LABORATORIO 3

ARSW

SANTIAGO OSPINA
ISABELLA MANRIQUE

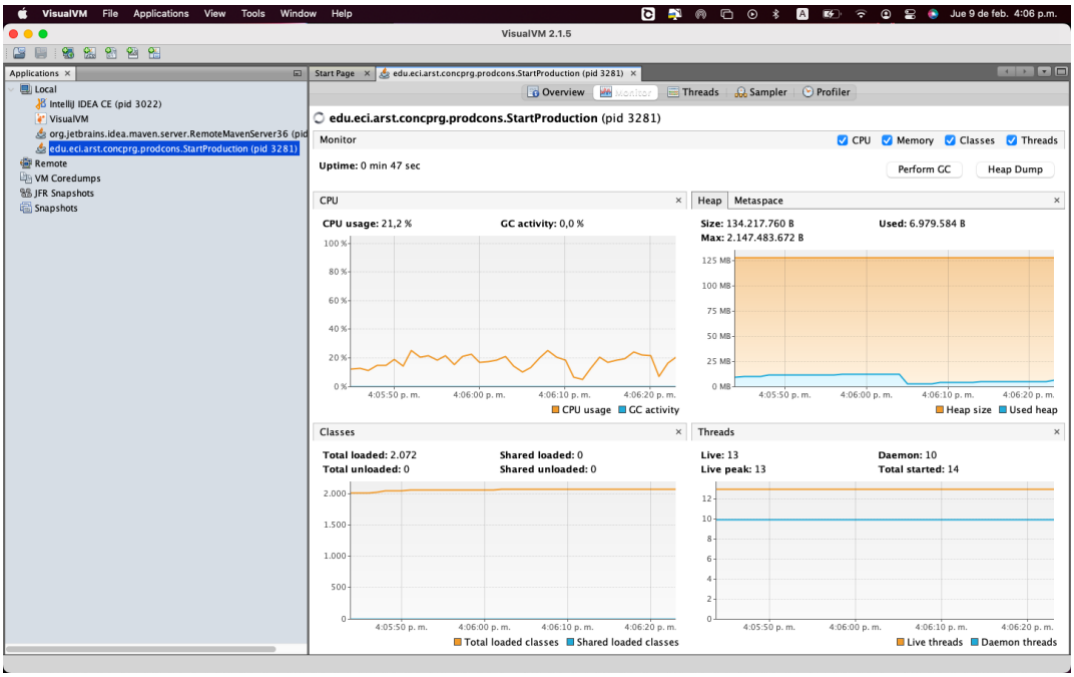
DIEGO TRIVIÑO

ESCUELA COLOMBIANA DE INGENIERIA JULIO GARAVITO
INGENIERIA DE SISTEMAS
2023-1
BOGOTA D.C.

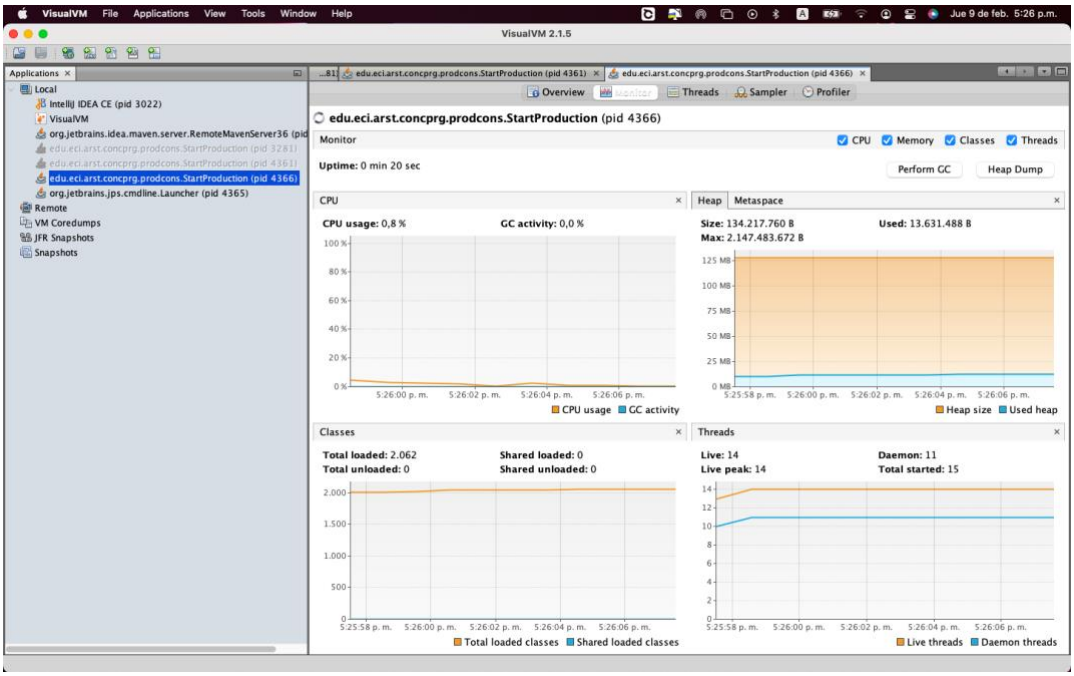
En este documento se presentan las evidencias en imágenes de cada uno de los puntos, las justificaciones se encuentran en el documento RESPUESTAS.txt

Primera Parte

1.



2. Consumo de CPU reducido



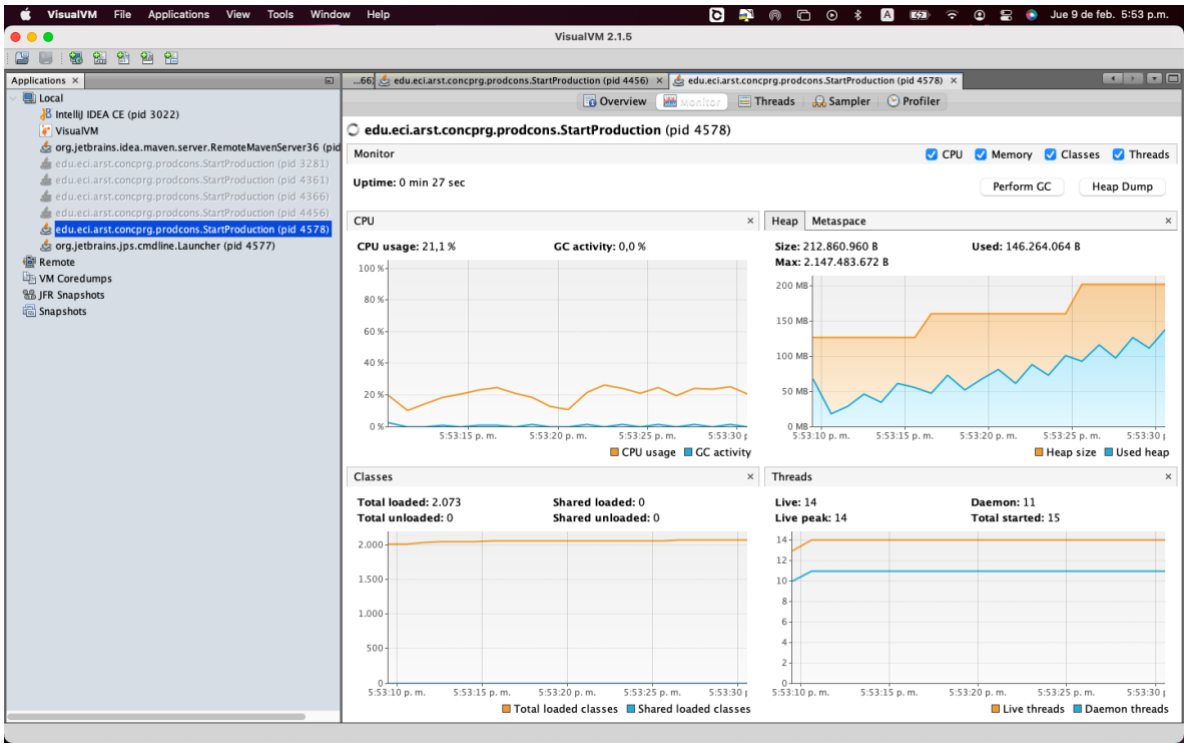
Clase Producer

```java
public void run() {
    while (true) {

        dataSeed = dataSeed + rand.nextInt( bound: 100);
        System.out.println("Producer added " + dataSeed);
        queue.add(dataSeed);
        synchronized (queue){
            queue.notifyAll();
        }
        try {
            Thread.sleep( millis: 1000);
        } catch (InterruptedException ex) {
            Logger.getLogger(Producer.class.getName()).log(Level.SEVERE,  msg: null, ex);
        }
    }
}
```

Clase Consumer

```
2 usages   Trivino Diego
public Consumer(Queue<Integer> queue) { this.queue=queue; }

Trivino Diego *
@Override
public void run() {
    while (true) {
        while(queue.size() == 0){
            synchronized (queue){
                try {
                    queue.wait();
                } catch (InterruptedException e) {
                    throw new RuntimeException(e);
                }
            }
        }
        if (queue.size() > 0) {
            int elem=queue.poll();
            System.out.println("Consumer consumes "+elem);
        }
```

3.



Clase Productor

Para que se respete el límite se agrega una condición que haga esperar al hilo, con el método wait, si la longitud de la cola es mayor al limite

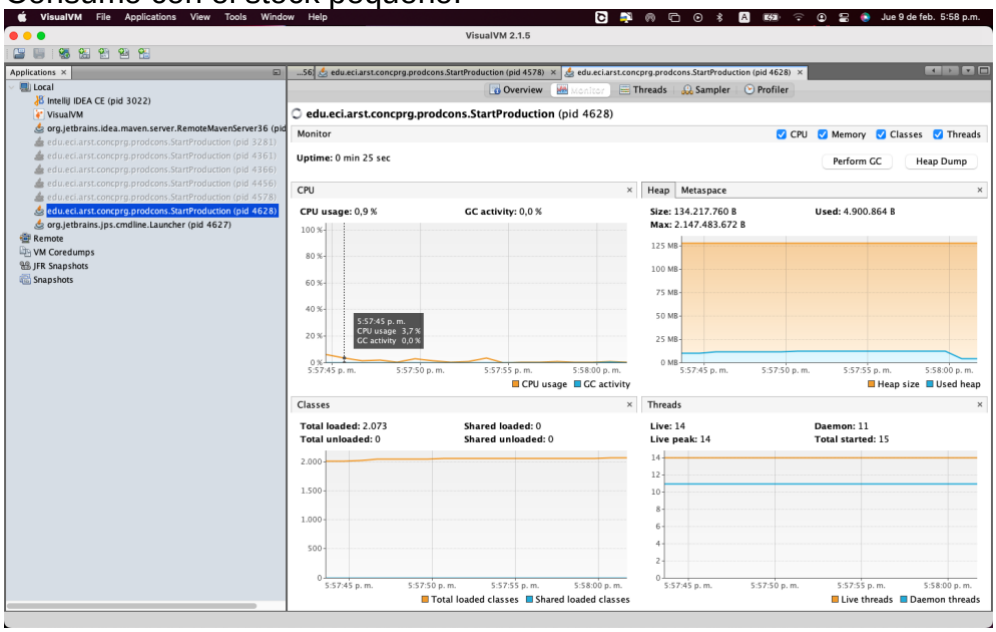
```
Trivino Diego *
@Override
public void run() {
    while (true) {
        while (queue.size() > stockLimit){
            try {
                queue.wait();
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
        }
        dataSeed = dataSeed + rand.nextInt( bound: 100);
        System.out.println("Producer added " + dataSeed);
        queue.add(dataSeed);
        synchronized (queue){
            queue.notifyAll();
        }
```

Clase Consumidor

Se agrega un notifyAll después de que se quita un elemento de la cola para que se notifique al productor que puede dejar de esperar y producir nuevamente.

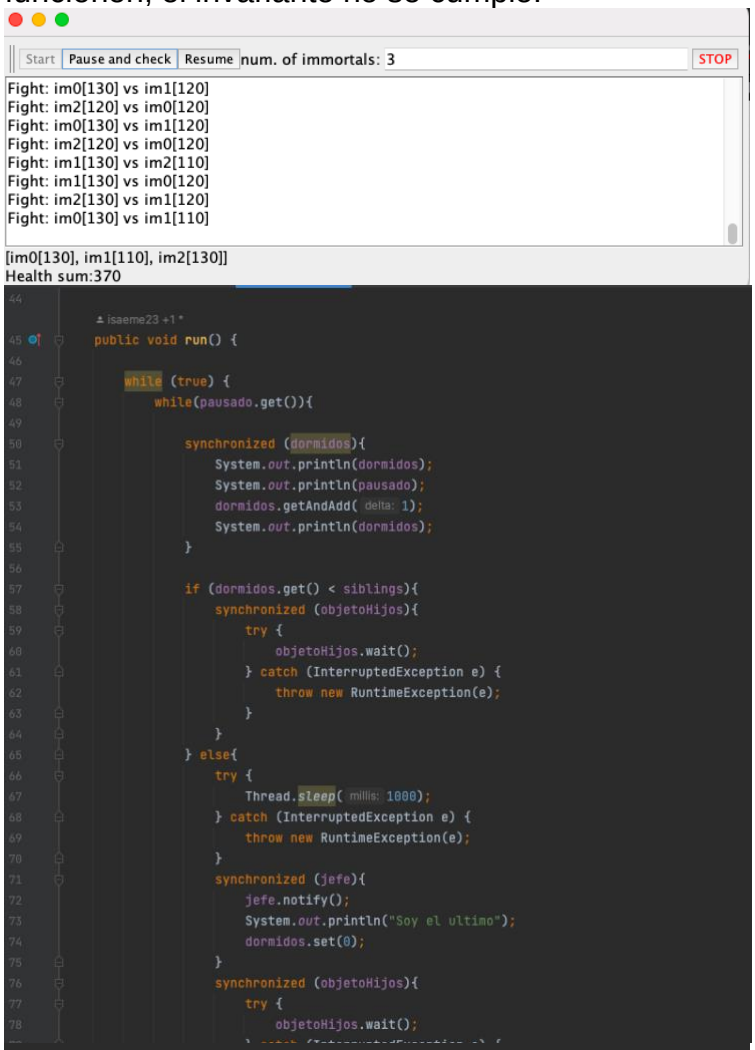
```java
@Override
public void run() {
    while (true) {
        while(queue.size() == 0){
            synchronized (queue){
                try {
                    queue.wait();
                } catch (InterruptedException e) {
                    throw new RuntimeException(e);
                }
            }
        }
        if (queue.size() > 0) {
            int elem=queue.poll();
            System.out.println("Consumer consumes "+elem);
        }
        synchronized (queue){
            queue.notifyAll();
        }
        try {
            Thread.sleep( millis: 1000);
        } catch (InterruptedException ex) {
            Logger.getLogger(Producer.class.getName()).log(Level.SEVERE,  msg: null, ex);
        }
    }
```

Consumo con el stock pequeño:



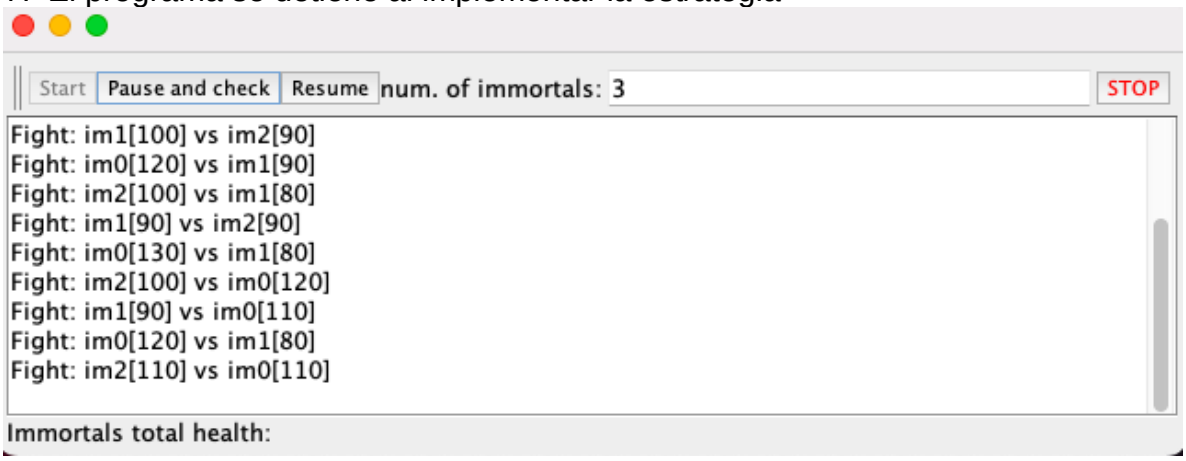
Segunda parte

4. Evidencia de que aunque los botones de de “Pause and Check” y “Resume” funcionen, el invariante no se cumple.



```java
    public void actionPerformed(ActionEvent e) {
        pausadoControl = true;
        try {
            pausar();
        } catch (InterruptedException ex) {
            throw new RuntimeException(ex);
        }

        /*
         * COMPLETAR
         */
        int sum = 0;
        for (Immortal im : immortals) {
            sum += im.getHealth();
        }

        statisticsLabel.setText("<html>"+immortals.toString()+"<br>Health sum:"+ sum);



    }
});
toolBar.add(btnPauseAndCheck);

JButton btnResume = new JButton( text: "Resume");

Trivino Diego +1
btnResume.addActionListener(new ActionListener() {
    Trivino Diego +1
    public void actionPerformed(ActionEvent e) {
        /**
         * IMPLEMENTAR
         */
        resume();
```

7. El programa se detiene al implementar la estrategia



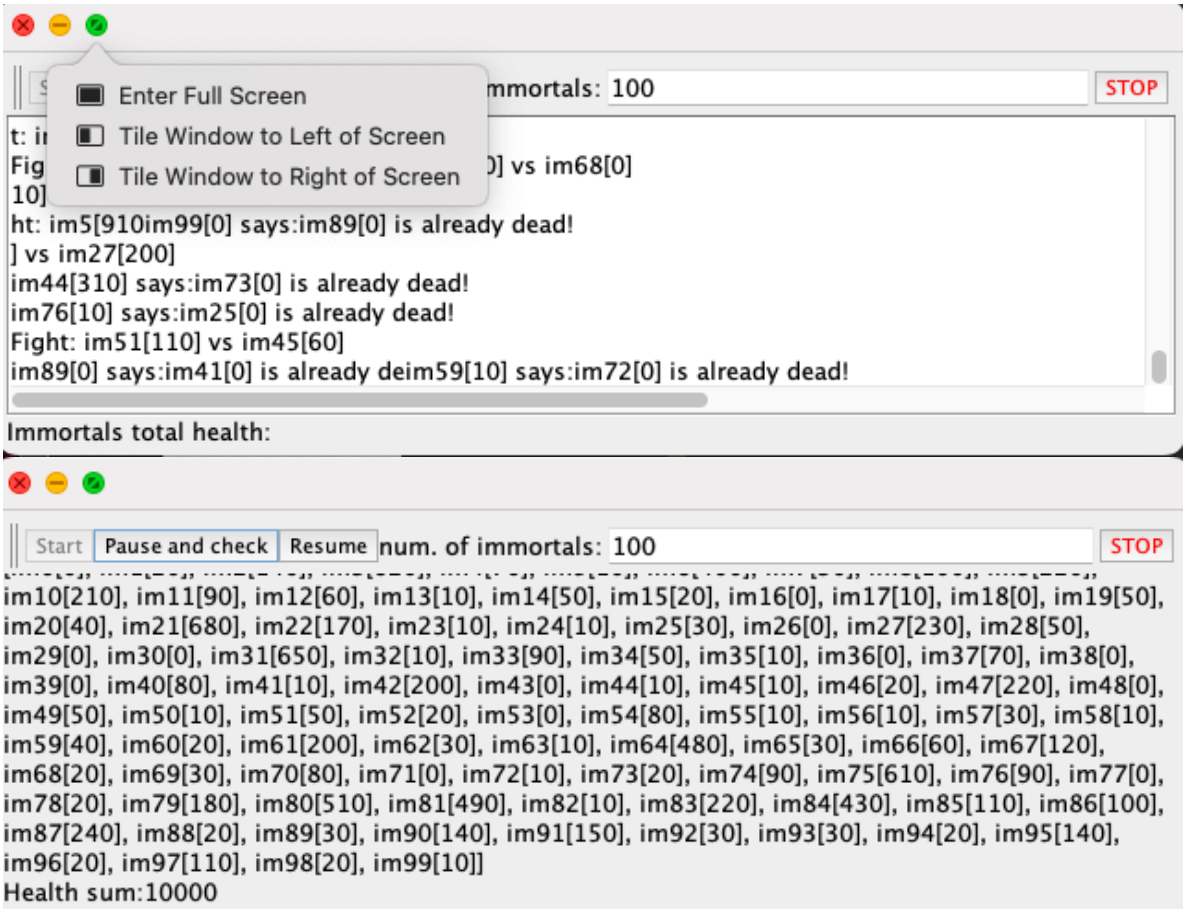
```java
1 usage   Trivino Diego *
public void fight(Immortal i2) {
    synchronized (this){
        synchronized (i2){
            if (i2.getHealth() > 0) {
                i2.changeHealth( v: i2.getHealth() - defaultDamageValue);
                this.health += defaultDamageValue;
                updateCallback.processReport("Fight: " + this + " vs " + i2+"\n");
            } else {
                updateCallback.processReport(this + " says:" + i2 + " is already dead!\n");
            }
        }
    }


}
```

Al ejecutar el programa y ver el thread dump en visualvm, logramos ver que al final del reporte los hilos vemos que todos se han quedado esperando simultáneamente.

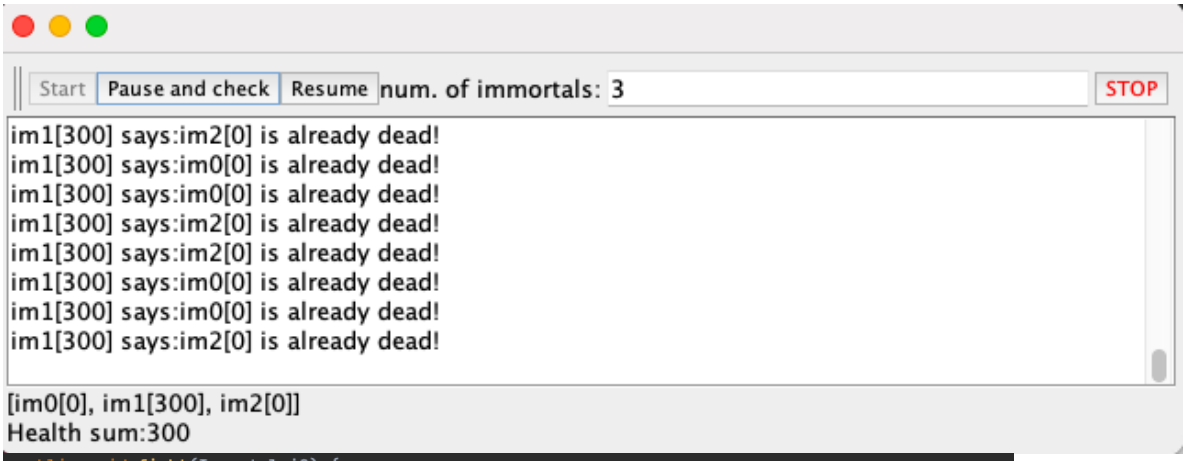
```
"im0" #20 prio=6 os_prio=31 cpu=7.55ms elapsed=4.30s tid=0x00007febc95dc800 nid=0x15267 waiting for monitor entry  [0x000070
   java.lang.Thread.State: BLOCKED (on object monitor)
        at edu.eci.arsw.highlandersim.Immortal.fight(Immortal.java:113)
        - waiting to lock <0x00000007874e3b48> (a edu.eci.arsw.highlandersim.Immortal)
        - locked <0x00000007874e25f0> (a edu.eci.arsw.highlandersim.Immortal)
        at edu.eci.arsw.highlandersim.Immortal.run(Immortal.java:98)

   Locked ownable synchronizers:
        - None

"im1" #21 prio=6 os_prio=31 cpu=3.49ms elapsed=4.30s tid=0x00007febca0de000 nid=0x10d0f waiting for monitor entry  [0x000070
   java.lang.Thread.State: BLOCKED (on object monitor)
        at edu.eci.arsw.highlandersim.Immortal.fight(Immortal.java:113)
        - waiting to lock <0x00000007874e3b48> (a edu.eci.arsw.highlandersim.Immortal)
        - locked <0x00000007874e3838> (a edu.eci.arsw.highlandersim.Immortal)
        at edu.eci.arsw.highlandersim.Immortal.run(Immortal.java:98)

   Locked ownable synchronizers:
        - None

"im2" #22 prio=6 os_prio=31 cpu=3.95ms elapsed=4.30s tid=0x00007febca0de800 nid=0xee03 waiting for monitor entry  [0x0000700
   java.lang.Thread.State: BLOCKED (on object monitor)
        at edu.eci.arsw.highlandersim.Immortal.fight(Immortal.java:113)
        - waiting to lock <0x00000007874e3838> (a edu.eci.arsw.highlandersim.Immortal)
        - locked <0x00000007874e3b48> (a edu.eci.arsw.highlandersim.Immortal)
        at edu.eci.arsw.highlandersim.Immortal.run(Immortal.java:98)

   Locked ownable synchronizers:
        - None
```

8. Estrategia planteada

```java
1 usage   Trivino Diego *
public void fight(Immortal i2) {
    if (i2.hashCode() > this.hashCode()){
        synchronized (this){
            synchronized (i2){
                if (i2.getHealth() > 0) {
                    i2.changeHealth( v: i2.getHealth() - defaultDamageValue);
                    this.health += defaultDamageValue;
                    updateCallback.processReport("Fight: " + this + " vs " + i2+"\n");
                } else {
                    updateCallback.processReport(this + " says:" + i2 + " is already dead!\n");
                }
            }
        }
    } else{
        synchronized (i2){
            synchronized (this){
                if (i2.getHealth() > 0) {
                    i2.changeHealth( v: i2.getHealth() - defaultDamageValue);
                    this.health += defaultDamageValue;
                    updateCallback.processReport("Fight: " + this + " vs " + i2+"\n");
                } else {
                    updateCallback.processReport(this + " says:" + i2 + " is already dead!\n");
                }
            }
        }
    }
}
```



9.




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

```java
public void run() {

    while (alive) {
        while(pausado.get()){

            synchronized (dormidos){
                System.out.println(dormidos);
                System.out.println(pausado);
                dormidos.getAndAdd( delta: 1);
                System.out.println(dormidos);
            }
```

10.

```
1 usage   new *
public void stop() throws InterruptedException {
    end.set(true);
}

btnStop.addActionListener(new ActionListener() {
    new *
    public void actionPerformed(ActionEvent e) {
        /**
         * IMPLEMENTAR
         */
        try {
            stop();
        } catch (InterruptedException ex) {
            throw new RuntimeException(ex);
        }
        btnResume.setEnabled(false);
        btnStart.setEnabled(false);
        btnPauseAndCheck.setEnabled(false);
    }
});
```