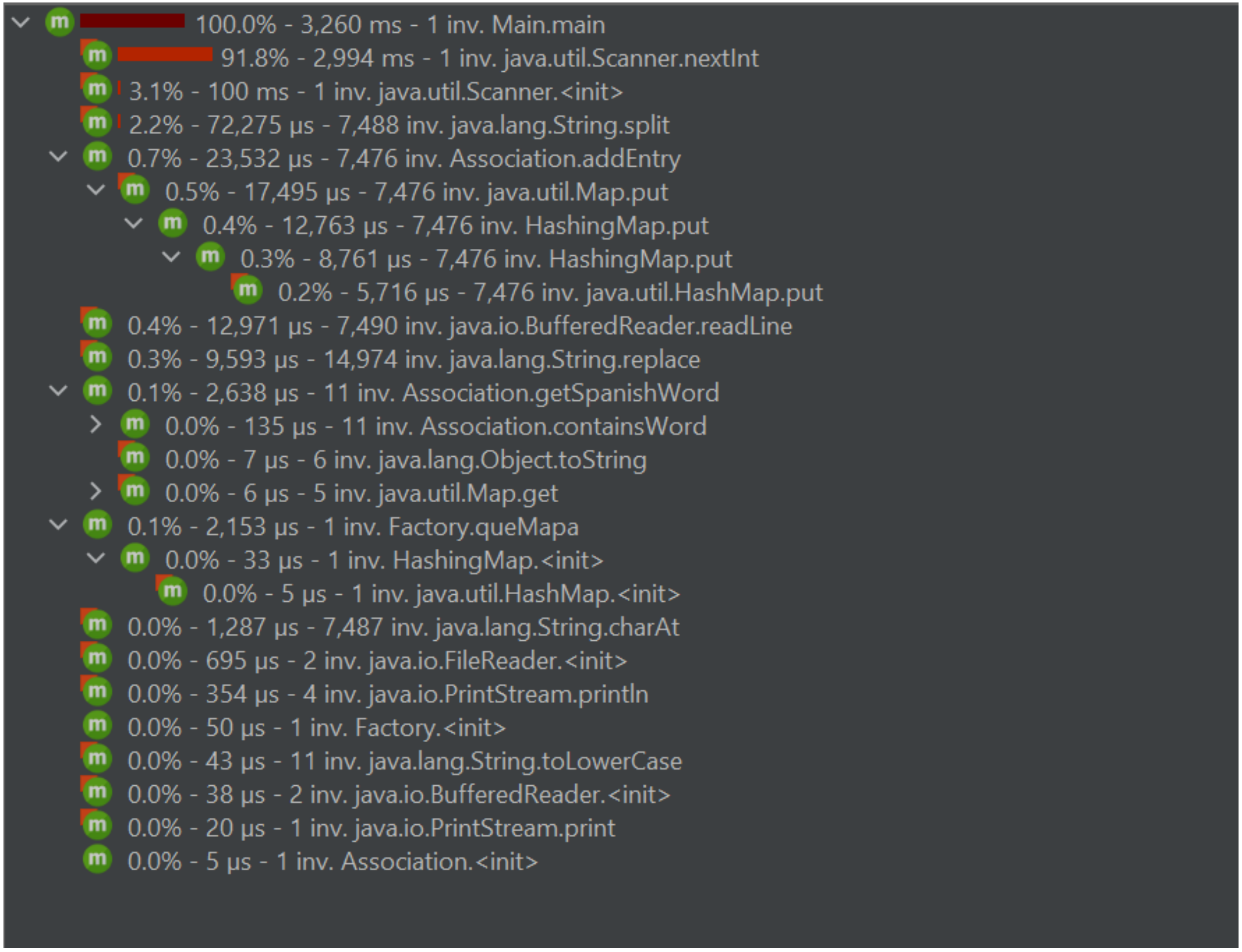
Algoritmos y Estructuras de Datos 13-05-2021
Comparación entre implementaciones usadas
Javier Mombiela
Mariano Reyes

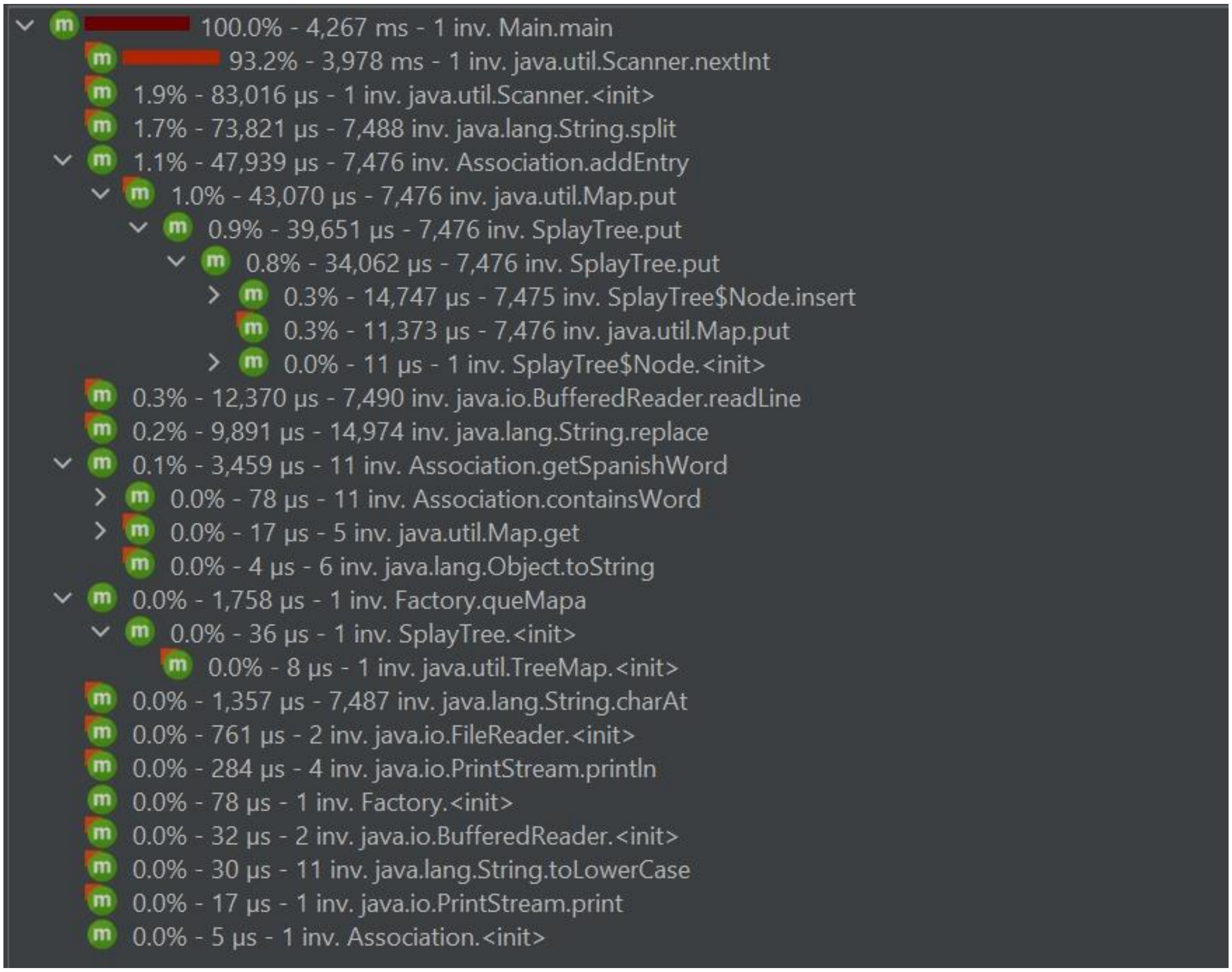
Hashing Map



```
100.0% - 3,260 ms - 1 inv. Main.main
    91.8% - 2,994 ms - 1 inv. java.util.Scanner.nextInt
    3.1% - 100 ms - 1 inv. java.util.Scanner.<init>
    2.2% - 72,275 µs - 7,488 inv. java.lang.String.split
    0.7% - 23,532 µs - 7,476 inv. Association.addEntry
        0.5% - 17,495 µs - 7,476 inv. java.util.Map.put
            0.4% - 12,763 µs - 7,476 inv. HashingMap.put
                0.3% - 8,761 µs - 7,476 inv. HashingMap.put
                    0.2% - 5,716 µs - 7,476 inv. java.util.HashMap.put
    0.4% - 12,971 µs - 7,490 inv. java.io.BufferedReader.readLine
    0.3% - 9,593 µs - 14,974 inv. java.lang.String.replace
    0.1% - 2,638 µs - 11 inv. Association.getSpanishWord
        0.0% - 135 µs - 11 inv. Association.containsWord
        0.0% - 7 µs - 6 inv. java.lang.Object.toString
        0.0% - 6 µs - 5 inv. java.util.Map.get
    0.1% - 2,153 µs - 1 inv. Factory.queMapa
        0.0% - 33 µs - 1 inv. HashingMap.<init>
            0.0% - 5 µs - 1 inv. java.util.HashMap.<init>
    0.0% - 1,287 µs - 7,487 inv. java.lang.String.charAt
    0.0% - 695 µs - 2 inv. java.io.FileReader.<init>
    0.0% - 354 µs - 4 inv. java.io.PrintStream.println
    0.0% - 50 µs - 1 inv. Factory.<init>
    0.0% - 43 µs - 11 inv. java.lang.String.toLowerCase
    0.0% - 38 µs - 2 inv. java.io.BufferedReader.<init>
    0.0% - 20 µs - 1 inv. java.io.PrintStream.print
    0.0% - 5 µs - 1 inv. Association.<init>
```

Algoritmos y Estructuras de Datos 13-05-2021
Comparación entre implementaciones usadas
Javier Mombiela
Mariano Reyes

Splay Tree



```
100.0% - 4,267 ms - 1 inv. Main.main
  93.2% - 3,978 ms - 1 inv. java.util.Scanner.nextInt
  1.9% - 83,016 µs - 1 inv. java.util.Scanner.<init>
  1.7% - 73,821 µs - 7,488 inv. java.lang.String.split
  1.1% - 47,939 µs - 7,476 inv. Association.addEntry
    1.0% - 43,070 µs - 7,476 inv. java.util.Map.put
      0.9% - 39,651 µs - 7,476 inv. SplayTree.put
        0.8% - 34,062 µs - 7,476 inv. SplayTree.put
          0.3% - 14,747 µs - 7,475 inv. SplayTree$Node.insert
          0.3% - 11,373 µs - 7,476 inv. java.util.Map.put
          0.0% - 11 µs - 1 inv. SplayTree$Node.<init>
  0.3% - 12,370 µs - 7,490 inv. java.io.BufferedReader.readLine
  0.2% - 9,891 µs - 14,974 inv. java.lang.String.replace
  0.1% - 3,459 µs - 11 inv. Association.getSpanishWord
    0.0% - 78 µs - 11 inv. Association.containsWord
    0.0% - 17 µs - 5 inv. java.util.Map.get
    0.0% - 4 µs - 6 inv. java.lang.Object.toString
  0.0% - 1,758 µs - 1 inv. Factory.queMapa
    0.0% - 36 µs - 1 inv. SplayTree.<init>
      0.0% - 8 µs - 1 inv. java.util.TreeMap.<init>
  0.0% - 1,357 µs - 7,487 inv. java.lang.String.charAt
  0.0% - 761 µs - 2 inv. java.io.FileReader.<init>
  0.0% - 284 µs - 4 inv. java.io.PrintStream.println
  0.0% - 78 µs - 1 inv. Factory.<init>
  0.0% - 32 µs - 2 inv. java.io.BufferedReader.<init>
  0.0% - 30 µs - 11 inv. java.lang.String.toLowerCase
  0.0% - 17 µs - 1 inv. java.io.PrintStream.print
  0.0% - 5 µs - 1 inv. Association.<init>
```

Algoritmos y Estructuras de Datos 13-05-2021
Comparación entre implementaciones usadas
Javier Mombiela
Mariano Reyes

Luego de comparadas las implementaciones podemos ver como el SplayTree es un poco más complejo por lo que lleva más tiempo de ejecución con respecto a la otra implementación usada la cual era el Hashing Map.