


TECNOLÓGICO NACIONAL DE MEXICO

INSTITUTO TECNOLÓGICO DE CIUDAD MADERO

Carrera: Sistemas Computacionales

Tema: Programa que cuente las líneas y palabras de un archivo de texto.

Alumno: Reyes Villar Luis Ricardo | 21070343

Profesor: Armando Becerra del Ángel

Materia: Lenguajes y Autómatas 1

Hora: 15:00 – 16:00 hrs

Grupo: 6501-B

Semestre: Agosto 2023 – Diciembre 2023

Clase LineasYPalabras

```java
package clases;
// @author LuisR
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;

public class LineasYPalabras {

    private final File archivo;

    private int numLineas;
    private int numPalabras;

    private final StringBuilder textoArchivo;

    private String textoArc;

    /*
     *  El constructor recibe un parámetro de tipo File, el cual es el archivo
     *  seleccionado por el usuario, posteriormente se instancia el atributo
     *  de tipo StringBuilder.
     */
    public LineasYPalabras(File archivo) {

        this.archivo = archivo;

        textoArchivo = new StringBuilder();

    }

    // Método que cuenta la cantidad de lineas del archivo
    private void cantLineas() throws IOException {

        /*
            Se hace uso de un objeto de la clase BufferedReader para una mayor
            optimización en la velocidad de la lectura del archivo, recibe como
            parámetro en el constructo una instancia de la clase FileReader
            que a su vez recibe este en el constructor el archivo seleccionado
            por el usuario.
         */
        BufferedReader lector = new BufferedReader(new FileReader(file: archivo));

        String linea = lector.readLine();

        /*
            Si la linea de texto es distinta de nulo, entonces la variable que
            lleva el conteo de líneas del archivo aumenta en 1, y el objeto
            textoArchivo concatena la linea en curso y le agrega un salto de
```

```
            línea al final de esta, posteriormente, la variable "linea" toma el
            valor de la línea siguiente del archivo, esto hasta que el método
            lance el valor null, en ese caso se termina el ciclo.

            Esto funciona aún si el archivo está vacío, en ese caso, el valor
            de la primer línea es null, entonces no se realizan las operaciones
            dentro del ciclo.
         */
        while (linea != null) {

            numLineas++;

            textoArchivo.append(str:linea).append(str:"\n");

            linea = lector.readLine();

        }

    }

    private void cantPalabras() {

        /*
            Al momento de realizar el conteo de líneas, se fue almacenando el
            contenido del archivo en el objeto textoArchivo, se evalua si
            el numero de letras en el objeto textoArchivo es distinto de cero,
            entonces quiere decir que el archivo de texto no está vacío y
            comienza a realizar el conteo de palabras.
         */
        if (textoArchivo.length() != 0) {

            /*
               El atributo textArc toma el valor de textoArchivo hasta justo
               antes del ultimo carácter debido a que al momento de realizar el
               conteo de línea, se fue concantenando un salto de línea al final,
               entonces, si el archivo no está vacío, el texto final de texto
               archivo siempre tendrá un salto de línea extra.
             */
            textoArc = textoArchivo.substring(start: 0, textoArchivo.length() - 1);

            int letraFinal = 0;

            /*
                Este ciclo for es para guardar la posición de la última letra
                o numero en textoArchivo, esto para posteriormente utilizar ese
                valor como punto final.
             */
            for (int i = 0; i < textoArchivo.length(); i++) {
```

```
98
99                  if (textoArchivo.charAt(index: i) >= 65 && textoArchivo.charAt(index: i) <= 90
100                         || textoArchivo.charAt(index: i) >= 97 && textoArchivo.charAt(index: i) <= 122
101                         || textoArchivo.charAt(index: i) >= 48 && textoArchivo.charAt(index: i) <= 57) {
102                     letraFinal = i;
103                 }
104
105             }
106
107             int temp = 0;
108
109             /*
110                 Este condicional es para evaluar que textoArchivo no esté vacío,
111                 es decir, si el archivo de texto seleccionado por el usuario,
112                 fue un archivo vacío, entonces no debe realizar el conteo de
113                 palabras y así el resultado del numero de palabras a retornar
114                 posteriormente, será 0.
115              */
116             if (letraFinal != 0 && textoArchivo.charAt(index: letraFinal) != 32
117                     || textoArchivo.charAt(index: letraFinal) != 10) {
118
119                 /*
120                     Dentro de un ciclo while infinito, se realizan otros dos
121                     ciclos for, el primero es para la busqueda de alguna letra
122                     del alfabeto o algún numero, si encuentra una letra o numero
123                     se interpreta como una palabra, entonces el valor de la
124                     variable (int) temporal será la posición en la que inicia
125                     o se interpreta que hay una palabra, si no se encuentra
126                     ninguna letra o número de igual forma el valor de temp
127                     será el valor de i que va en incremento conforme pasa
128                     carácter por carácter, esto para continuar con la lectura
129                     del texto y el ciclo tenga un final.
130                  */
131                 while (true) {
132
133                     for (int i = temp; i <= letraFinal; i++) {
134
135                         if (textoArchivo.charAt(index: i) >= 65 && textoArchivo.charAt(index: i) <= 90
136                                 || textoArchivo.charAt(index: i) >= 97 && textoArchivo.charAt(index: i) <= 122
137                                 || textoArchivo.charAt(index: i) >= 48 && textoArchivo.charAt(index: i) <= 57) {
138
139                             temp = i;
140
141                             break;
142
143                         } else {
144                             temp = i;
145                         }
```

```
146
147                     }
148
149                     /*
150                         Una vez encontrada alguna letra o número, se sale del
151                         ciclo anterior e inmediatamente empieza el segundo ciclo
152                         a buscar un espacio, salto de línea, o la letra o numero
153                         final del texto, esto debido a que la definición de
154                         palabra dice:
155                             1. Unidad léxica constituida por un sonido o
156                                conjunto de sonidos articulados que tienen un
157                                significado fijo y una categoría gramatical.
158                             2. Representación gráfica de estos sonidos, que
159                                consiste en una letra o un grupo de letras
160                                delimitado por espacios blancos.
161                         Basandome en el punto 2, se comprende que una palabra
162                         termina cuando hay un espacio en blanco, o en su defecto
163                         salto de línea o en el ultimo de los casos, que se
164                         inicie la busque de la palabra y esta finalice en la
165                         última letra o numero del texto.
166                      */
167
168                     for (int i = temp; i <= letraFinal; i++) {
169
170                         if (textoArchivo.charAt(index: i) == 32 || textoArchivo.charAt(index: i) == 10
171                                 || i != 0 && i == letraFinal) {
172                             numPalabras++;
173
174                             temp = i;
175
176                             break;
177
178                         }
179
180                     }
181
182                     /*
183                       En caso de que el valor de la variable temporal usada para
184                       indicar la posición en la que se buscan los inicios y
185                       finales de palabras, llegue a la posición de la letra
186                       final, quiere decir que ya no hay más palabras en el texto
187                       por ende, se procede a terminar con el ciclo infinito.
188                      */
189                     if (temp == letraFinal) {
190                         break;
191                     }
192
193                 }
194
```

```
195                 }
196
197             }
198
199         }
200
201         public void cantLineasYPalabras() throws IOException {
202             cantLineas();
203             cantPalabras();
204         }
205
206         public int getNumLineas() {
207             return numLineas;
208         }
209
210         public int getNumPalabras() {
211             return numPalabras;
212         }
213
214         public String getTextoArchivo() {
215             return textoArc;
216         }
217
218     }
219
```

GUI FileView

```
1   package gui;
2
3   import clases.LineasYPalabras;
4   import java.io.File;
5   import java.io.IOException;
6   import javax.swing.JFileChooser;
7   import javax.swing.JOptionPane;
8   import javax.swing.filechooser.FileNameExtensionFilter;
9
10  // @author LuisR
11  public class FileView extends javax.swing.JFrame {
12
13      /** Creates new form FileView ...3 lines */
16      public FileView() {
17          initComponents();
18      }
19
20      /** This method is called from within the constructor to initialize the form ...5 lines */
25      @SuppressWarnings("unchecked")
26      Generated Code
80
81      private void clean() {
82          String vacio = "";
83          lblResultados.setText(text: vacio);
84          taArchivo.setText(t: vacio);
85      }
86
```

```java
    private void btnSelectActionPerformed(java.awt.event.ActionEvent evt) {

        /*
         * Muestra el cuadro de diálogo de archivos, para la selección del
         * archivo de texto que se desee abrir
         */
        JFileChooser selectorArchivos = new JFileChooser();
        selectorArchivos.setFileSelectionMode(mode:JFileChooser.FILES_AND_DIRECTORIES);

        // Filtra los tipos de archivos que se mostrarán en el selector de archivos
        FileNameExtensionFilter filtro = new FileNameExtensionFilter(description:"Archivos de Texto",
                extensions: "java", extensions: "txt", extensions: "csv", extensions: "html",
                extensions: "htm", extensions: "css", extensions: "js", extensions: "xml",
                extensions: "json", extensions: "yaml", extensions: "yml", extensions: "md",
                extensions: "bat", extensions: "cmd", extensions: "srt", extensions: "sql",
                extensions: "ini", extensions: "properties", extensions: "rtf",
                extensions: "log", extensions: "syslog", extensions: "htaccess",
                extensions: "lnk");

        selectorArchivos.setFileFilter(filter: filtro);

         // Indica cual fue la acción del usuario sobre el JFileChooser
         int resultado = selectorArchivos.showOpenDialog(parent: this);

         // Obtiene el archivo seleccionado por el usuario
         File archivo = selectorArchivos.getSelectedFile();

         /*
            Evalua que el archivo seleccionado tenga como extensión alguna de
            las ingresadas anteriormente, si el archivo son de algun tipo de las
            extensiones marcadas, se realiza el conteo de palabras, lineas y
            se muestran los resultados, en caso contrario, se muestra un mensaje
            indicando que sólo se permiten archivos de texto.
          */
         if (filtro.accept(f: archivo)) {

            LineasYPalabras contador = new LineasYPalabras(archivo);

            try {

                contador.cantLineasYPalabras();

                lblResultados.setText("Número de Palabras: " + contador.getNumPalabras()
                        + " | Número de Líneas: " + contador.getNumLineas());

                taArchivo.setText(t: contador.getTextoArchivo());

            } catch (IOException ex) {
                JOptionPane.showMessageDialog(parentComponent:null, message:"Seleccione sólo archivos de texto", title: "Error",
                        messageType:JOptionPane.ERROR_MESSAGE);
                clean();
            }

        } else {
            JOptionPane.showMessageDialog(parentComponent:null, message:"Seleccione sólo archivos de texto",
                    title: "Error", messageType:JOptionPane.ERROR_MESSAGE);
            clean();
        }

    }
```

```
147
148     /**...3 lines */
151     public static void main(String args[]) {
152         /* Set the Nimbus look and feel */
153          Look and feel setting code (optional)
168
169         //</editor-fold>
170
171         /* Create and display the form */
            java.awt.EventQueue.invokeLater(new Runnable() {
                public void run() {
174                 new FileView().setVisible(b: true);
175             }
176         });
177     }
178
179     // Variables declaration - do not modify
180     private javax.swing.JButton btnSelect;
181     private javax.swing.JScrollPane jScrollPanel;
182     private javax.swing.JLabel lblResultados;
183     private javax.swing.JTextArea taArchivo;
184     // End of variables declaration
185 }
186
```