Programación Orientada a Objetos. Práctica 4.2

Tema 4. Clases Básicas de Java y Entrada/Salida

Ejercicio 1. (proyecto prCuentaPalabrasSimpleFicheros)

Se va a crear una aplicación para contar el número de veces que aparece cada palabra en un texto dado. Para ello se crearán las clases `PalabraEnTexto`, `ContadorPalabras` y `ContadorPalabrasSig`.

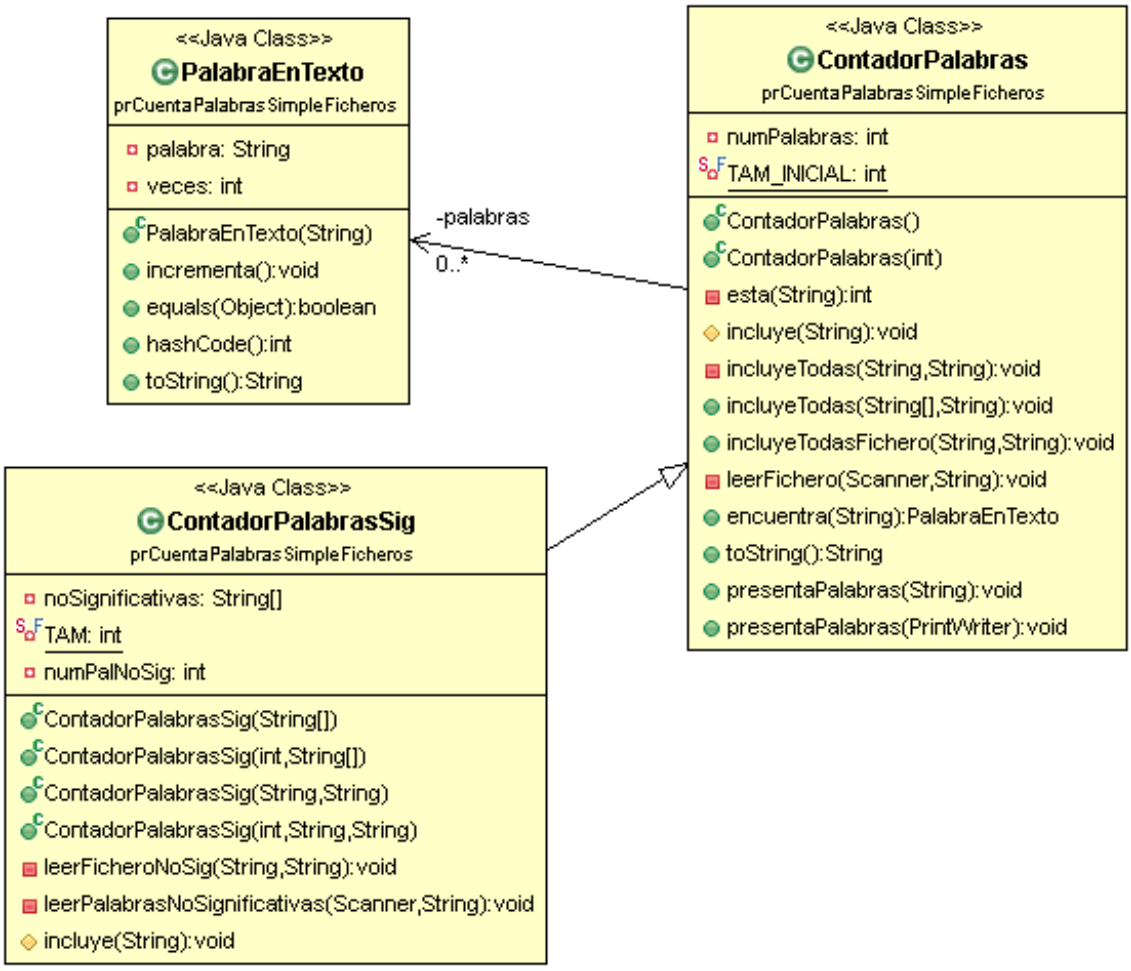

Figura 1: Diagrama de clases UML

Clase `PalabraEnTexto`

Crea la clase `PalabraEnTexto` para mantener información de una palabra (`String`), así como del número de veces que aparece en un determinado texto (`int`).

1. La clase tendrá un constructor en el que se proporciona la palabra. Al construir el objeto, el número de veces que aparece la palabra se considera 1. Además, la palabra se almacenará en mayúsculas.
2. Dos objetos de la clase `PalabraEnTexto` son iguales si coinciden las palabras que contiene. El número de apariciones no se tiene en cuenta.
3. La representación de un objeto `PalabraEnTexto` debe mostrar la palabra que contiene y el número de veces que aparece, por ejemplo, `GORRA: 2`.
4. El método `void incrementa()` incrementa en uno el número de veces que aparece la palabra.

Aplicación `PruebaPalabraEnTexto`

Crea una aplicación (clase distinguida `PruebaPalabraEnTexto`) para probar la clase anterior. En esta aplicación se crean dos objetos `PalabraEnTexto` con las palabras `gorra` y `Gorra`. Después se incrementa una vez el número de apariciones de la primera, y se muestra por pantalla el contenido de ambos objetos. Por último, se comprueba si ambas palabras son iguales, indicándolo por pantalla. La ejecución de la aplicación producirá la siguiente salida por pantalla:

```
Palabra 1 = GORRA: 2
Palabra 2 = GORRA: 1
Las palabras son iguales
```

Clase `ContadorPalabras`

Crea la clase `ContadorPalabras` que almacena en un array de `PalabraEnTexto` las palabras que aparecen en un texto. También guardará el número de palabras (`int`) almacenadas en un momento dado.

Si en el transcurso de las operaciones que se hagan con objetos de esta clase, el array llega a tener un tamaño insuficiente, deberá crecer de manera que siempre quepan las palabras que se le proporcionen.

1. La clase dispondrá de dos constructores; el primero de ellos, sin argumentos, que crea el array con un tamaño de 10 (constante `TAM_INICIAL`); el segundo, con un argumento entero, que indicará el tamaño inicial del array. En ambos casos, el número de palabras almacenadas será 0.

   - `public ContadorPalabras();`
   - `public ContadorPalabras(int);`

2. El siguiente método privado devuelve la posición en la que se encuentra la palabra que corresponde a `pal` en el array o -1 si no está.

   - `private int esta(String pal);`

3. El siguiente método protegido deberá incrementar el número de apariciones de la palabra que corresponda a la cadena `pal` en el contador de palabras si es que ya existía, o incluir una palabra nueva en caso contrario.

   - `protected void incluye(String pal);`

4. El siguiente método privado permite extraer de `linea` las palabras usando los delimitadores incluidos en `del`. Cada una de las palabras obtenidas se irán acumulando en el contador.

   - `private void incluyeTodas(String linea, String del);`

5. El siguiente método público incluye todas las palabras que se encuentran en el array `texto`. Cada elemento del array será una línea de texto y en cada línea, las palabras se deben separar usando los delimitadores incluidos en `del`.

   - `public void void incluyeTodas(String[] texto, String del);`

6. El siguiente método público incluye todas las palabras que se encuentran en el fichero. Cada elemento del fichero será una línea de texto y en cada línea, las palabras se deben separar usando los delimitadores incluidos en `del`. Este método crea un flujo de entrada (`Scanner`) e invoca al método privado `leerFichero()` que llevará a cabo la lectura del fichero línea a línea.

   - `public void incluyeTodasFichero(String nomFich, String del);`
   - `private void leerFichero(Scanner sc, String del);`

7. El siguiente método público que, dada una cadena de caracteres `pal` que representa una palabra, encuentra la instancia de `PalabraEnTexto` en el array que coincide con ella y la devuelve. Si la palabra no se encuentra en el texto deberá lanzar la excepción `NoSuchElementException`.

   - `public PalabraEnTexto encuentra(String pal);`

8. La clase dispondrá de una representación de los objetos como la que se muestra en el ejemplo final. Usar `StringJoiner` o `StringBuilder` para crear la representación, y obsérvese que, tras la última palabra, no hay coma.

9. La clase además dispondrá de los dos siguientes métodos públicos que generarán una presentación del índice en el siguiente formato:

   ```
   GUERRA: 5
   TENÍA: 2
   UNA: 2
   JARRA: 3
   Y: 1
   ...
   ```

   Uno de los métodos recibirá como parámetro el nombre del fichero (de tipo `String`) donde almacenar la información y el otro recibirá como parámetro el flujo de salida (de tipo `PrintWriter`) donde llevar a cabo la acción.

   - `public void presentaPalabras(String fichero);`
   - `public void presentaPalabras(PrintWriter pw);`

Aplicación `PruebaContadorPalabras`

Crea una aplicación (clase distinguida `PruebaContadorPalabras`) para probar la clase anterior. En esta aplicación se crea un objeto de la clase `ContadorPalabras` con un array de tamaño 5. Posteriormente se invoca a su método `incluyeTodas()` pasándole como parámetros:

- Como primer parámetro, el siguiente array:

  ```
  String [] datos = {
      "Esta es la primera frase del ejemplo",
      "y esta es la segunda frase"
  };
  ```

- Como segundo parámetro, la siguiente cadena (el espacio en blanco es el único delimitador) :

  - `"[ ]"`

Para terminar, la aplicación mostrará por pantalla el contenido del objeto creado. La ejecución de la aplicación producirá la siguiente salida por pantalla:

```
[ESTA: 2, ES: 2, LA: 2, PRIMERA: 1, FRASE: 2, DEL: 1, EJEMPLO: 1, Y: 1, SEGUNDA: 1]
```

Clase `ContadorPalabrasSig`

Crea la clase `ContadorPalabrasSig` que representa objetos contadores de palabras que, en los procedimientos de inclusión, no incluyen las palabras consideradas **"no significativas"**. Para ello, la clase deberá contener un array de `String` (`noSignificativas`) que almacene estas palabras no significativas (se almacenarán en **mayúsculas**), así como el número de palabras (`numPalNoSig`) almacenadas en él (de tipo `int`)

1. Definir los dos constructores siguientes. El primero recibe el tamaño inicial (`n`) del array de `PalabraEnTexto` y un array de `String` (`palsNS`) con las palabras no significativas. El segundo sólo recibe este array. Las palabras no significativas que vengan en este array parámetro (`palsNS`) deberán almacenarse en mayúsculas en el objeto (en el array `noSignificativas`). La variable `numPalNoSig` se inicializará al tamaño de estos arrays.

   - `public ContadorPalabrasSig(int n, String[] palsNS);`
   - `public ContadorPalabrasSig(String[] palsNS);`

2. Definir los dos constructores siguientes, para permitir que la relación de palabras no significativas sea obtenida desde un fichero. Estos constructores no reciben como parámetro un array con las palabras no significativas, sino que reciben un parámetro de tipo `String` con el nombre del fichero

de entrada que contendrá la relación de palabras no significativas, y otro parámetro también de tipo `String` que contendrá la cadena con los caracteres delimitadores de dichas palabras en el fichero. En ambos casos, la variable `noSignificativas` se instanciará con un array de `String` de tamaño `TAM`, y la variable `numPalNoSig` se inicializará a 0.

- `public ContadorPalabrasSig(String filNoSig, String del);`
- `public ContadorPalabrasSig(int n, String filNoSig, String del);`

Ambos constructores llamarán al método privado `leerFicheroNoSig()` que crea un flujo de entrada (`Scanner`), que a su vez llama al método privado `leerPalabrasNoSignificativas()`, que recibe como parámetro el flujo de entrada (`Scanner`) y que llevará a cabo la lectura del fichero palabra a palabra, rellenando el array `noSignificativas` (al igual que en el caso 1, las palabras que se obtengan del fichero se almacenarán en mayúsculas). Si se considera necesario, se pueden utilizar más métodos privados para la implementación de los pedidos.

- `private void leerFicheroNoSig(String filNoSig, String del);`
- `private void leerPalabrasNoSignificativas(Scanner sc, String del);`

En el fichero de palabras no significativas, las palabras no significativas se encuentran organizadas en líneas, es decir, puede haber multiples líneas con las palabras, donde cada línea, tiene a su vez palabras que estarán separadas por los delimitadores.

3. Conseguir que las instancias de la clase `ContadorPalabrasSig` se comporten como las de `ContadorPalabras`, a excepción de que los procedimientos de inclusión de palabras no realicen ninguna acción cuando éstas no sean significativas.

Aplicación `Main`

Aquí se presenta un ejemplo de uso más completo de todas las clases y la salida correspondiente:

```
import java.util.NoSuchElementException;
import prCuentaPalabrasSimpleFicheros.*;
import java.io.*;
public class Main {
    public static void main(String [] args) {
        String [] datos = {
            "Guerra tenía una jarra y Parra tenía una perra, ",
            "pero la perra de Parra rompió la jarra de Guerra.",
            "Guerra pegó con la porra a la perra de Parra. ",
            "¡Oiga usted buen hombre de Parra! ",
            "Por qué ha pegado con la porra a la perra de Parra.",
            "Porque si la perra de Parra no hubiera roto la jarra de Guerra,",
            "Guerra no hubiera pegado con la porra a la perra de Parra."
        };
        String delimitadores = "[ .,:;\\-\\!\\¡\\¿\\?]+"; // " .,:;-!¡¿?" una o varias apariciones
        String [] noSig = {"Con", "La", "A", "De", "NO", "SI", "y", "una"};
        ContadorPalabras contador = null, contadorSig = null;
        // Si no se incluye un argumento numérico, se crea por defecto.
        try {
            int n = Integer.parseInt(args[0]);
            System.out.println("Con argumento " + n);
            contador = new ContadorPalabras(n);
            contadorSig = new ContadorPalabrasSig(n, noSig);
        } catch (RuntimeException e) {
            System.out.println("Por defecto...");
            contador = new ContadorPalabras();
            contadorSig = new ContadorPalabrasSig(noSig);
        }
        // Incluimos todas las palabras que hay en datos
        // teniendo en cuenta los delimitadores
        contador.incluyeTodas(datos, delimitadores);
        contadorSig.incluyeTodas(datos, delimitadores);
```

```
        System.out.println(contador + "\n");
        System.out.println(contadorSig + "\n");
        try {
            System.out.println(contador.encuentra("parra"));
            System.out.println(contador.encuentra("Gorra"));
        } catch (NoSuchElementException e) {
            System.err.println(e.getMessage());
        }
        //Repetimos la salida con E/S desde ficheros
        System.out.println("Repetimos la ejecución tomando la E/S desde/a fichero");
        ContadorPalabras contadorSigFich = null;
        // Si no se incluye un argumento numérico, se crea por defecto.
        try {
            int n = Integer.parseInt(args[0]);
            System.out.println("Con argumento " + n);
            contador = new ContadorPalabras(n);
            contadorSigFich = new ContadorPalabrasSig(n, "fichNoSig.txt", delimitadores);
        } catch (RuntimeException e) {
            System.out.println("Por defecto...");
            contador = new ContadorPalabras();
            try {
                contadorSigFich= new ContadorPalabrasSig("fichNoSig.txt", delimitadores);
            } catch (IOException e1) {
                //
                System.out.println("ERROR:"+ e1.getMessage());
            }
        } catch (IOException e) {
            System.out.println("ERROR:"+ e.getMessage());
        }
        // Incluimos todas las palabras que hay en datos.txt teniendo en cuenta los separadores
        try {
            contador.incluyeTodasFichero("datos.txt", delimitadores);
            contadorSigFich.incluyeTodasFichero("datos.txt", delimitadores);
            System.out.println(contador + "\n");
            System.out.println(contadorSigFich + "\n");
            //métodos para presentar por pantalla
            PrintWriter pw = new PrintWriter(System.out, true);
            contador.presentaPalabras(pw);
            //salida a fichero
            contador.presentaPalabras("salida.txt");
            //métodos para presentar por pantalla para No Significativas
            System.out.println();
            contadorSigFich.presentaPalabras(pw);
            //salida a fichero
            contadorSigFich.presentaPalabras("salidaNoSig.txt");
        } catch (IOException e) {
            System.out.println("ERROR:"+ e.getMessage());
        }
    }
}
```

Los ficheros `datos.txt` y `fichNoSig.txt` deberán ser copiados a la carpeta raíz (carpeta base) del proyecto, en el espacio de trabajo del alumno. **Nota**: al copiar el contenido de los ficheros del documento PDF, las letras con tildes se copian con una codificación **errónea**.

El fichero `datos.txt` contiene la siguiente información:

```
Guerra tenía una jarra y Parra tenía una perra,
pero la perra de Parra rompió la jarra de Guerra.
Guerra pegó con la porra a la perra de Parra.
¡Oiga usted buen hombre de Parra!
Por qué ha pegado con la porra a la perra de Parra.
```

```
Porque si la perra de Parra no hubiera roto la jarra de Guerra,
Guerra no hubiera pegado con la porra a la perra de Parra.
```

El fichero `fichNoSig.txt` contiene la siguiente información:

```
Con La A De NO SI y una
```

A continuación se presenta la salida correspondiente a la clase Main:

```
Por defecto...
[GUERRA: 5, TENÍA: 2, UNA: 2, JARRA: 3, Y: 1, PARRA: 7, PERRA: 6, PERO: 1, LA: 10, DE: 8, ROMPIÓ: 1,
PEGÓ: 1, CON: 3, PORRA: 3, A: 3, OIGA: 1, USTED: 1, BUEN: 1, HOMBRE: 1, POR: 1, QUÉ: 1, HA: 1,
PEGADO: 2, PORQUE: 1, SI: 1, NO: 2, HUBIERA: 2, ROTO: 1]

[GUERRA: 5, TENÍA: 2, JARRA: 3, PARRA: 7, PERRA: 6, PERO: 1, ROMPIÓ: 1, PEGÓ: 1, PORRA: 3, OIGA: 1,
USTED: 1, BUEN: 1, HOMBRE: 1, POR: 1, QUÉ: 1, HA: 1, PEGADO: 2, PORQUE: 1, HUBIERA: 2, ROTO: 1]

PARRA: 7
No existe la palabra Gorra
Repetimos la ejecución tomando la E/S desde/a fichero
Por defecto...
[GUERRA: 5, TENÍA: 2, UNA: 2, JARRA: 3, Y: 1, PARRA: 7, PERRA: 6, PERO: 1, LA: 10, DE: 8, ROMPIÓ: 1,
PEGÓ: 1, CON: 3, PORRA: 3, A: 3, OIGA: 1, USTED: 1, BUEN: 1, HOMBRE: 1, POR: 1, QUÉ: 1, HA: 1,
PEGADO: 2, PORQUE: 1, SI: 1, NO: 2, HUBIERA: 2, ROTO: 1]

[GUERRA: 5, TENÍA: 2, JARRA: 3, PARRA: 7, PERRA: 6, PERO: 1, ROMPIÓ: 1, PEGÓ: 1, PORRA: 3, OIGA: 1,
USTED: 1, BUEN: 1, HOMBRE: 1, POR: 1, QUÉ: 1, HA: 1, PEGADO: 2, PORQUE: 1, HUBIERA: 2, ROTO: 1]

GUERRA: 5
TENÍA: 2
UNA: 2
JARRA: 3
Y: 1
PARRA: 7
PERRA: 6
PERO: 1
LA: 10
DE: 8
ROMPIÓ: 1
PEGÓ: 1
CON: 3
PORRA: 3
A: 3
OIGA: 1
USTED: 1
BUEN: 1
HOMBRE: 1
POR: 1
QUÉ: 1
HA: 1
PEGADO: 2
PORQUE: 1
SI: 1
NO: 2
HUBIERA: 2
ROTO: 1

GUERRA: 5
TENÍA: 2
JARRA: 3
PARRA: 7
PERRA: 6
PERO: 1
```

ROMPIÓ: 1
PEGÓ: 1
PORRA: 3
OIGA: 1
USTED: 1
BUEN: 1
HOMBRE: 1
POR: 1
QUÉ: 1
HA: 1
PEGADO: 2
PORQUE: 1
HUBIERA: 2
ROTO: 1