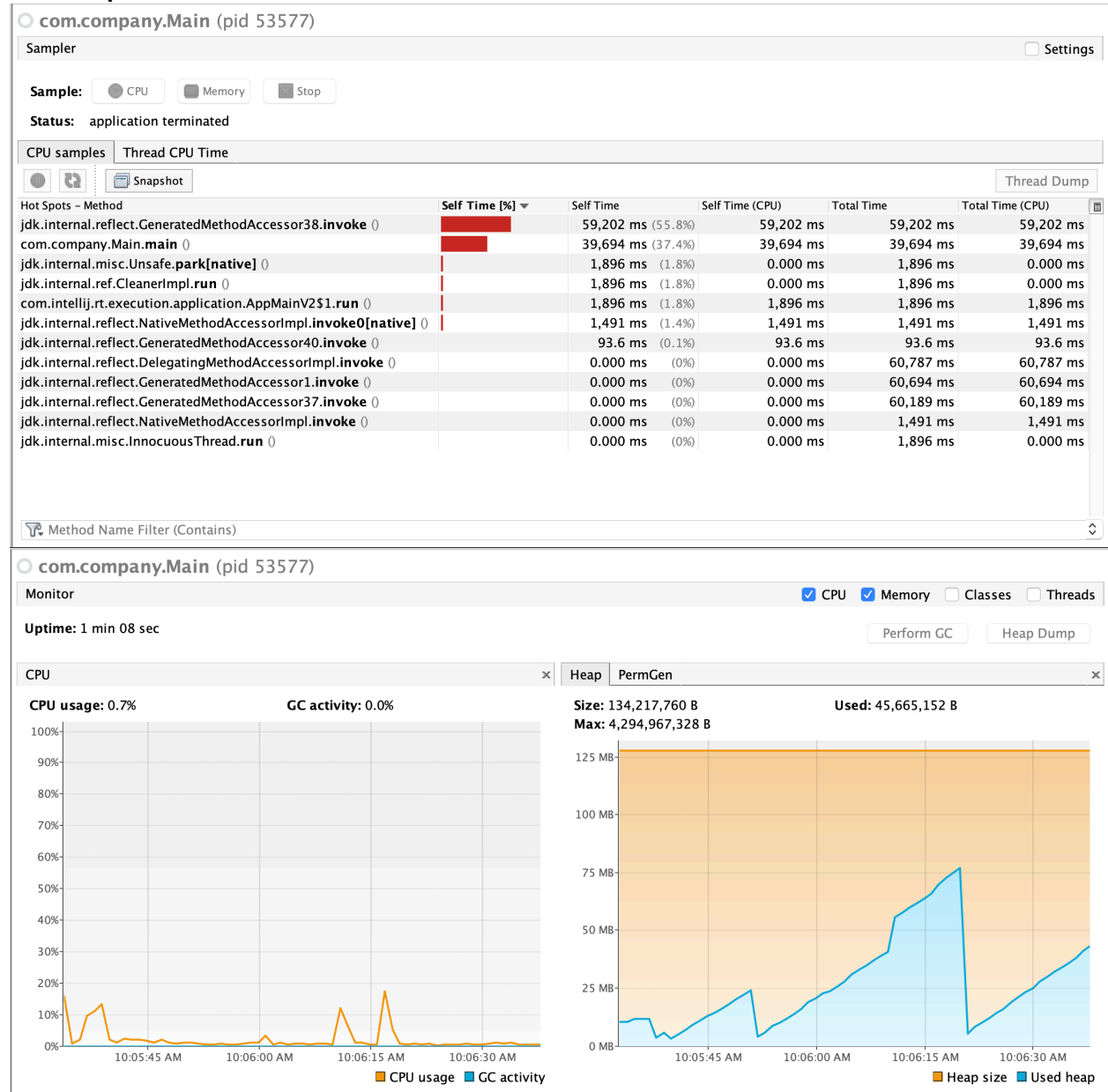


Douglas De León Molina  
Carné 18037

## HashMap



TreeMap

com.company.Main (pid 53725)

Sampler

Settings

Sample:

CPU

Memory

Stop

Status: application terminated

CPU samples

Thread CPU Time

Snapshot

Thread Dump

Hot Spots - Method

	Self Time [%]	Self Time	Self Time (CPU)	Total Time	Total Time (CPU)
jdk.internal.reflect.GeneratedMethodAccessor38.invoke ()		60,491 ms (47.8%)	60,491 ms	60,491 ms	60,491 ms
com.company.Main.main ()		43,806 ms (34.6%)	43,806 ms	43,892 ms	43,892 ms
jdk.internal.misc.Unsafe.park[native] ()		20,394 ms (16.1%)	0.000 ms	20,394 ms	0.000 ms
jdk.internal.reflect.NativeMethodAccessorImpl.invoke0[native] ()		1,494 ms (1.2%)	1,494 ms	1,494 ms	1,494 ms
jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke ()		97.9 ms (0.1%)	97.9 ms	62,273 ms	62,273 ms
jdk.internal.reflect.GeneratedMethodAccessor47.invoke ()		95.6 ms (0.1%)	95.6 ms	95.6 ms	95.6 ms
jdk.internal.reflect.GeneratedMethodAccessor3.invoke ()		94.2 ms (0.1%)	94.2 ms	94.2 ms	94.2 ms
com.company.Main.lambda\$main\$0 ()		85.3 ms (0.1%)	85.3 ms	85.3 ms	85.3 ms
jdk.internal.reflect.GeneratedMethodAccessor1.invoke ()		0.000 ms (0%)	0.000 ms	62,179 ms	62,179 ms
jdk.internal.reflect.GeneratedMethodAccessor37.invoke ()		0.000 ms (0%)	0.000 ms	61,679 ms	61,679 ms
jdk.internal.reflect.NativeMethodAccessorImpl.invoke ()		0.000 ms (0%)	0.000 ms	1,494 ms	1,494 ms
com.intellij.rt.execution.application.AppMainV2\$1.run ()		0.000 ms (0%)	0.000 ms	0.000 ms	0.000 ms
jdk.internal.misc.InnocuousThread.run ()		0.000 ms (0%)	0.000 ms	0.000 ms	0.000 ms
com.company.Main\$\$Lambda\$104.1845066581.accept ()		0.000 ms (0%)	0.000 ms	85.3 ms	85.3 ms
jdk.internal.ref.CleanerImpl.run ()		0.000 ms (0%)	0.000 ms	0.000 ms	0.000 ms

Method Name Filter (Contains)

com.company.Main (pid 53725)

Monitor

CPU

Memory

Classes

Threads

Uptime: 1 min 11 sec

Perform GC

Heap Dump

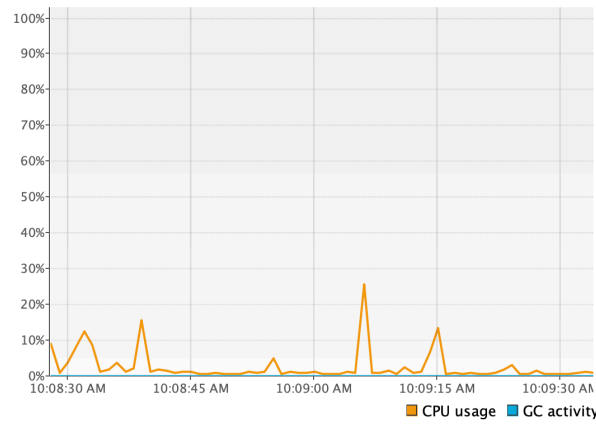
CPU

Heap

PermGen

CPU usage: 1.0%

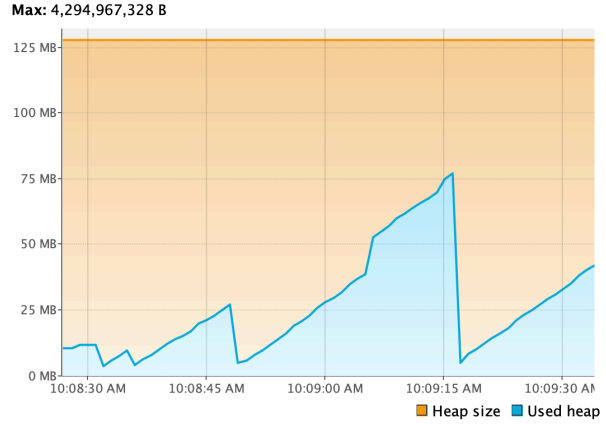
GC activity: 0.0%



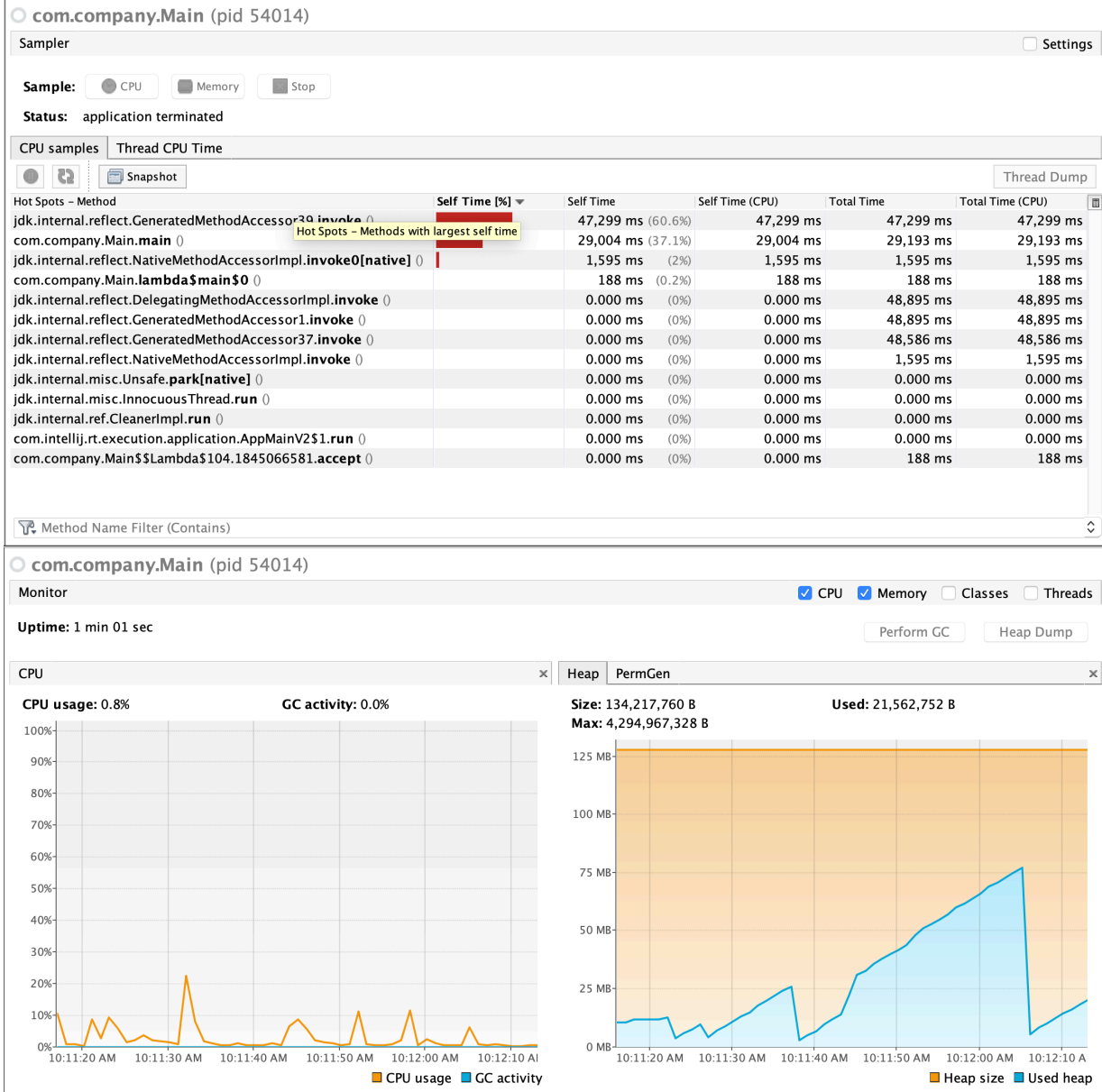
Size: 134,217,760 B

Max: 4,294,967,328 B

Used: 44,606,976 B



## LinkedHashMap



## **Conclusión y complejidad de tiempo de implementación HashMap**

Luego de realizar las pruebas se llegó a la conclusión que la implementación más rápida es con LinkedHashMap.

Luego de realizar varias pruebas de la complejidad de tiempo de la implementación de HashMap para mostrar todas las cartas se llegó a la conclusión de que la complejidad es de  $O(n)$ . Para llegar a esta conclusión se realizaron varias pruebas de tiempo con el profiler VisualVM alterando la cantidad de datos que se procesaban. Se observó que cuando se reducía la cantidad de datos, el tiempo bajaba proporcionalmente y esa observación se utilizó para argumentar mi conclusión.