Análisis y diseño de software

dit
UPM

ejercicio 4

José A. Mañas

18.3.2016

ETSIT UPM

ejercicio 4

- algoritmos
  - diccionarios
  - ordenación
- programar y probar

tareas

- programar un contador de palabras en un texto
  1. leer de un fichero llevando la cuenta
  2. ordenar por número de apariciones
  3. extraer algunas conclusiones sobre las palabras usadas

ejemplo de ejecución

- sobre el quijote
  - 384.324 palabras
  - 23.591 palabras diferentes
  - 11.522 palabras se usan 1 sola vez

top(10):
20628: que
18217: de
18189: y
10363: la
9882: a
8242: en
8210: el
6345: no
4748: los
4691: se

top(-10):
1: mereciese
1: curaría
1: acostaráse
1: reduciéndolo
1: leeré
1: efetos
1: apeado
1: fiscal
1: zanoguera
1: anchas

class Registro

```
public class Registro {
    private final String clave;
    private int cnt;

    Registro(String clave) {
        this.clave = clave;
        this.cnt = 1;
    }
```

```
    public String getClave() {
        return clave;
    }

    public int getCnt() {
        return cnt;
    }

    public void inc() {
        this.cnt++;
    }
}
```

class WordCounter (1/4)

```
/**
 * Analizador de textos.
 */
public class WordCounter {

   /**
    * Constructor.
    */
   public WordCounter() { ... }

   /**
    * Carga un fichero de texto.
    * @param file fichero.
    * @throws IOException si hay problemas con el fichero.
    */
   public void load(File file) throws IOException { ... }
```

class WordCounter (2/4)

```
public void load(File file)
        throws IOException {
    diccionario.clear();
    Scanner scanner = new Scanner(file, "UTF-8");
    scanner.useDelimiter("[^\\p{javaLowerCase}\\p{javaUpperCase}]+");
    while (scanner.hasNext()) {
        String word = scanner.next().toLowerCase();
        // cargar la palabra en la tabla 1
    }
    scanner.close();
}
```

class WordCounter (3/4)

```
/**
 * Tamano de la tabla de contadores y del registro de palabras.
 * @return numero de palabras.
 */
public int size() { ... }
```

```
/**
 * Devuelve las n palabras mas usadas (si n es positivo).
 * Devuelve las n palabras menos utilizadas (si n es negativo).
 * @param n
 * @return
 */
public List<Registro> getTop(int n) { ... }
```

class WordCounter (4/4)

```
/**
    * Devuelve cuantas palabras hay por debajo de un umbral c.
    *
    * @param c umbral de cuenta.
    * @return numero de palabras que aparcecen en el texto menos de c veces.
    */
   public int countBelow(int c) { ... }
```

¿cómo se hace?

1. primero, se van leyendo palabras y metiendo en un diccionario de forma que
   - cuando la palabra es nueva, se crea un registro con valor inicial 1,
   - y cuando la palabra ya estaba, se incrementa el contador
2. segundo, se crea un array con todos los registros
   - Registro[] datos = new Registro[size()]
   - se recorre el diccionario y se van pasando los registros

¿cómo se hace?

3. tercero, se ordena el array de registros
   – use un algoritmo eficiente para N grande
4. cuarto, sobre el array ordenado se responde a las preguntas: getTop(), countBelow()

- los pasos 2 y 3 se deben hacer de forma perezosa: cuando se llama al método de uso, si el array no está creado, se crea y se ordena

pruebas

- use ficheros de texto
  - pequeños para validar los resultados a mano
  - grandes para ver que el programa admite volumen

- por ejemplo
  - Don Quijote
    http://www.gutenberg.org/ebooks/2000

entrega

- package es.upm.dit.adsw.ej4
- todas las clases java que haya hecho
  - con los nombres que quiera
  - hay 2 clases con nombre fijo

    Registro.java

    WordCounter.java