Arboles Binarios de Búsqueda

Práctica 13
Aplicación de árboles binarios de búsqueda

Objetivo

El objetivo de esta práctica es que el alumno adquiera experiencia en el uso de árboles binarios de búsqueda para la resolución de problemas. En esta práctica se utilizará dicha estructura de datos para implementar un contador de frecuencia de palabras contenidas en un archivo de texto.

Descripción general

En esta práctica se busca implementar un programa para analizar de manera simple y útil el contenido de un archivo de texto. Se implementará un contador de frecuencia de palabras el cual leerá un archivo de texto y determinará las palabras que conforman el archivo, así como la frecuencia con la que aparecen en el mismo. La información generada se presentará al usuario en orden alfabético, mostrando la palabra junto a su frecuencia.

Se considera una palabra como una cadena de caracteres alfanuméricos incluyendo acentos y la letra ñ. Se excluye todo tipo de caracteres de puntuación y espacios en blanco. El programa no distinguirá entre mayúsculas y minúsculas, así que toda palabra leída del archivo se transformará a minúsculas.

Ayuda: Se sugiere hacer uso de la clase de Java *Scanner* para la lectura del archivo y utilizar su método *useDelimiter* que permite especificar los delimitadores deseados para separar los tokens.

Material

El material de esta práctica consta de los siguientes archivos:

- **NodoArbol.class** clase que implementa los nodos de un árbol binario.
- **ArbolBinarioBusqueda.class** clase que implementa la estructura de datos para árboles binarios de búsqueda.
- **Palabra.java** clase que implementará la estructura de una palabra. Se usará para almacenar objetos de esta clase en el árbol binario de búsqueda.

- **ComparaPalabras.java** clase que implementará un comparador para determinar la relación de orden entre objetos de la clase `Palabra`, el cual se hará en orden lexicográfico. Se usará como el comparador requerido por el árbol binario de búsqueda al momento de su inicialización.
- **ContadorFrecuenciaPalabras.java** clase que implementará el contador de frecuencia de palabras encontradas en un archivo.
- **PruebaContadorFrecPalabras.class** programa para probar la clase `ContadorFrecuenciaPalabras`.
- **prueba.txt** archivo de pruebas que se utiliza en la clase `PruebaContadorFrecPalabras`.
- Documentación:
    - NodoArbol.html documentación de la clase `NodoArbol`.
    - ArbolBinarioBusqueda.html documentación de la clase `ArbolBinarioBusqueda`.
    - Scanner.html documentación de la clase Scanner de Java.

Desarrollo

1. Descargar los archivos `NodoArbol.class, ArbolBinarioBusqueda.class, Palabra.java, ComparaPalabras.java, ContadorFrecuenciaPalabras.java, PruebaContadorFrecPalabras.class` y `prueba.txt` en el directorio donde se va a desarrollar la práctica.

2. Leer la documentación para conocer los detalles del material que se proporciona. Recordar que la firma de los métodos proporcionados y requeridos en el desarrollo de la práctica no podrán ser modificados.

3. Dentro de la clase `Palabra` se implementarán los siguientes métodos:

    a. **constructor** que recibe una cadena String correspondiente a una palabra. Inicializa la frecuencia de la palabra con valor cero.
    b. **métodos de acceso**
    c. **incrementarFrec**() método para incrementar en una unidad la frecuencia de la palabra.
    d. **toString**() método para crear una cadena con la palabra y su frecuencia.
    e. **equals**(**Object**) método para determinar que dos palabras son iguales siempre y cuando tengan la misma cadena, sin importar mayúsculas o minúsculas.

4. En la clase `ComparaPalabras` se implementará la interfaz Comparator de Java. Determinará la relación de orden entre dos Palabras de acuerdo al orden lexicográfico de sus cadenas.

5. En la clase `ContadorFrecuenciaPalabras` se implementarán los siguientes métodos:
    a. **constructor** que recibe una cadena String correspondiente al nombre del archivo que será leído. Inicializa un árbol binario de búsqueda con un comparador de tipo `ComparaPalabras`.
    b. **generarFrecuencia**() método que realiza la lectura del archivo. Puebla el árbol binario de búsqueda y realiza el conteo de frecuencia de palabras. Despliega en pantalla las palabras encontradas en orden alfabético junto con su frecuencia.

6. Probar la implementación realizada ejecutando el programa `PruebaContadorFrecPalabras`. Si el programa funciona adecuadamente se verán los siguientes mensajes:

```
Generando frecuencia de palabras del archivo Prueba.txt

a: 1
abandonar: 1
abrir: 1
aceptar: 2
alas: 1
alcanzar: 1
alma: 2
amor: 1
aunque: 6
aún: 5
bajar: 1
benedetti: 1
cada: 1
calle: 2
canto: 1
cedas: 2
celebrar: 1
cerrojos: 1
cielo: 1
cielos: 1
cierto: 1
comenzar: 1
comienzo: 1
```

continuar: 1
correr: 1
cure: 1
de: 3
deseo: 1
desplegar: 1
destapar: 1
destrabar: 1
día: 1
e: 1
el: 19
en: 4
ensayar: 1
enterrar: 1
es: 5
escombros: 1
esconda: 1
eso: 1
esta: 1
estás: 2
existe: 1
extender: 1
favor: 2
frío: 2
fuego: 2
guardia: 1
has: 1
hay: 5
heridas: 1
hora: 1
intentar: 1
la: 7
las: 4
lastre: 1
liberar: 1
lo: 1
los: 3
manos: 1
mario: 1
mejor: 1
miedo: 2
miedos: 1
momento: 1
muerda: 2

```
murallas: 1
no: 10
nuevo: 3
perseguir: 1
ponga: 1
por: 2
porque: 9
protegieron: 1
puertas: 1
que: 3
queme: 2
querido: 1
quiero: 2
quitar: 1
recuperar: 1
reto: 1
retomar: 2
rindas: 5
risa: 1
se: 4
sol: 2
solo: 1
sombras: 1
sueños: 3
también: 1
te: 8
tiempo: 3
tu: 2
tus: 5
tuya: 1
tuyo: 1
un: 2
viaje: 1
vida: 6
viento: 2
vino: 1
vivir: 1
vuelo: 1
y: 11
yo: 1
```

Ejercicio opcional

Como una variación de la aplicación, se propone realizar una implementación para obtener la frecuencia de los siguientes signos de puntuación contenidos en un archivo de texto plano.

Punto	.
Coma	,
Punto y coma	;
Dos puntos	:
Paréntesis	( )

Corchetes	[y]
Llaves	{ y }
Signos de interrogación	¿ y ?
Signos de exclamación	¡ y !
Guion	-