



MANEJO DE CADENAS

Java Básico

Ejercicio.

2

- ❑ Lo han contratado en la Real Academia de la lengua Española.
- ❑ Le han solicitado que realice un programa que entregue estadísticas acerca del uso del lenguaje.
- ❑ El programa debe leer archivos de texto y entregar gráficas sobre la cantidad de palabras, cantidad de letras, longitud de las palabras, longitud de los enunciados, etc.



Análisis

3

- Se requiere:
 - ▣ Leer archivos de texto.
 - ▣ Separar texto en enunciados.
 - ▣ Contar palabras y letras.
 - ▣ Graficar resultados (modo texto).

API

4

- File
 - Representa un archivo o un directorio de la máquina.
- FileReader
 - Objeto que puede leer un archivo
- Scanner
 - Lee flujos de datos
- String
 - Representa una cadena de texto

Clase File

5

- <http://docs.oracle.com/javase/7/docs/api/java/io/File.html>

Constructors

Constructor and Description

`File(File parent, String child)`

Creates a new `File` instance from a parent abstract pathname and a child pathname string.

`File(String pathname)`

Creates a new `File` instance by converting the given pathname string into an abstract pathname.

`File(String parent, String child)`

Creates a new `File` instance from a parent pathname string and a child pathname string.

`File(URI uri)`

Creates a new `File` instance by converting the given `file:` URI into an abstract pathname.

Clase File

6

Methods

Modifier and Type	Method and Description
boolean	<code>isAbsolute()</code> Tests whether this abstract pathname is absolute.
boolean	<code>isDirectory()</code> Tests whether the file denoted by this abstract pathname is a directory.
boolean	<code>isFile()</code> Tests whether the file denoted by this abstract pathname is a normal file.
boolean	<code>isHidden()</code> Tests whether the file named by this abstract pathname is a hidden file.

Clase Scanner

7

- <http://docs.oracle.com/javase/7/docs/api/java/util/Scanner.html>

For example, this code allows a user to read a number from `System.in`:

```
Scanner sc = new Scanner(System.in);  
int i = sc.nextInt();
```

As another example, this code allows `long` types to be assigned from entries in a file `myNumbers`:

```
Scanner sc = new Scanner(new File("myNumbers"));  
while (sc.hasNextLong()) {  
    long aLong = sc.nextLong();  
}
```

The scanner can also use delimiters other than whitespace. This example reads several items in from a string:

```
String input = "1 fish 2 fish red fish blue fish";  
Scanner s = new Scanner(input).useDelimiter("\\s*fish\\s*");  
System.out.println(s.nextInt());  
System.out.println(s.nextInt());  
System.out.println(s.next());  
System.out.println(s.next());  
s.close();
```


Clase Scanner

8

Constructors

Constructor and Description

`Scanner(File source)`

Constructs a new `Scanner` that produces values scanned from the specified file.

`Scanner(File source, String charsetName)`

Constructs a new `Scanner` that produces values scanned from the specified file.

`Scanner(InputStream source)`

Constructs a new `Scanner` that produces values scanned from the specified input stream.

`Scanner(InputStream source, String charsetName)`

Constructs a new `Scanner` that produces values scanned from the specified input stream.

`Scanner(Path source)`

Constructs a new `Scanner` that produces values scanned from the specified file.

`Scanner(Path source, String charsetName)`

Constructs a new `Scanner` that produces values scanned from the specified file.

`Scanner(Readable source)`

Constructs a new `Scanner` that produces values scanned from the specified source.

`Scanner(ReadableByteChannel source)`

Constructs a new `Scanner` that produces values scanned from the specified channel.

`Scanner(ReadableByteChannel source, String charsetName)`

Constructs a new `Scanner` that produces values scanned from the specified channel.

`Scanner(String source)`

Constructs a new `Scanner` that produces values scanned from the specified string.

Clase Scanner

9

Methods

Modifier and Type	Method and Description
<code>Scanner</code>	<code>useDelimiter (Pattern pattern)</code> Sets this scanner's delimiting pattern to the specified pattern.
<code>Scanner</code>	<code>useDelimiter (String pattern)</code> Sets this scanner's delimiting pattern to a pattern constructed from the specified String.
<code>String</code>	<code>next ()</code> Finds and returns the next complete token from this scanner.
<code>boolean</code>	<code>hasNext ()</code> Returns true if this scanner has another token in its input.

- Nombre:
 - ▣ RAESTatistics

- Agregar variable estática

```
public class RAESTatistics {  
  
    static int letras [] = new int [300];  
  
    public static void main(String[] args)
```

Agregamos un método.

11

```
private static void procesaPalabra (String palabra) {  
  
    System.out.println ("Procesando: "+palabra);  
  
    for (int i = 0; i<palabra.length(); i++) {  
        char letra = palabra.charAt(i);  
        if (letra < letras.length) {  
            letras[letra] ++;  
        }  
    }  
}
```

```
public static void main(String[] args) {  
  
    procesaPalabra("Mi mama me mima");  
    for (int i = 0; i < letras.length; i++) {  
        if (letras[i] != 0) {  
            System.out.println "["+(char)i+"]"+"="+letras[i]);  
        }  
    }  
}
```



Ejecutar

```
private static void procesaEnunciado(String enunciado) {  
  
    Scanner escaner = new Scanner (enunciado);  
    escaner.useDelimiter("[ ,!;'¿?:;\t\n]");  
    while (escaner.hasNext()) {  
        System.out.println (escaner.next());  
    }  
}
```








```
public static void main(String[] args) {  
  
    procesaEnunciado("Más vale pajarero en mano que sienta volando");  
  
}
```



Ejecutar

Copiamos el archivo en la ruta del proyecto

15

	build	29/01/2013 11:37 a...	Carpeta de archivos	
	lib	29/01/2013 09:46 a...	Carpeta de archivos	
	nbproject	29/01/2013 09:46 a...	Carpeta de archivos	
	src	29/01/2013 09:46 a...	Carpeta de archivos	
	build.xml	29/01/2013 09:46 a...	Documento XML	4 KB
	manifest.mf	29/01/2013 09:46 a...	Archivo MF	1 KB
	Quijote.txt	01/02/2013 02:50 ...	Archivo TXT	11 KB

Agregamos otro método

16

```
public static void main(String[] args)
    throws FileNotFoundException {

    procesaArchivo (new File ("Quijote.txt"));
}

private static void procesaArchivo(File selectedFile)
    throws FileNotFoundException {
    Scanner escaner = new Scanner (new FileReader (selectedFile));
    escaner.useDelimiter("\\\\.");
    while (escaner.hasNext()){
        System.out.println (escaner.next());
    }
}
```



Ejecutar

```
public class RAESStatistics {  
  
    static int numEnunciados = 0;  
    static int longEnunciados = 0;  
    static int numPalabras = 0;  
    static int longPalabras = 0;  
  
    static int letras [] = new int [300];  
  
    public static void main(String[] args)  
        throws FileNotFoundException {
```

```
private static void procesaArchivo(File selectedFile)
    throws FileNotFoundException {
    Scanner escaner = new Scanner (new FileReader (selectedFile));
    escaner.useDelimiter("\\.");
    while (escaner.hasNext()) {
        procesaEnunciado(escaner.next());
    }
}

private static void procesaEnunciado(String enunciado) {
    numEnunciados++;
    longEnunciados += enunciado.length();
    Scanner escaner = new Scanner (enunciado);
    escaner.useDelimiter("[ ,!;'?:;\\t\\n]");
    while (escaner.hasNext()) {
        procesaPalabra(escaner.next());
    }
}
```

```
private static void procesaPalabra(String palabra) {  
  
    numPalabras++;  
    longPalabras += palabra.length();  
    System.out.println ("Procesando: "+palabra);  
  
    for (int i = 0; i<palabra.length(); i++){  
        char letra = palabra.charAt(i);  
        if (letra < letras.length){  
            letras[letra] ++;  
        }  
    }  
}
```

Modificando el main

20

```
public static void main(String[] args)
    throws FileNotFoundException {

    procesaArchivo(new File ("Quijote.txt"));

    System.out.println ("Numero de enunciados:"+numEnunciados );
    System.out.println ("Numero de palabras:"+numPalabras );
    System.out.println ("Longitud promedio de los enunciados:"
        +(longEnunciados / numEnunciados ));
    System.out.println ("Longitud promedio de las palabras:"
        +(longPalabras / numPalabras) );
    System.out.println ("Cantidad de letras:");

    for (int i = 0; i < letras.length; i++){
        if (letras[i] != 0){
            System.out.println("[ "+(char)i+" ] "+"=" +letras[i]);
        }
    }
}
```



Ejecutar


```
public static void main(String[] args)
    throws FileNotFoundException {

    JFileChooser jfc = new JFileChooser ();
    if (jfc.showOpenDialog(null) == JFileChooser.APPROVE_OPTION) {
        procesaArchivo (jfc.getSelectedFile());
    }

    System.out.println ("Numero de enunciados:"+numEnunciados );
}
```



Ejecutar

Ejercicios

22

- ❑ Elimina los saltos de línea antes de procesar los enunciados.
 - ❑ Revisa el API de String, particularmente el método `replaceAll`.
 - ❑ Un salto de línea se puede escribir como: `'\n'`
- ❑ Cuenta como la misma letra las mayúsculas y minúsculas.
 - ❑ Puedes convertir la cadena en minúscula antes de enviarla a procesar.
- ❑ Agrega una validación, si la palabra está vacía, no la proceses.
 - ❑ Puedes eliminar los espacios con `trim`.
 - ❑ Para saber si una cadena está vacía puedes usar el método `equals("")`.



Ejercicios más complicados



23

- Calcula cuantas palabras en promedio tienen los enunciados.
- Has un conteo de cada palabra, como se hizo con las letras.
- Grafica con asteriscos la cantidad de letras y palabras. (Usa porcentajes, 20 asteriscos es 100%)

