

Hoja de trabajo #6

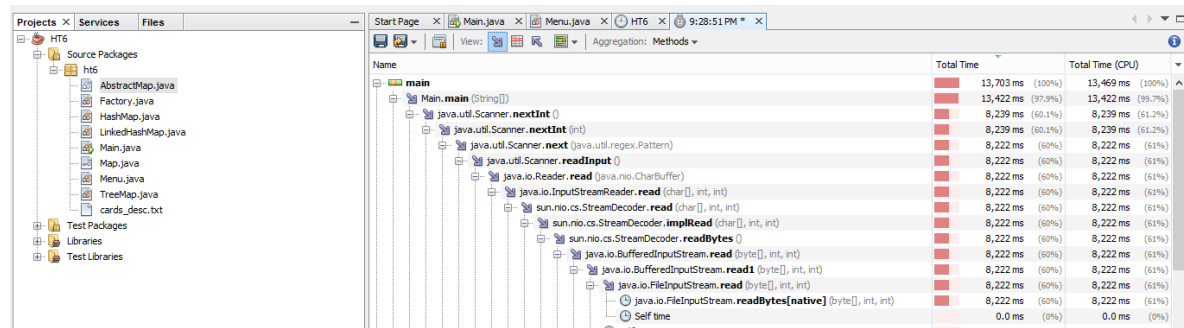
Profiling utilizado: Profiling NetBeans IDE

Dirección para el GitHub: <https://github.com/JennsiS/HDT6>

Medición de tiempo para cada implementación

- Mostrar todas las cartas

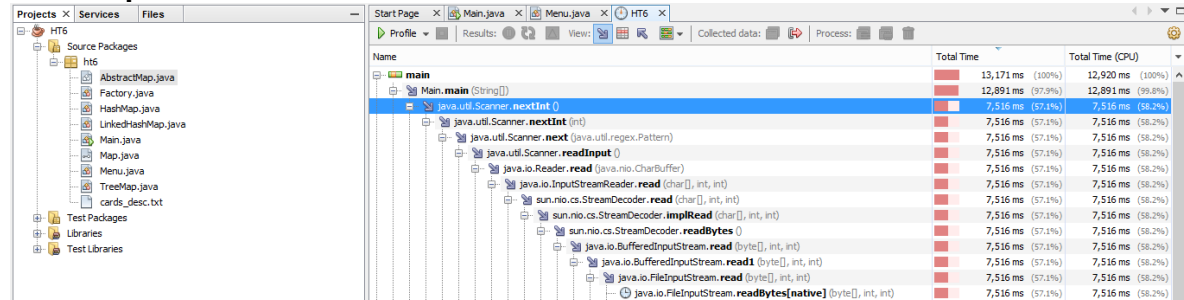
HashMap



Name	Total Time	Total Time (CPU)
main	13,703 ms (100%)	13,469 ms (100%)
main.Main (String[])	13,422 ms (97.9%)	13,422 ms (99.9%)
java.util.Scanner.nextInt ()	8,239 ms (60.1%)	8,239 ms (61.2%)
java.util.Scanner.next (int)	8,239 ms (60.1%)	8,239 ms (61.2%)
java.util.Scanner.next (java.util.regex.Pattern)	8,222 ms (60%)	8,222 ms (61%)
java.util.Scanner.readInput ()	8,222 ms (60%)	8,222 ms (61%)
java.io.Reader.read (java.nio.CharBuffer)	8,222 ms (60%)	8,222 ms (61%)
java.io.InputStreamReader.read (char[], int, int)	8,222 ms (60%)	8,222 ms (61%)
sun.nio.cs.StreamDecoder.read (char[], int, int)	8,222 ms (60%)	8,222 ms (61%)
sun.nio.cs.StreamDecoder.implRead (char[], int, int)	8,222 ms (60%)	8,222 ms (61%)
sun.nio.cs.StreamDecoder.readBytes ()	8,222 ms (60%)	8,222 ms (61%)
java.io.BufferedReader.read (byte[], int, int)	8,222 ms (60%)	8,222 ms (61%)
java.io.FileInputStream.read (byte[], int, int)	8,222 ms (60%)	8,222 ms (61%)
java.io.FileInputStream.readBytes (native) (byte[], int, int)	8,222 ms (60%)	8,222 ms (61%)
Self time	0.0 ms (0%)	0.0 ms (0%)

8,222 ms = 8.222 s

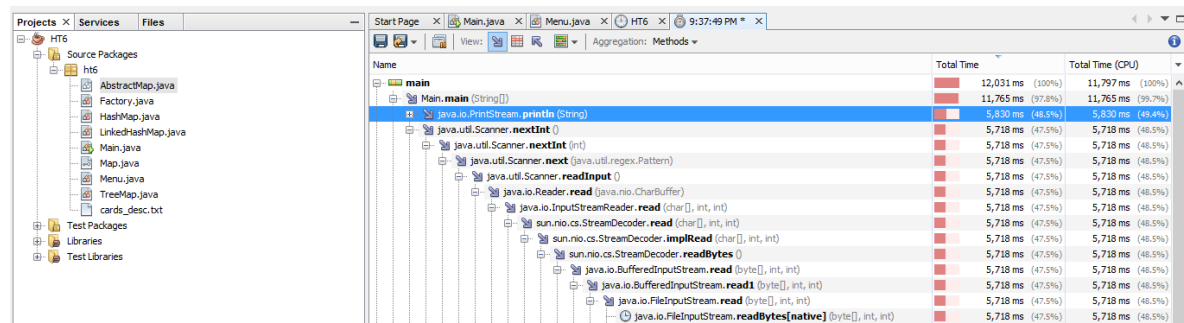
TreeMap



Name	Total Time	Total Time (CPU)
main	13,171 ms (100%)	12,920 ms (100%)
main.Main (String[])	12,891 ms (97.9%)	12,891 ms (99.8%)
java.util.Scanner.nextInt ()	7,516 ms (57.1%)	7,516 ms (58.2%)
java.util.Scanner.next (int)	7,516 ms (57.1%)	7,516 ms (58.2%)
java.util.Scanner.next (java.util.regex.Pattern)	7,516 ms (57.1%)	7,516 ms (58.2%)
java.util.Scanner.readInput ()	7,516 ms (57.1%)	7,516 ms (58.2%)
java.io.Reader.read (java.nio.CharBuffer)	7,516 ms (57.1%)	7,516 ms (58.2%)
java.io.InputStreamReader.read (char[], int, int)	7,516 ms (57.1%)	7,516 ms (58.2%)
sun.nio.cs.StreamDecoder.read (char[], int, int)	7,516 ms (57.1%)	7,516 ms (58.2%)
sun.nio.cs.StreamDecoder.implRead (char[], int, int)	7,516 ms (57.1%)	7,516 ms (58.2%)
sun.nio.cs.StreamDecoder.readBytes ()	7,516 ms (57.1%)	7,516 ms (58.2%)
java.io.BufferedReader.read (byte[], int, int)	7,516 ms (57.1%)	7,516 ms (58.2%)
java.io.FileInputStream.read (byte[], int, int)	7,516 ms (57.1%)	7,516 ms (58.2%)
java.io.FileInputStream.readBytes (native) (byte[], int, int)	7,516 ms (57.1%)	7,516 ms (58.2%)
Self time	0.0 ms (0%)	0.0 ms (0%)

7,516 ms = 7.516 s

LinkedHashMap



Name	Total Time	Total Time (CPU)
main	12,031 ms (100%)	11,797 ms (100%)
main.Main (String[])	11,765 ms (97.8%)	11,765 ms (99.7%)
java.io.PrintStream.println (String)	5,830 ms (48.4%)	5,830 ms (49.4%)
java.util.Scanner.nextInt ()	5,718 ms (47.5%)	5,718 ms (48.5%)
java.util.Scanner.next (int)	5,718 ms (47.5%)	5,718 ms (48.5%)
java.util.Scanner.next (java.util.regex.Pattern)	5,718 ms (47.5%)	5,718 ms (48.5%)
java.util.Scanner.readInput ()	5,718 ms (47.5%)	5,718 ms (48.5%)
java.io.Reader.read (java.nio.CharBuffer)	5,718 ms (47.5%)	5,718 ms (48.5%)
java.io.InputStreamReader.read (char[], int, int)	5,718 ms (47.5%)	5,718 ms (48.5%)
sun.nio.cs.StreamDecoder.read (char[], int, int)	5,718 ms (47.5%)	5,718 ms (48.5%)
sun.nio.cs.StreamDecoder.implRead (char[], int, int)	5,718 ms (47.5%)	5,718 ms (48.5%)
sun.nio.cs.StreamDecoder.readBytes ()	5,718 ms (47.5%)	5,718 ms (48.5%)
java.io.BufferedReader.read (byte[], int, int)	5,718 ms (47.5%)	5,718 ms (48.5%)
java.io.FileInputStream.read (byte[], int, int)	5,718 ms (47.5%)	5,718 ms (48.5%)
java.io.FileInputStream.readBytes (native) (byte[], int, int)	5,718 ms (47.5%)	5,718 ms (48.5%)
Self time	0.0 ms (0%)	0.0 ms (0%)

5718 ms= 5.718 s

- Mostrar todas las cartas en orden

HashMap

Name	Total Time	Total Time (CPU)
main	11,485 ms (100%)	11,282 ms (100%)
Main.main(String[])	11,236 ms (97.8%)	11,236 ms (99.6%)
java.util.Scanner.nextInt()	5,502 ms (47.9%)	5,502 ms (48.8%)
java.util.Scanner.next(java.util.regex.Pattern)	5,502 ms (47.9%)	5,502 ms (48.8%)
java.util.Scanner.readInput()	5,502 ms (47.9%)	5,502 ms (48.8%)
java.io.Reader.read(java.io.CharBuffer)	5,502 ms (47.9%)	5,502 ms (48.8%)
java.io.InputStreamReader.read(char[], int, int)	5,502 ms (47.9%)	5,502 ms (48.8%)
sun.nio.cs.StreamDecoder.read(char[], int, int)	5,502 ms (47.9%)	5,502 ms (48.8%)
sun.nio.cs.StreamDecoder.implRead(char[], int, int)	5,502 ms (47.9%)	5,502 ms (48.8%)
java.io.BufferedReader.readBytes()	5,502 ms (47.9%)	5,502 ms (48.8%)
java.io.BufferedReader.read(byte[], int, int)	5,502 ms (47.9%)	5,502 ms (48.8%)
java.io.FileInputStream.read(byte[], int, int)	5,502 ms (47.9%)	5,502 ms (48.8%)
java.io.FileInputStream.readBytes(native)(byte[], int, int)	5,502 ms (47.9%)	5,502 ms (48.8%)
Self time	0.0 ms (0%)	0.0 ms (0%)

5502 ms= 5.502 s

TreeMap

Name	Total Time	Total Time (CPU)
main	10,945 ms (100%)	10,725 ms (100%)
Main.main(String[])	10,678 ms (97.6%)	10,678 ms (99.6%)
java.util.Scanner.nextInt()	5,711 ms (52.2%)	5,711 ms (53.3%)
java.util.Scanner.next(java.util.regex.Pattern)	5,695 ms (52%)	5,695 ms (53.1%)
java.util.Scanner.readInput()	5,695 ms (52%)	5,695 ms (53.1%)
java.io.Reader.read(java.io.CharBuffer)	5,695 ms (52%)	5,695 ms (53.1%)
java.io.InputStreamReader.read(char[], int, int)	5,695 ms (52%)	5,695 ms (53.1%)
sun.nio.cs.StreamDecoder.read(char[], int, int)	5,695 ms (52%)	5,695 ms (53.1%)
sun.nio.cs.StreamDecoder.implRead(char[], int, int)	5,695 ms (52%)	5,695 ms (53.1%)
java.io.BufferedReader.readBytes()	5,695 ms (52%)	5,695 ms (53.1%)
java.io.BufferedReader.read(byte[], int, int)	5,695 ms (52%)	5,695 ms (53.1%)
java.io.FileInputStream.read(byte[], int, int)	5,695 ms (52%)	5,695 ms (53.1%)
java.io.FileInputStream.readBytes(native)(byte[], int, int)	5,695 ms (52%)	5,695 ms (53.1%)
Self time	0.0 ms (0%)	0.0 ms (0%)

5695 ms= 5.695 s

LinkedHashMap

Name	Total Time	Total Time (CPU)
main	10,436 ms (100%)	10,264 ms (100%)
Main.main(String[])	10,219 ms (97.9%)	10,219 ms (99.6%)
java.io.PrintStream.println(String)	5,080 ms (48.7%)	5,080 ms (49.5%)
java.util.Scanner.nextInt()	4,736 ms (45.4%)	4,736 ms (46.1%)
Menu.printMenu2()	124 ms (1.2%)	124 ms (1.2%)
java.io.BufferedReader.readLine()	77.6 ms (0.7%)	77.6 ms (0.8%)
java.util.Scanner.<init>(java.io.InputStream)	38.1 ms (0.4%)	38.1 ms (0.4%)
java.util.LinkedHashMap.get(Object)	33.5 ms (0.3%)	33.5 ms (0.3%)
java.lang.StringBuilder.toString()	30.2 ms (0.3%)	30.2 ms (0.3%)
Self time	0.0 ms (0%)	0.0 ms (0%)

4736 ms= 4.736 s

Análisis

Cuando las cartas están desordenadas y cuando están ordenadas linkedhashmap es la implementación más rápida. Esto se debe a que tiene diferente complejidad para agregar cosas al map y para mostrar lo que contiene en el map.

Complejidad calculada para HashMap

En el mejor de los casos debería de ser $O(1)$ pero esto si únicamente tuviera que mostrar un elemento, en el caso promedio es $O(n)$. Esto indica que la complejidad de un HashMap depende del número de elementos que contenga. En este caso puede ser variable dependiendo de que cuantas cargas tenga.