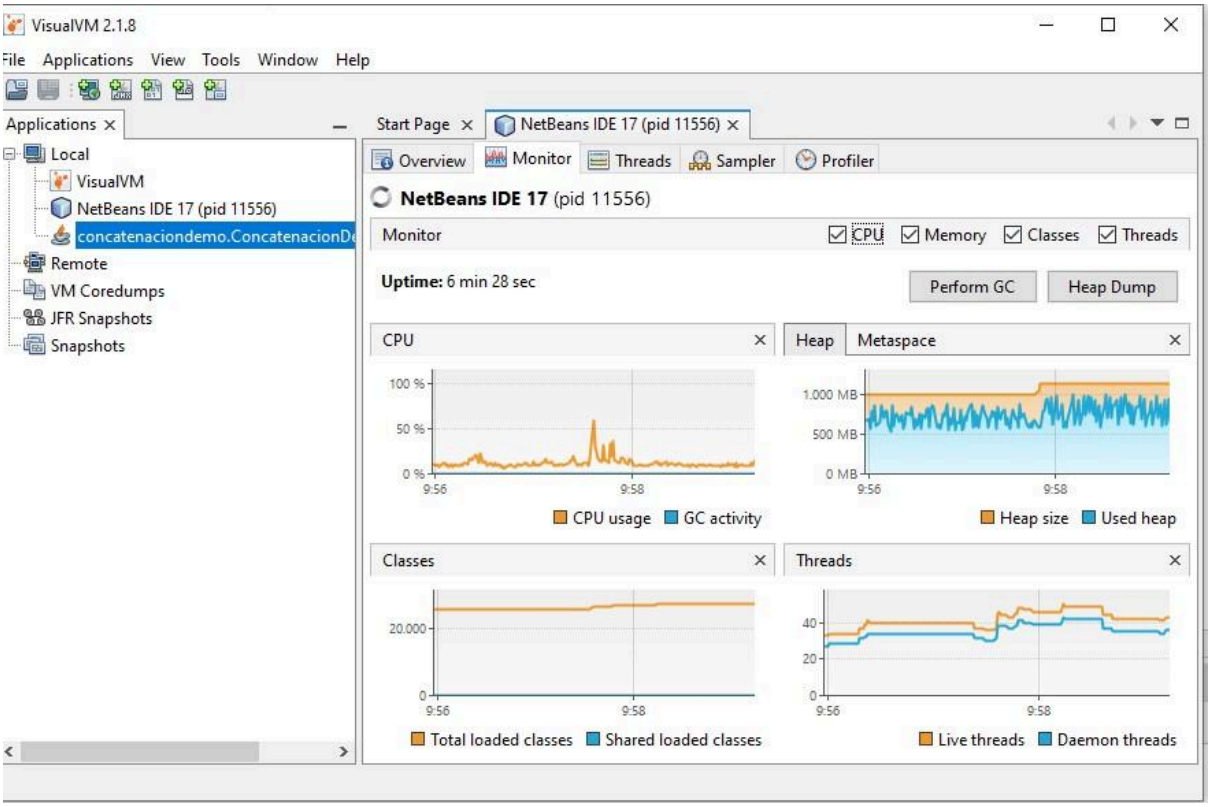
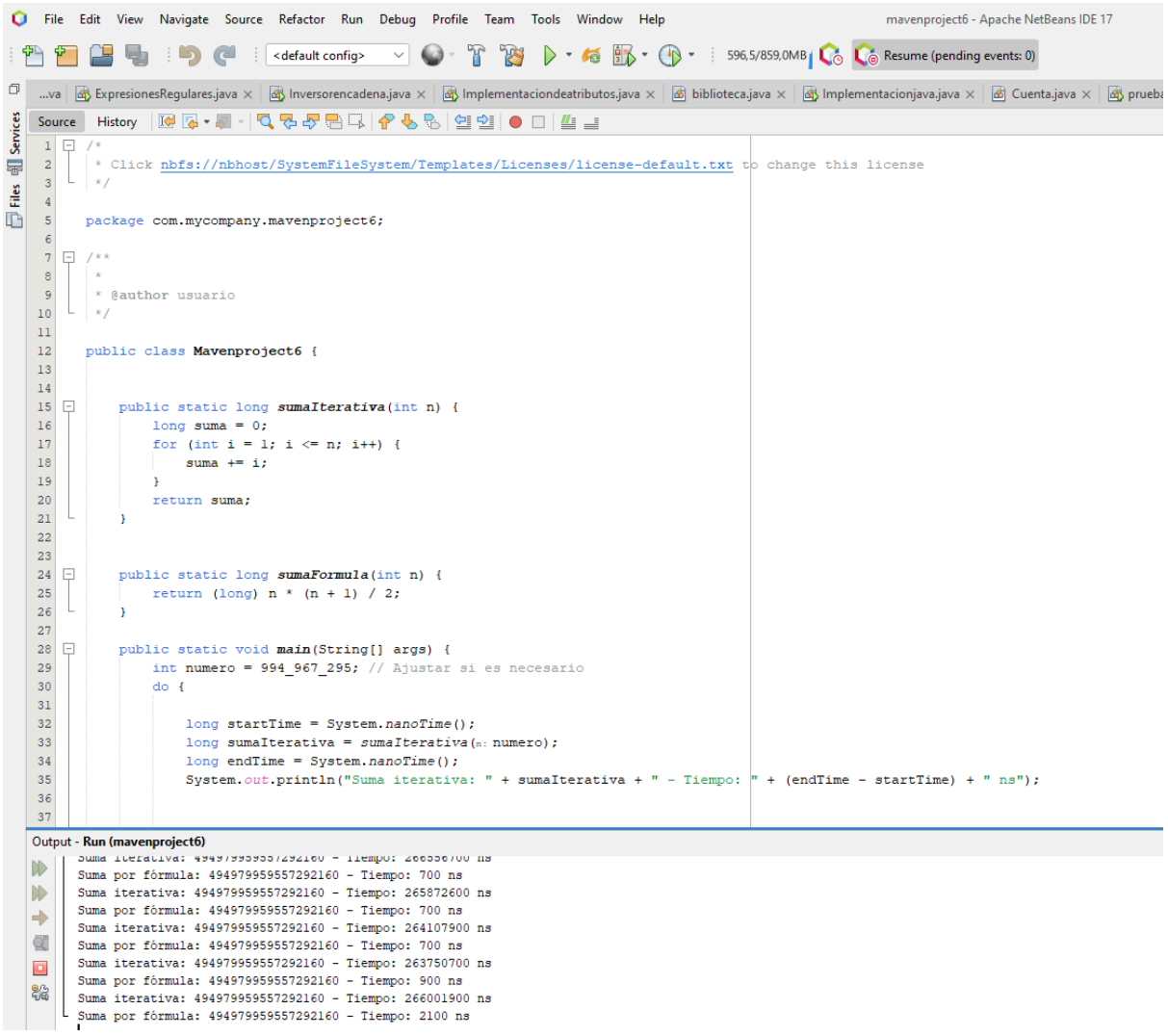
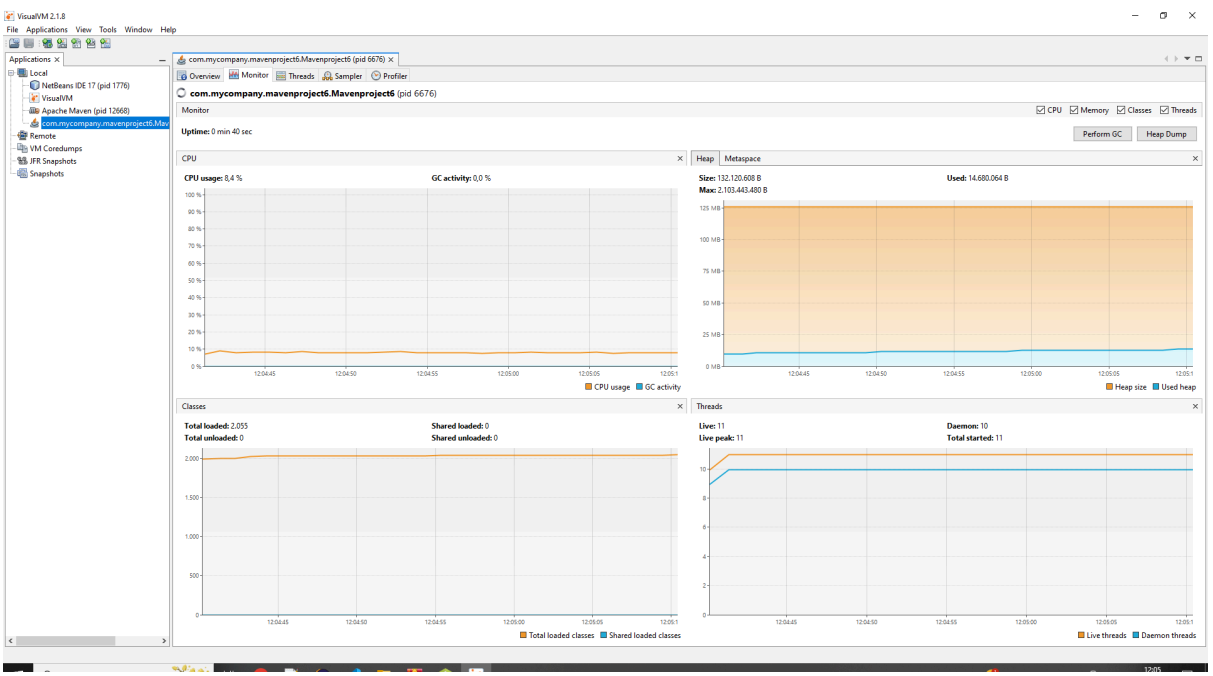

VisualVM 2.1.8
File Applications View Tools Window Help
Applications
Local
VisualVM
NetBeans IDE 17 (pid 11556)
concatenacionDemo.ConcatenacionD
Remote
VM Coredumps
JFR Snapshots
Snapshots
Start Page
NetBeans IDE 17 (pid 11556)
Overview
Monitor
Threads
Sampler
Profiler
NetBeans IDE 17 (pid 11556)
Monitor
CPU
Memory
Classes
Threads
Uptime: 6 min 28 sec
Perform GC
Heap Dump
CPU
100 %
50 %
0 %
9:56
9:58
CPU usage
GC activity
Heap
Metaspace
1.000 MB
500 MB
0 MB
Heap size
Used heap
Classes
20.000
Total loaded classes
Shared loaded classes
Threads
40
20
Live threads
Daemon threads


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
mavenproject6 - Apache NetBeans IDE 17
<default config>
596,5/859,0MB
Resume (pending events: 0)
ExpresionesRegulares.java
Inversorencadena.java
Implementaciondeatributos.java
biblioteca.java
Implementacionjava.java
Cuenta.java
Source
History
/*
* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
*/
package com.mycompany.mavenproject6;
/**
*
* @author usuario
*/
public class Mavenproject6 {
public static long sumaIterativa(int n) {
long suma = 0;
for (int i = 1; i <= n; i++) {
suma += i;
}
return suma;
}
public static long sumaFormula(int n) {
return (long) n * (n + 1) / 2;
}
public static void main(String[] args) {
int numero = 994_967_295; // Ajustar si es necesario
do {
long startTime = System.nanoTime();
long sumaIterativa = sumaIterativa(n: numero);
long endTime = System.nanoTime();
System.out.println("Suma iterativa: " + sumaIterativa + " - Tiempo: " + (endTime - startTime) + " ns");
Output - Run (mavenproject6)
Suma por fórmula: 494979959557292160 - Tiempo: 700 ns
Suma iterativa: 494979959557292160 - Tiempo: 265872600 ns
Suma por fórmula: 494979959557292160 - Tiempo: 700 ns
Suma iterativa: 494979959557292160 - Tiempo: 264107900 ns
Suma por fórmula: 494979959557292160 - Tiempo: 700 ns
Suma iterativa: 494979959557292160 - Tiempo: 263750700 ns
Suma por fórmula: 494979959557292160 - Tiempo: 900 ns
Suma iterativa: 494979959557292160 - Tiempo: 266001900 ns
Suma por fórmula: 494979959557292160 - Tiempo: 2100 ns


VisualVM 2.1.8
File Applications View Tools Window Help
com.mycompany.mavenproject6.Mavenproject6 (pid 6676)
Overview Monitor Threads Sampler Profiler
com.mycompany.mavenproject6.Mavenproject6 (pid 6676)
Monitor
Uptime: 0 min 40 sec
Perform GC
Heap Dump
CPU
CPU usage: 8,4 %
GC activity: 0,0 %
Heap
Metaspace
Size: 132.120.608 B
Used: 14.680.064 B
Max: 2.103.443.480 B
Classes
Total loaded: 2.055
Total unloaded: 0
Shared loaded: 0
Shared unloaded: 0
Threads
Live: 11
Live peak: 11
Daemon: 10
Total started: 11

Tras la observación con VisualVM ¿qué puedes decir del Heap en el apartado A? ¿y en el B? ¿A qué se debe? ¿Qué puedes decir del uso de CPU de apartado B?
En el Apartado A, la concatenación causa un aumento significativo en el uso de memoria debido a la La creación de objetos temporales, mientras que StringBuilder es más eficiente al evitar la creación de objetos adicionales. En el Apartado B, los cálculos intensivos generan un uso notable de la CPU,especialmente en el método de suma iterativa, debido al bucle que recorre muchas veces para calcular la suma. La fórmula de Gauss es más eficiente en términos de tiempo de ejecución que el enfoque iterativo, ya que realiza el cálculo de manera más directa. En ambos casos, VisualVM proporciona una herramienta útil para observar el comportamiento del heap y el uso de CPU durante la ejecución del código?