concurrencia puede requerir el uso de herramientas como jps y jstack para identificar bloqueos y condiciones de carrera.
- La simulación de sistemas concurrentes puede ser un desafío, y se deben tomar decisiones cuidadosas para garantizar la coherencia y eficiencia del programa.
- Es importante considerar el rendimiento y la eficiencia al diseñar y desarrollar aplicaciones concurrentes.