Programación Avanzada I. Práctica 4.2

Tema 4. Clases Básicas de Java y Entrada/Salida

Ejercicio 1. (proyecto pa1p42, paquete cuentapalabras)

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` en el paquete `cuentapalabras`.

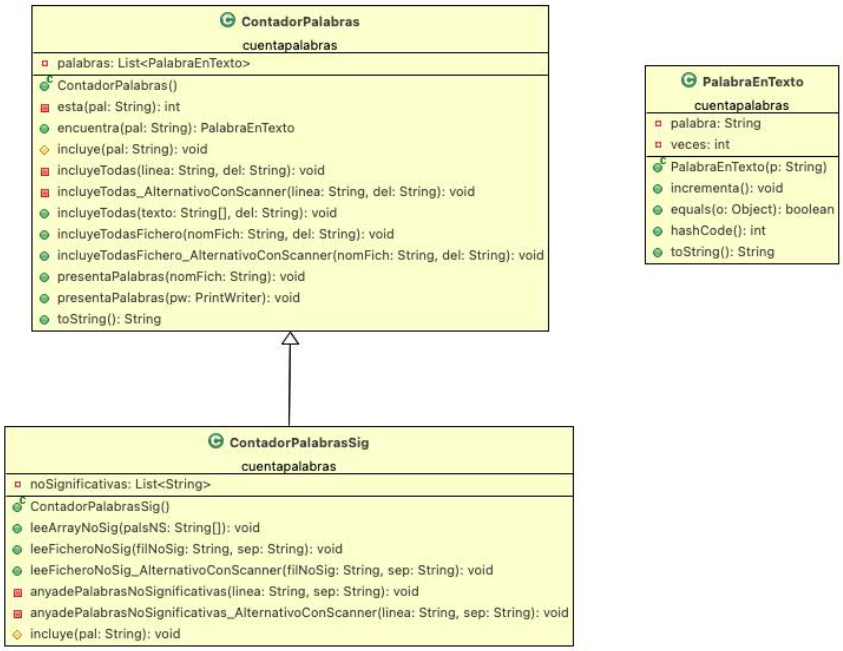

Figura 1: Diagrama de clases UML

Clase `PalabraEnTexto`

Crea la clase `PalabraEnTexto` para mantener información de una palabra (`String`), así como del número de veces que aparece esa palabra 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. Nótese que las palabras están almacenadas en mayúsculas.
3. La representación textual 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 una lista de `PalabraEnTexto` las palabras que aparecen en un texto.

1. La clase dispondrá de un constructor que crea la lista de palabras vacía.

   - `public ContadorPalabras();`

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

   - `private int esta(String pal);`

   Debe tenerse en cuenta que el parámetro es de tipo `String`, pero que los elementos almacenados en la lista son de tipo `PalabraEnTexto`, por lo que, para compararlos con el método `equals`, es necesario crear un objeto *auxiliar* de tipo `PalabraEnTexto` conteniendo la palabra recibida como parámetro.

3. El siguiente método **protegido** deberá incrementar el número de apariciones de la palabra (en la lista) que corresponda a la cadena `pal` en el contador de palabras si es que ya existía, o añadir una palabra nueva en caso contrario. Nótese que si la palabra recibida como parámetro está vacía, entonces no se hará nada.

   - `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, invocando al método `incluye`.

   - `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 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`.

   - `public void incluyeTodasFichero(String nomFich, String del); // throws IOException`

7. El siguiente método público que, dada una cadena de caracteres `pal` que representa una palabra, encuentra la instancia de `PalabraEnTexto` en la lista 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 siguiente ejemplo. Usar `StringJoiner` o `StringBuilder` para crear la representación, y obsérvese que, tras la última palabra, no hay guión.

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

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); // throws FileNotFoundException`
   - `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`. 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 una lista de `String` (`noSignificativas`) que almacene estas palabras no significativas (se almacenarán en **mayúsculas**).

1. El constructor contruye el objeto con una lista de palabras no significativas (`noSignificativas`) vacía, y una lista de palabras vacía.
2. Define el método `leeArrayNoSig` que recibe un array de `String` (`palsNS`) con las palabras no significativas. Las palabras que hubiese en la lista `noSignificativas` serán eliminadas. A continuación, las palabras no significativas recibidas en el parámetro (`palsNS`) deberán almacenarse en mayúsculas en la lista `noSignificativas`. Si alguna palabra está vacía, entonces será ignorada.

   - `public void leeArrayNoSig(String[] palsNS);`
3. Define el método `leeFicheroNoSig` para permitir que la relación de palabras no significativas sea obtenida desde un fichero. Este método recibe 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.

   - `public void leeFicheroNoSig(String filNoSig, String del); // throws IOException`

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

   Al igual que en el caso 1, las palabras que hubiese en la lista `noSignificativas` serán eliminadas. A continuación, las palabras que se obtengan del fichero se almacenarán en mayúsculas en la lista `noSignificativas`. Para ello, por cada línea leída del fichero, invoca al método privado `anyadePalabrasNoSignificativas`.
4. El siguiente método privado permite extraer de `linea` las palabras usando los delimitadores incluidos en `del`. Cada una de las palabras obtenidas se añade en mayúsculas a la lista de palabras no significativas. Si alguna palabra está vacía, entonces será ignorada.

   - `private void anyadePalabrasNoSignificativas(String linea, String del);`
5. Consigue que las instancias de la clase `ContadorPalabrasSig` se comporten como las de `ContadorPalabras`, a excepción de que los procedimientos de inclusión de palabras (debe redefinir el método protegido `incluye`) no realicen ninguna acción cuando las palabras a incluir no sean significativas. Para ello, invoca al método `contains` de listas.

Aplicación `Main`

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

```java
import cuentapalabras.*;
import java.util.NoSuchElementException;
import java.io.PrintWriter;
import java.io.IOException;
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"};

        System.out.println("ContadorPalabras ...");
        ContadorPalabras contador = new ContadorPalabras();
        ContadorPalabrasSig contadorSig = new ContadorPalabrasSig();
        contadorSig.leeArrayNoSig(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");
        ContadorPalabrasSig contadorSigFich = null;
        try {
            contador = new ContadorPalabras();
            contadorSigFich = new ContadorPalabrasSig();
            contadorSigFich.leeFicheroNoSig("fichNoSig.txt", delimitadores);
        } 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 (se han añadido saltos de línea para mejorar la legibilidad):

```
ContadorPalabras ...
[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
Repetimos la ejecución tomando la E/S desde/a fichero
No existe la palabra Gorra
[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