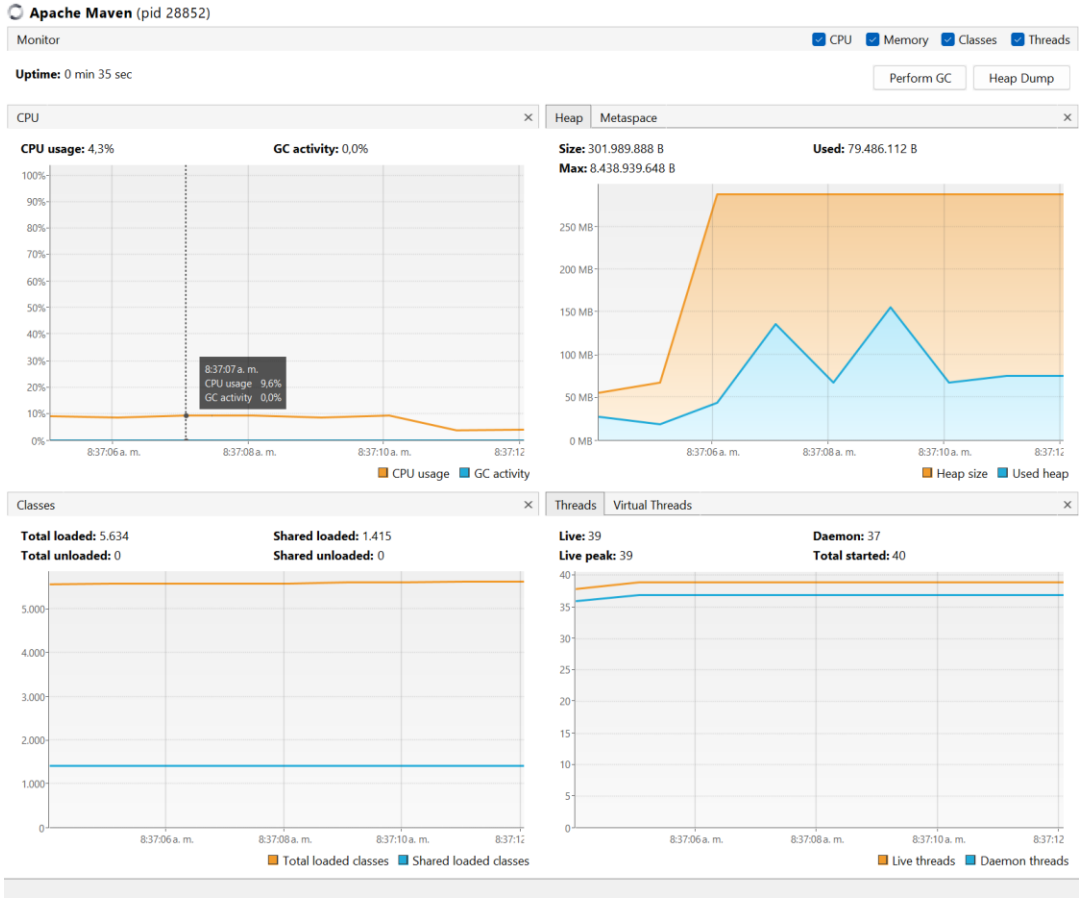
## Parte I — (Antes de terminar la clase) wait/notify: Productor/Consumidor

1. Ejecuta el programa de productor/consumidor y monitorea CPU con **jVisualVM**. ¿Por qué el consumo alto? ¿Qué clase lo causa?

Respuesta:

```
C:\Users\juanp\OneDrive\Desktop\Lab_busy_wait_vs_wait_notify>mvn -q -DskipTests exec:java -Dexec.mainClass=edu.eci.arsw.pc.PCApp -Dmode=spin -Dproducers=1 -Dconsumers=1 -D
capacity=8 -DprodDelayMs=0 -DconsDelayMs=0 -DdurationSec=30
Picked up JAVA_TOOL_OPTIONS: -Dstdout.encoding=UTF-8 -Dstderr.encoding=UTF-8
WARNING: A terminally deprecated method in sun.misc.Unsafe has been called
WARNING: sun.misc.Unsafe::staticFieldBase has been called by com.google.inject.internal.aop.HiddenClassDefiner (file:/C:/Program%20Files/opt/maven/apache-maven-3.9.11/lib/
guice-5.1.0-classes.jar)
WARNING: Please consider reporting this to the maintainers of class com.google.inject.internal.aop.HiddenClassDefiner
WARNING: sun.misc.Unsafe::staticFieldBase will be removed in a future release
PCApp mode=spin producers=1 consumers=1 capacity=8 prodDelay=0ms consDelay=0ms duration=30s
```




El consumo alto se debe al método aplicado al hilo (onSpinWait) en donde no pone el hilo no se bloquea, haciendo que el hilo se ejecute demasiadas veces ósea que se mantiene en estado RUNNABLE; esto también contribuye el ciclo while que permanece en un ciclo infinito y evalúa la condición millones de veces.

La clase que lo causa es “BusySpinQueue” y en donde tiene todo lo anterior mencionado.

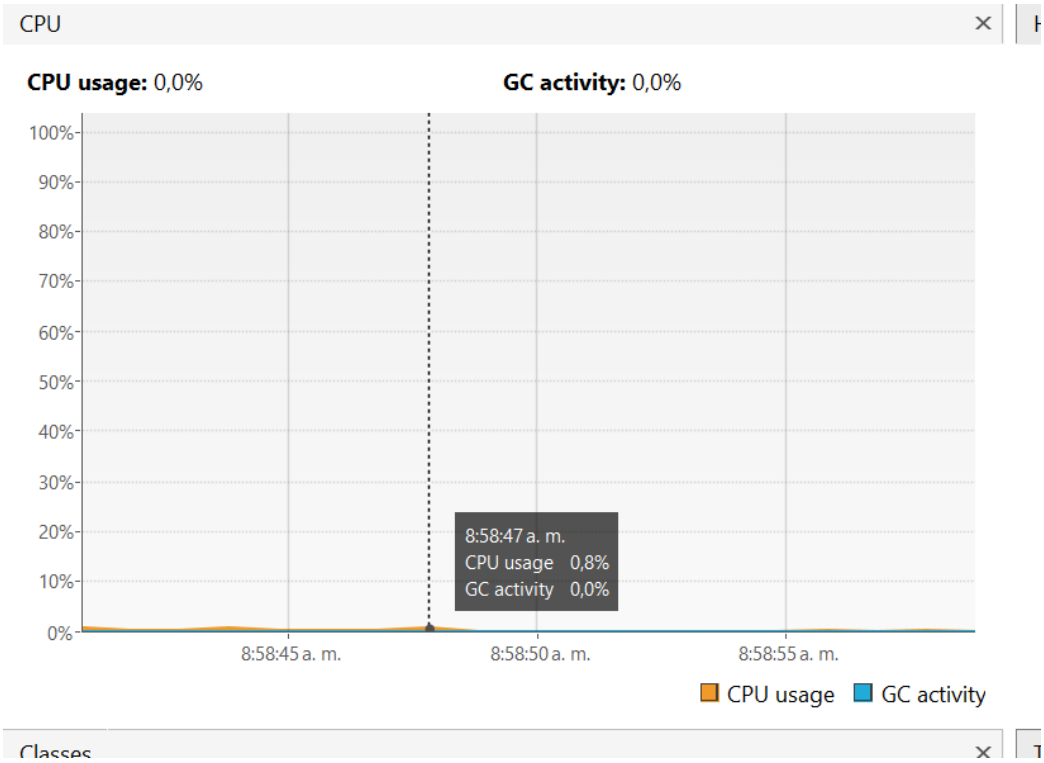
2. Ajusta la implementación para **usar CPU eficientemente** cuando el **productor es lento** y el **consumidor es rápido**. Valida de nuevo con VisualVM.

Respuesta:

```java
package edu.eci.arsw.pc;

import java.util.ArrayDeque;
import java.util.Deque;

/** Implementación correcta con monitores: synchronized + wait/notifyAll. */
public final class BoundedBuffer<T> {
  private final Deque<T> q = new ArrayDeque<>();
  private final int capacity;

  public BoundedBuffer(int capacity) {
    if (capacity <= 0)
      throw new IllegalArgumentException(s: "capacity must be > 0");
    this.capacity = capacity;
  }

  public void put(T item) throws InterruptedException {
    synchronized (this) {
      while (q.size() == capacity) {
        this.wait(); // espera hasta que haya espacio
      }
      q.addLast(item);
      this.notifyAll(); // despierta consumidores
    }
  }

  public T take() throws InterruptedException {
    synchronized (this) {
      while (q.isEmpty()) {
        this.wait(); // espera hasta que haya elementos
      }
      T v = q.removeFirst();
      this.notifyAll(); // despierta productores
      return v;
    }
  }

  public synchronized int size() {
    return q.size();
  }

  public int capacity() {
    return capacity;
  }
}
```

**Productor lento  y consumidor rápidos.**

```
C:\Users\juanp\OneDrive\Desktop\Lab_busy_wait_vs_wait_notify>mvn -q -DskipTests exec:java -Dexec.mainClass=edu.eci.arsw.pc.PCApp -Dmode=monitor -Dproducers=1 -Dconsumers=1
otify>mvn -q -DskipTests exec:java -Dexec.mainClass=edu.eci.arsw.pc.PCApp -Dmode=monitor -Dproducers=1 -Dconsumers=1 -Dcapacity=8 -DprodDelayMs=50 -DconsDelayMs=1 -Ddurati
onSec=30
Picked up JAVA_TOOL_OPTIONS: -Dstdout.encoding=UTF-8 -Dstderr.encoding=UTF-8
WARNING: A terminally deprecated method in sun.misc.Unsafe has been called
WARNING: sun.misc.Unsafe::staticFieldBase has been called by com.google.inject.internal.aop.HiddenClassDefiner (file:/C:/Program%20Files/opt/maven/apache-maven-3.9.11/lib/
guice-5.1.0-classes.jar)
WARNING: Please consider reporting this to the maintainers of class com.google.inject.internal.aop.HiddenClassDefiner
WARNING: sun.misc.Unsafe::staticFieldBase will be removed in a future release
PCApp mode=monitor producers=1 consumers=1 capacity=8 prodDelay=50ms consDelay=1ms duration=30s
```




Así que baja el consumo de cpu ya que el hilo no queda en estado activo si no que cada vez que evalué la condición el hilo debe esperar hasta que haya espacio y luego aparte de ese ciclo despertar los consumidores y productores.

- Los consumidores: esperan si no hay espacio.

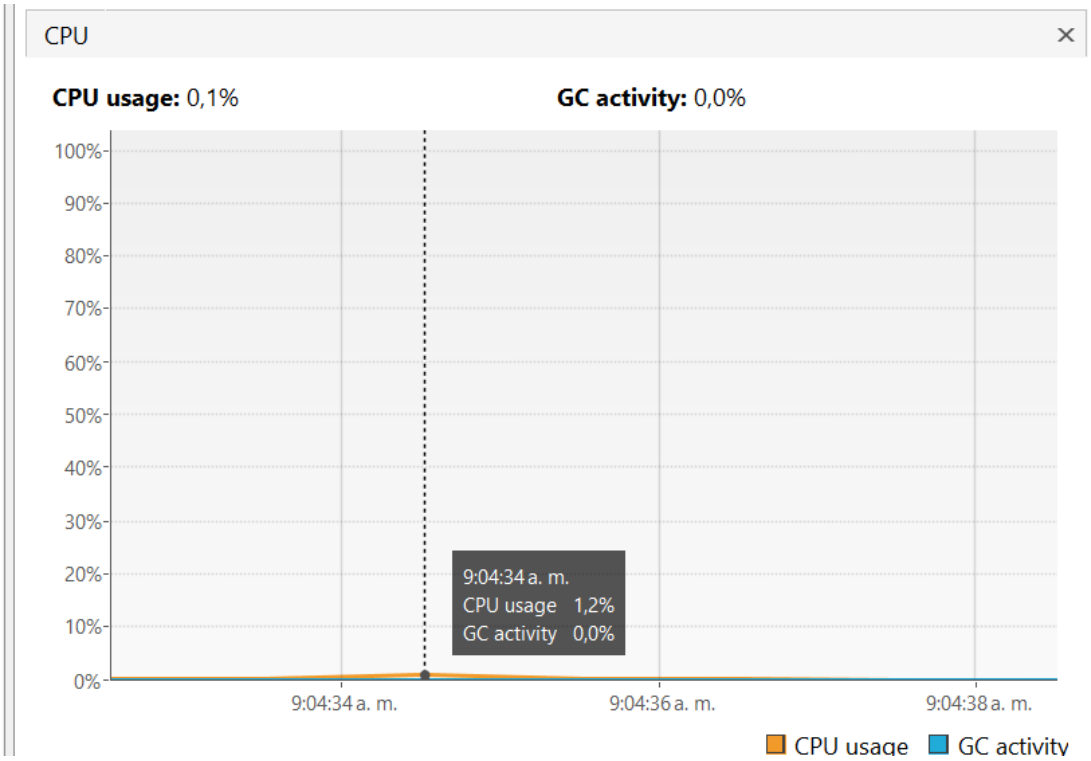
- Los productores: esperan si no hay datos.

3. Ahora **productor rápido** y **consumidor lento** con **límite de stock** (cola acotada): garantiza que el límite se respete **sin espera activa** y valida CPU con un stock pequeño.

Respuesta:

**Productor rápido + consumidor lento + cola acotada.**

```
C:\Users\juanp\OneDrive\Desktop\Lab_busy_wait_vs_wait_notify>mvn -q -DskipTests exec:java -Dexec.mainClass=edu.eci.arsw.pc.PCApp -Dmode=monitor -Dproducers=1 -Dconsumers
 -Dcapacity=4 -DprodDelayMs=0 -DconsDelayMs=100 -DdurationSec=120
Picked up JAVA_TOOL_OPTIONS: -Dstdout.encoding=UTF-8 -Dstderr.encoding=UTF-8
WARNING: A terminally deprecated method in sun.misc.Unsafe has been called
WARNING: sun.misc.Unsafe::staticFieldBase has been called by com.google.inject.internal.aop.HiddenClassDefiner (file:/C:/Program%20Files/opt/maven/apache-maven-3.9.11/li
guice-5.1.0-classes.jar)
WARNING: Please consider reporting this to the maintainers of class com.google.inject.internal.aop.HiddenClassDefiner
WARNING: sun.misc.Unsafe::staticFieldBase will be removed in a future release
PCApp mode=monitor producers=1 consumers=1 capacity=4 prodDelay=0ms consDelay=100ms duration=120s
```



**Productor**

- Productor rápido → prodDelayMs pequeño o 0

- Productor lento → prodDelayMs grande

**Consumidor**

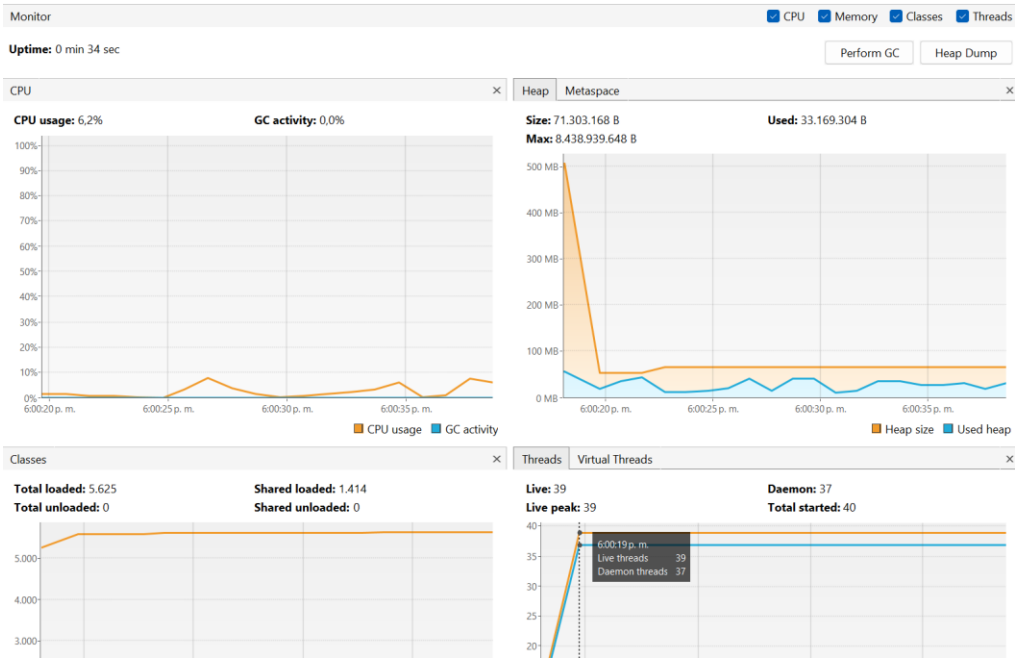
- Consumidor rápido → consDelayMs pequeño o 0

- Consumidor lento → consDelayMs grande

**Mode = spin VS Mode = monitor**

**Mode = spin (Productor rápido / consumidor rápido)**

*mvn -q -DskipTests exec:java -Dexec.mainClass=edu.eci.arsw.pc.PCApp -Dmode=spin -Dproducers=1 -Dconsumers=1 -Dcapacity=8 -DprodDelayMs=0 -DconsDelayMs=0 -DdurationSec=30*



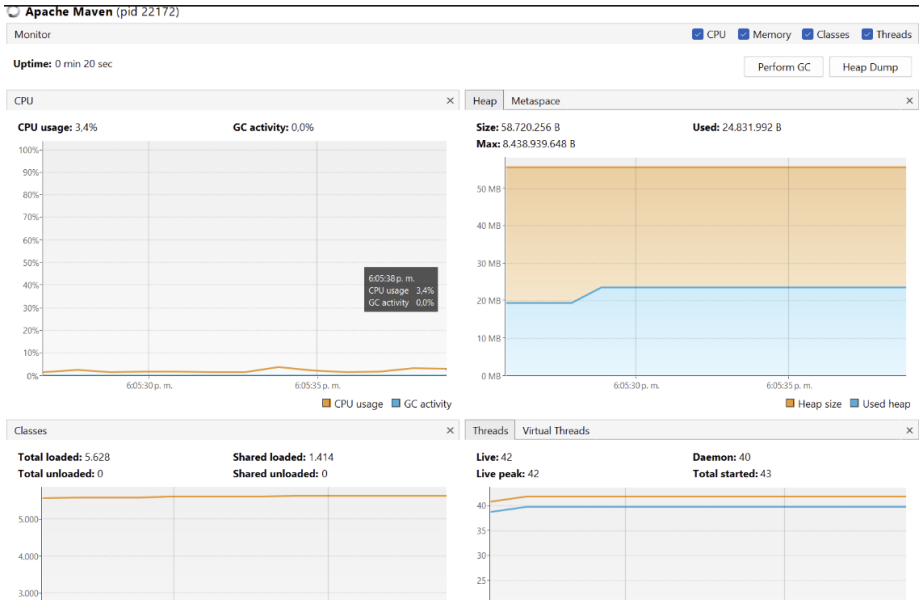

-La cpu nunca baja completamente a 0  mientras  corre.

- Existe espera activa.

- Estado RUNNABLE.

**Mode = spin (Productor lento / consumidor rápido)**

*mvn -q -DskipTests exec:java -Dexec.mainClass=edu.eci.arsw.pc.PCApp -Dmode=spin -Dproducers=1 -Dconsumers=1 -Dcapacity=8 -DprodDelayMs=0 -DconsDelayMs=100 -DdurationSec=30*



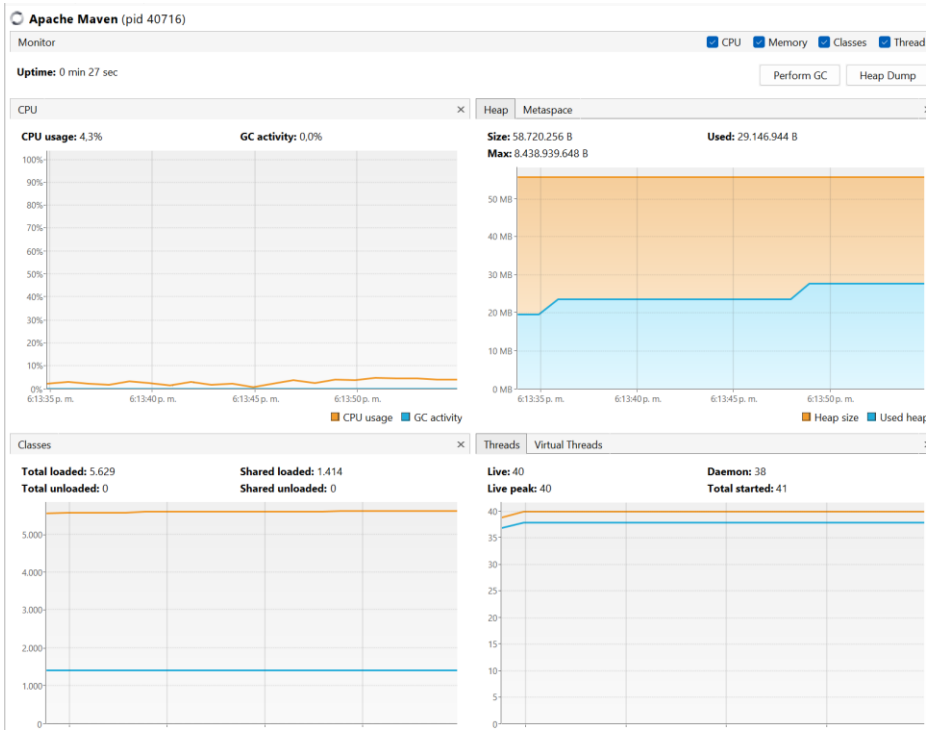

- El buffer se llena rápido.

- El productor entra en busy-wait.

- CPU alta constante.

**Mode = spin (Productor rápido / consumidor lento)**

*mvn -q -DskipTests exec:java -Dexec.mainClass=edu.eci.arsw.pc.PCApp -Dmode=spin -Dproducers=1 -Dconsumers=1 -Dcapacity=8 -DprodDelayMs=100 -DconsDelayMs=0 -DdurationSec=30*



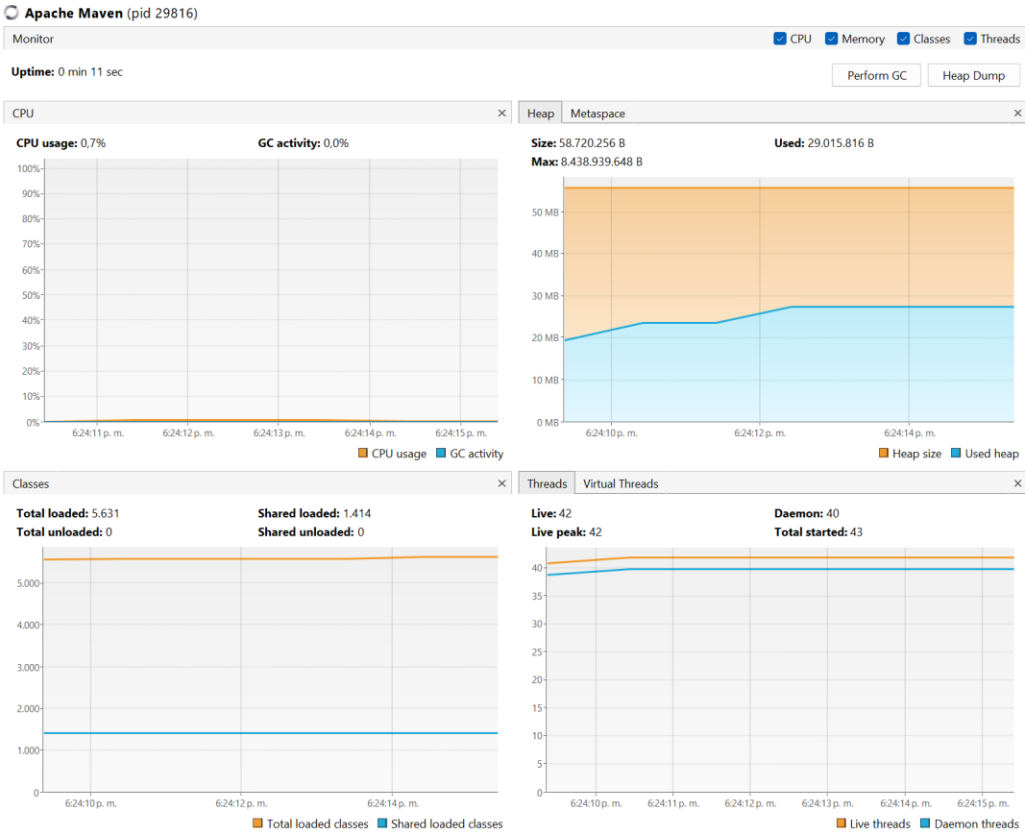

- El consumidor espera elementos

- Busy-wait del consumidor

- CPU alta aunque el buffer esté vacío

**Mode = spin (Productor lento / consumidor lento)**

mvn -q -DskipTests exec:java -Dexec.mainClass=edu.eci.arsw.pc.PCApp -Dmode=spin -Dproducers=1 -Dconsumers=1 -Dcapacity=8 -DprodDelayMs=100 -DconsDelayMs=100 -DdurationSec=30



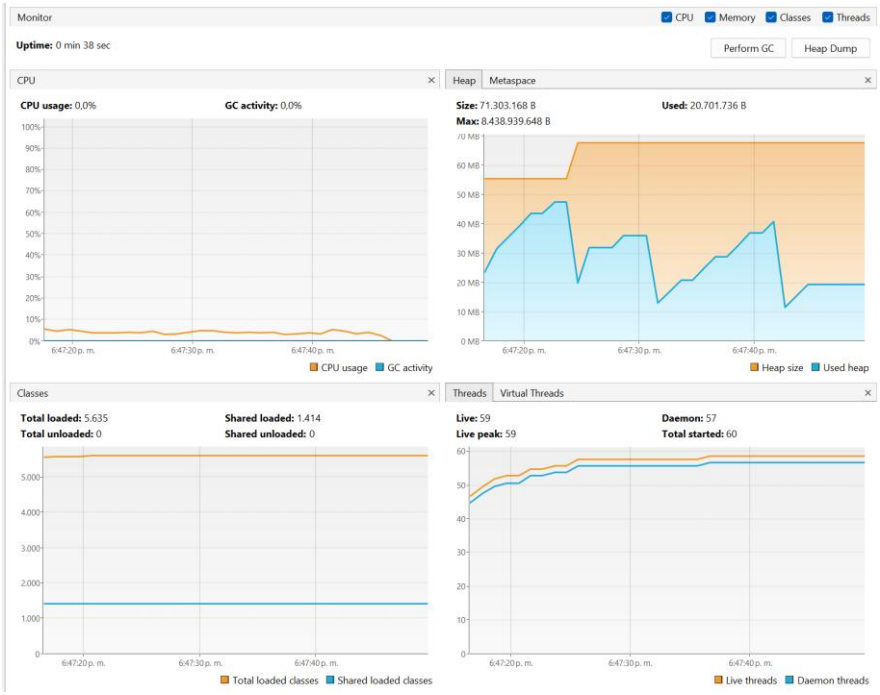

- Ambos trabajan despacio.

- La CPU no se ve tan sobrecargada.

- sigue habiendo espera activa.

**Mode = monitor (Productor rápido / consumidor rápido)**

*mvn -q -DskipTests exec:java -Dexec.mainClass=edu.eci.arsw.pc.PCApp -Dmode=monitor -Dproducers=1 -Dconsumers=1 -Dcapacity=8 -DprodDelayMs=0 -DconsDelayMs=0 -DdurationSec=30*



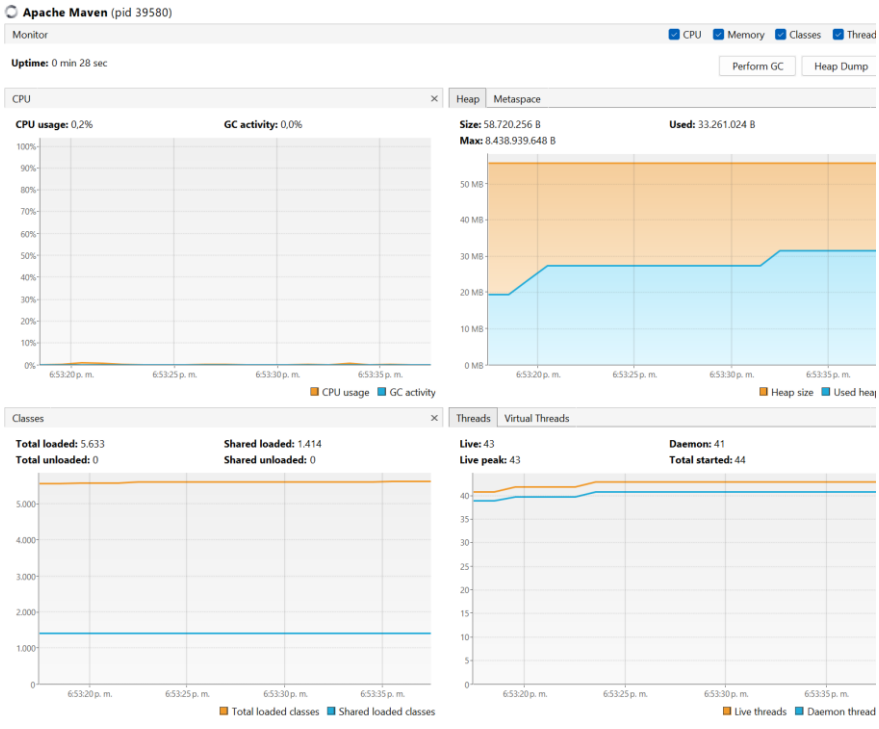

- Casi no hay bloqueos.

- CPU baja.

- Los Threads entrar brevemente en WAITING.

**Mode = monitor (Productor rápido / consumidor lento)**

mvn -q -DskipTests exec:java -Dexec.mainClass=edu.eci.arsw.pc.PCApp -Dmode=monitor -Dproducers=1 -Dconsumers=1 -Dcapacity=8 -DprodDelayMs=0 -DconsDelayMs=100 -DdurationSec=30




}

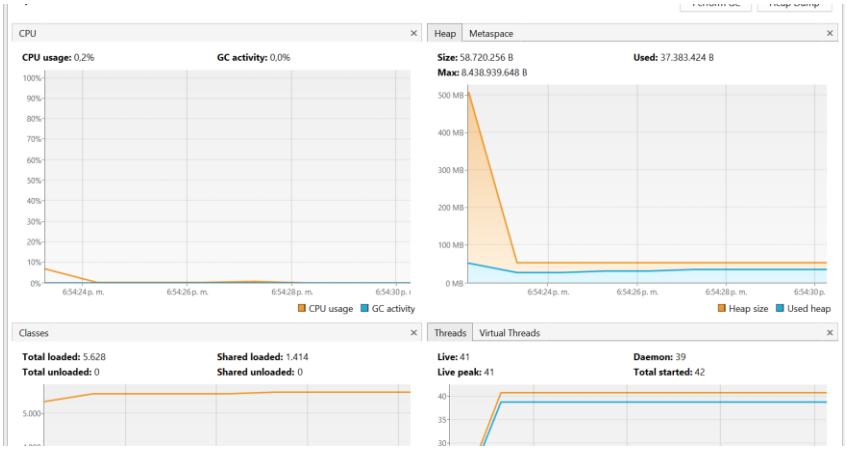
- El buffer se llena.

- El productor ejecuta wait().

- La CPU baja casi a 0 mientras espera.

**Mode = monitor (Productor lento / consumidor rápido)**

mvn -q -DskipTests exec:java -Dexec.mainClass=edu.eci.arsw.pc.PCApp -Dmode=monitor -Dproducers=1 -Dconsumers=1 -Dcapacity=8 -DprodDelayMs=100 -DconsDelayMs=0 -DdurationSec=30



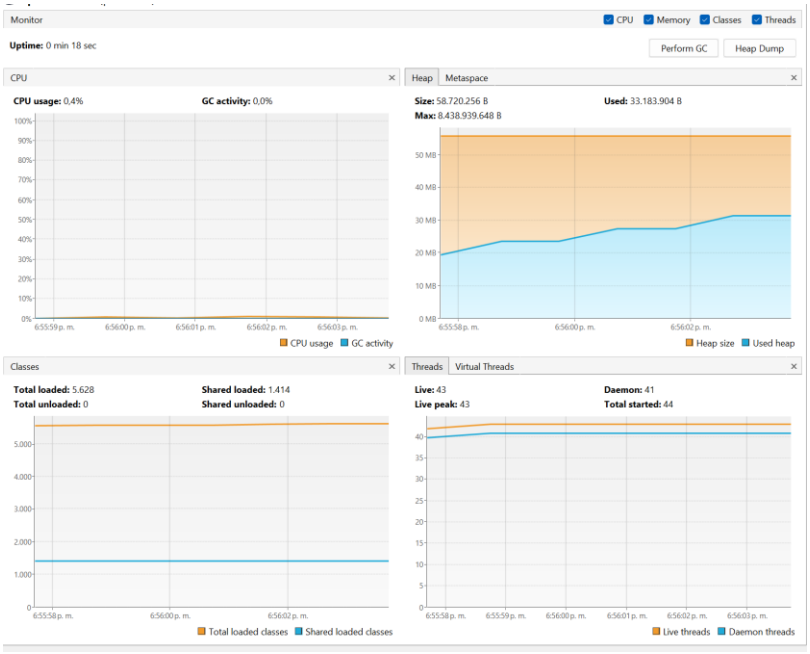

- El buffer se vacía.

- El consumidor ejecuta wait().

- La CPU muy baja.

**Mode = monitor (Productor lento / consumidor lento)**

mvn -q -DskipTests exec:java -Dexec.mainClass=edu.eci.arsw.pc.PCApp -Dmode=monitor -Dproducers=1 -Dconsumers=1 -Dcapacity=8 -DprodDelayMs=100 -DconsDelayMs=100 -DdurationSec=30




- Ambos trabajan despacio.

- Se bloquean cuando corresponde.

- La CPU baja y estable.