

### Hoja de trabajo No. 8

**Realizar:** Implementación de BST y MAPEO.

**Realizarse:** en grupos de tres alumnos. (**Recuerde colocar en su tarea el nombre y carnet de todos los integrantes**).

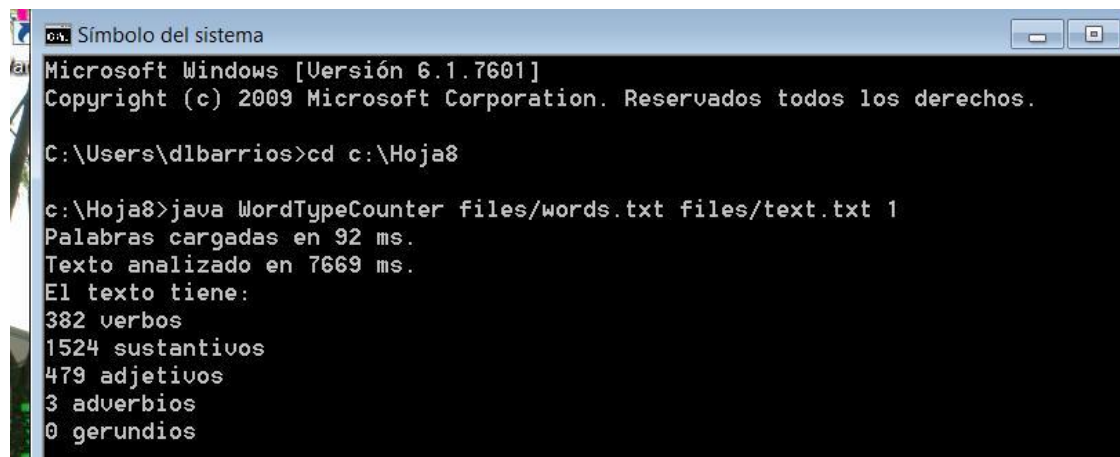
#### Objetivos:

- Implementación de árboles binarios de búsqueda autobalanceados (Spaly Tree y Red Black Tree)
- Implementación de MAPEO (con tabla de hash y con árboles).
- Utilización del patrón de diseño: Factory
- Utilización del Java Collection Framework

#### Programas a mejorar:

- En el .zip adjunto encontrará el programa WordTypeCounter. Este programa cuenta el tipo de palabras en base a lo definido en una lista de palabras que carga (files/words.txt). El programa se ejecuta desde la línea de comando y se le deben pasar 3 parámetros:
  - el archivo con las palabras (files/words.txt)
  - el archivo con el texto a analizar (files/text.txt)
  - la implementación que se desea utilizar (1=original, 2=RBT, 3=SPLAY, 4=Hash Table, 5=Tree Map)

En la siguiente figura se mira la corrida del programa, empleando la implementación = 1, que es la que viene originalmente con el programa:



```
Símbolo del sistema
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\d1barrios>cd c:\Hoja8

c:\Hoja8>java WordTypeCounter files/words.txt files/text.txt 1
Palabras cargadas en 92 ms.
Texto analizado en 7669 ms.
El texto tiene:
382 verbos
1524 sustantivos
479 adjetivos
3 adverbios
0 gerundios
```

#### Tareas:

- Utilizar la implementación de una estructura de datos para mejorar el rendimiento del programa. Debe usar diversas implementaciones para guardar el conjunto de palabras y ser mejor que la implementación que trae el programa original. Las implementaciones que deberá desarrollar o utilizar son:
  - BST que se auto balancean vistos en clase (SPLAY y RedBlackTree) puede consultar el libro de texto, en capítulo 14.5 Splay Tree y capítulo 14.7 Red Black Tree.
  - Mapeo basado en Hash table con Open Addressing. Usar la implementación HashMap del Java Collection Framework.
  - Utilizar la implementación de TreeMap que proporciona el Java Collection Framework.
- Necesitará crear cuatro implementaciones diferentes para la interface WordSet.java (que se proporciona entre los archivos de la tarea) para almacenar el conjunto de palabras.
  - Implementación basada en Red Black Tree (RBT)
  - Implementación basada en Splay Tree
  - Implementación basada en Hash Table usando HashMap del Java Collection Framework
  - Implementación usando TreeMap de Java Collection Framework.

- c. Modificar el archivo WordSetFactory para seleccionar a tiempo de corrida, una de las implementaciones construidas en el paso anterior.
- d. Utilizar un profiler para comparar el desempeño de las estructuras de datos, tanto con la clase original SimpleSet.java que se proporciona, como el resto de las implementaciones construidas.  
NOTA: aunque el programa registra el tiempo de corrida, sabemos que un profiler realiza mejor esta tarea.
- e. Colocar el programa y todas las clases en la plataforma Blackboard. **Solo uno de los participantes debe subirla, pero debe indicar el CARNET y NOMBRE de cada uno de los integrantes del grupo.**
- f. Recuerde guardar todas las clases en su repositorio git (o similar). Envíe el enlace a su repositorio al auxiliar. Se debe notar el trabajo de cada integrante del grupo.
- g. Explicar cuál es la implementación que su grupo propone como la más adecuada para este programa. Razone su respuesta con los tiempos de ejecución logrados, y las complejidades en tiempo y espacio de cada implementación.

NOTA: La clase Word.java que se proporciona con esta tarea **NO DEBE SER MODIFICADA**. Simplemente observe que implementa los métodos equals y compareTo. Esto permite que las clases que usted implemente en el punto **b**, utilicen el “natural ordering” para que cada palabra pueda ser colocada en la posición correcta de su árbol u otras estructuras que usted seleccione.

**Calificación:** su programa debe funcionar para ser calificado.

Aspecto	Puntos
Explicación de la estructura de datos escogida y porque es mejor.	20
Utilización de repositorio de git para guardar las versiones del programa. Debe reflejarse el trabajo de cada miembro del equipo.	10
Cuatro implementaciones realizadas (RBT, SPLAY tree, Hash Table, TreeMap). (10 puntos por cada implementación)	40
Pruebas unitarias para cada una de las implementaciones. Solo se prueban los dos métodos que exige la interface WordSet: add / get (5 puntos por las pruebas de cada implementación)	20
Gráficas del profiler con las implementaciones utilizadas. (Tiempo y Espacio)	10
TOTAL:	100