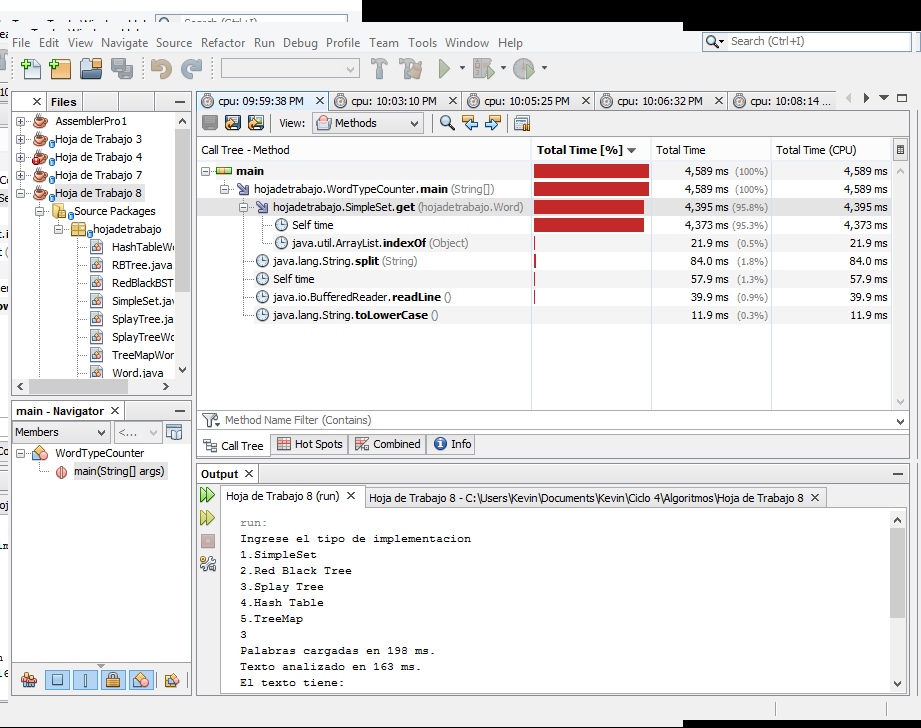
HOJA DE TRABAJO 8

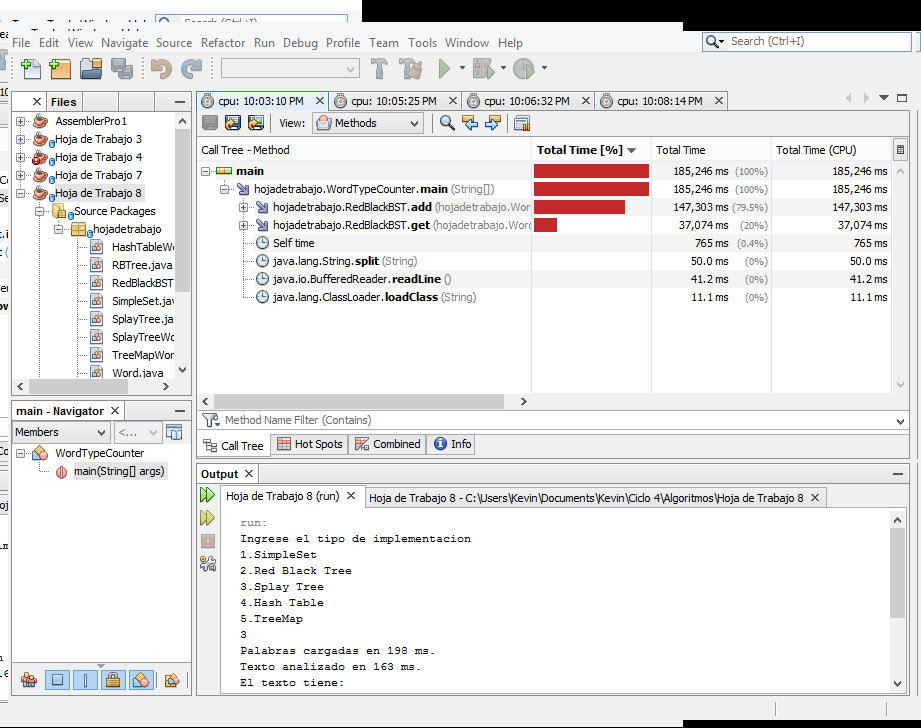
IMPLEMENTACIONES DE ORDENAMIENTO

1. Simple Set



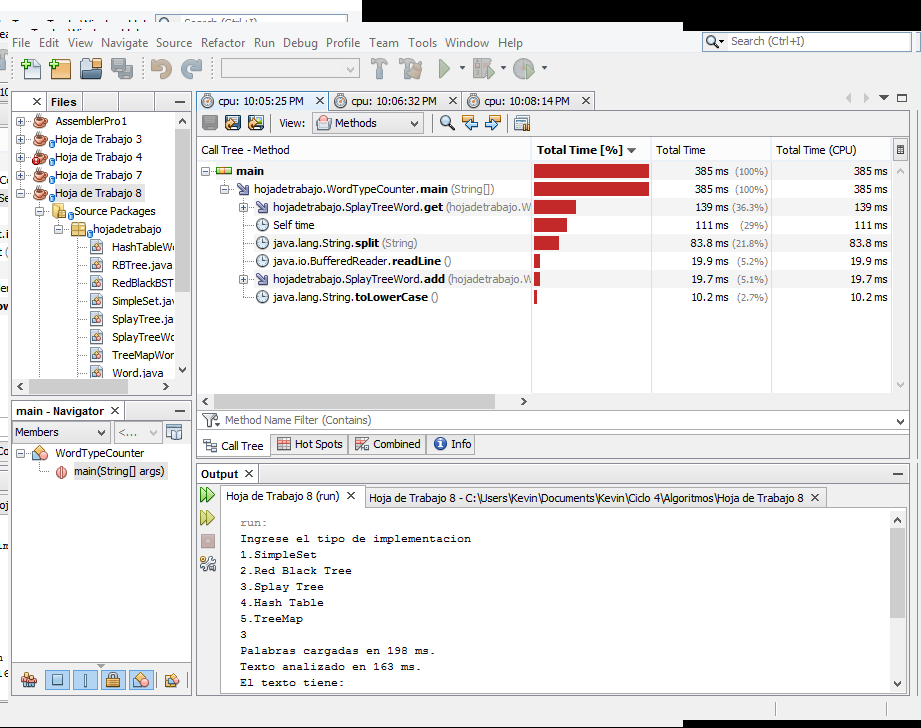
El Simple Set requirió de 4 segundos en total para realizar el proceso. El profiler muestra que la operación más cargada es el **get** con el cual tardó la mayor parte del tiempo, por lo que al momento de añadir las palabras al **ArrayList** no hay problema por si linealidad.

1. Red Black Tree



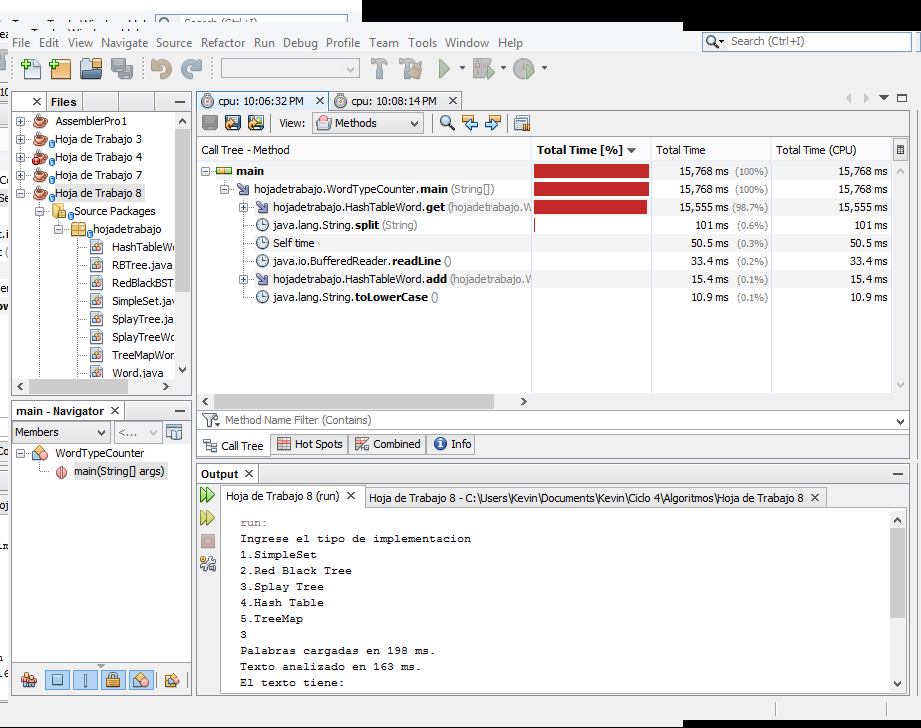
El Red Black Tree requirió 3 minutos y 5 segundos. La mayor carga está en el método de agregar **add**, este método debe recorrer todo el árbol ingresando la llave (**key**) del nodo y el valor tipo **Word**. Al momento de agregar un nodo, un árbol red-black se reordena para mantener balanceado las trayectorias de recorrido. El método de obtención **get** mostró ser significativamente menor que el método de agregado pero aún mucho mayor que el tiempo del método de obtención del **SimpleSet.**

1. Splay Tree



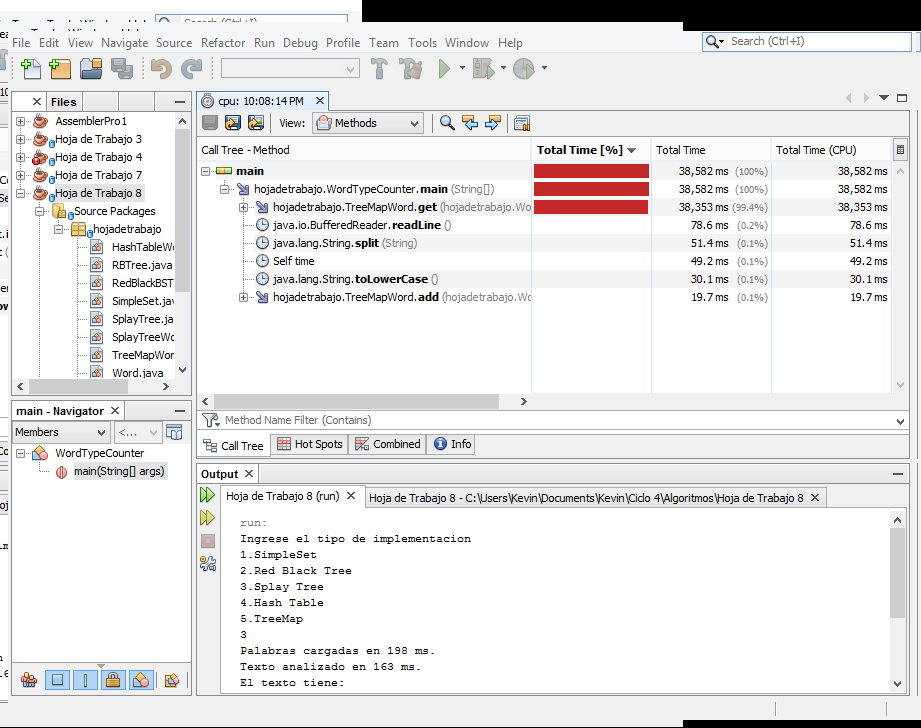
El Splay Tree tomó 385 ms en total. El método de obtención fue el de mayor con 139 ms comparado con el de agregado que solamente tomo 19 ms.

1. Hash Table



El Hash Table tomó 15 segundos en correr. El método de mayor carga fue el de obtención con 15 segundos necesarios para su uso. El método para agregar elementos tomo solo 15ms.

1. TreeMap



El Tree Map tomó 38 segundos en correr. El método de mayor trabajo fue el de obtención con 38 segundos necesarios para su uso. El método de agregado tomo solamente 19 ms.

**Mejor implementación**

A nuestro parecer, la mejor implementación es el SplayTree. Al momento de implementar el SplayTree se tomó la decisión de realizar una rotación hacia la raíz de cada nodo ingresado. Con esto el árbol varía cada vez para que el nuevo nodo ingresado sea la cabeza. Al hacer esto vemos que el profiler determina que el método muestra el menor tiempo de corrida.

No solamente se debe analizar el tiempo de corrida en general, debido a que el algoritmo puede ser utilizado para búsquedas mayores posteriormente, se debe analizar por separado el tiempo de agregado y el tiempo de búsqueda de cada algoritmo. Aun así, vemos que el SplayTree mantiene el menor tiempo de obtención, mientras que el SimpleSet mantiene el menor tiempo de agregado. Para implementaciones mayores, es recomendable tener un algoritmo que se tarde solamente una vez para agregar las palabras a su base de datos pero que su búsqueda sea la más rápida posible.

Debido a ello, proponemos el algoritmo de Splay Tree como el más eficiente al momento de implementarlo con la búsqueda de palabras.