Práctica de Programación. Expresiones Regulares

Integrantes del Equipo:

- Ayuso Contreras Gael Antonio
- Cen Santana Cristopher Israel

Explicación

La ***manera en que implementamos el uso de expresiones regulares para la búsqueda de patrones*** fue que se utilizó la función de **RegEx** (Regular Expression) para revisar lo siguiente:

1. **Primero** si la expresión regular es válida, en el caso que no lo fuera el código no correría.
2. **Posteriormente** realiza la búsqueda de palabras que cumplan con la expresión regular en cada línea del archivo de texto y recorre todas las que lo cumplan en la línea.
3. Al **final** añade todas las coincidencias a un archivo de texto nuevo.

En base a lo anterior, utilizamos principalmente **dos funciones** de nuestra clase `ExpressionValidation`:

`ValidSyntax(String expression)`

- Esta verifica si la expresión regular que ingresamos es válida antes de intentar buscar coincidencias, evitando así cometer errores de compilación en tiempo de ejecución.
- **Cómo se utilizó:**
    - Primero, se pasa la cadena que el usuario ingresó en la interfaz.
    - Si retorna `true`, seguimos con la búsqueda. En caso de ser `false`, muestra un mensaje indicando que la expresión no es válida
    - **Ejemplo de uso:**

```
if (ExpressionValidation.ValidSyntax(expresion)) {
        return true;
    // Continua con la búsqueda
} else {
        return false;
    // Muestra el mensaje de error
}
```

`TestExpression(String expression, String inputFilePath, String outputFilePath)`

- Se encarga de leer el archivo de texto línea por línea, buscar todas las palabras que coincidan con la expresión regular y guardarlas en un archivo de salida.
- **Ejemplo de uso:**

```
 public static void TestExpression(String expression, String inputFilePath, Strin
g outputFilePath) {
        try {
            // Si el archivo no existe, lo crea; si existe, lo sobreescribe.
            BufferedWriter writer = new BufferedWriter(new FileWriter(outputFileP
ath));
            File readingfile = new File(inputFilePath);
            // Lee las lineas del archivo de entrada
            Scanner scanner = new Scanner(readingfile);
            Pattern pattern = Pattern.compile(expression, Pattern.CASE_INSENSITI
VE);

            //Valida que haya una siguiente palabra
            while (scanner.hasNext()) {
                String data = scanner.next();
                Matcher matcher = pattern.matcher(data);
                //Escribe la palabra que se encuentra en una string hasta que ya no
haya
                while (matcher.find()) {
                    writer.write(matcher.group() + "\n");
```

```
            }
        }
```

Código

Main

```
public class Main {
    public static void main(String[] args) {
        //Llama a la interfaz en donde se conecta con el ExpressionValidation
        javax.swing.SwingUtilities.invokeLater(() -> new InterfazRegex());
    }
}
```

Validador de Expresiones

```
import java.io.*;
import java.util.Scanner;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import java.util.regex.PatternSyntaxException;

public class ExpressionValidation {

    //Expresion que valida que la expresion es valida
    public static boolean ValidSyntax(String expression) {
        try {
            //Si la sintaxis es correcta retorna true
            Pattern pattern = Pattern.compile(expression, Pattern.CASE_INSENSITI
VE);
            return true;
        } catch (PatternSyntaxException e) {
```

```
            //System.out.println("La expresion regular no es valida"); *Lo quite por
que en teoria no se ve la terminal
            return false;
        }
    }
    //Funcion que realiza la busqueda de las palabras con la expresion regular
    public static void TestExpression(String expression, String inputFilePath, Str
ing outputFilePath) {
        try {
            // Escribe y crea el archivo
            BufferedWriter writer = new BufferedWriter(new FileWriter(outputFileP
ath));
            File readingfile = new File(inputFilePath);
            // Lee las lineas del archivo de entrada
            Scanner scanner = new Scanner(readingfile);
            Pattern pattern = Pattern.compile(expression, Pattern.CASE_INSENSITI
VE);

            //Valida que haya una siguiente palabra
            while (scanner.hasNext()) {
                String data = scanner.next();
                Matcher matcher = pattern.matcher(data);
                //Escribe la palabra que se encuentra en una string hasta que ya no
haya
                while (matcher.find()) {
                    writer.write(matcher.group() + "\n");
                }
            }

            //Cerrar archivos
            writer.close();
            scanner.close();
        } catch (FileNotFoundException e) {
            //System.out.println("Archivo no encontrado");
        } catch (IOException e) {
            throw new RuntimeException(e);
```

```
        }
    }
}
```

Interfaz

```
import javax.swing.*;
import javax.swing.filechooser.FileNameExtensionFilter;
import java.awt.*;
import java.awt.event.*;
import java.io.File;

class InterfazRegex extends JFrame {
    //Aqui se declarará la expresión a buscar
    private final JTextField zonaDeExpresion;
    //Etiqueta que indicara el archivo seleccionado
    private final JLabel archivo;
    private File archivoSeleccionado;
    //Boton de busqueda de expresiones y de guardar resultados
    private final JButton buscadorDeExpresiones;
    //Etiqueta que indica el estado del resultado (donde se guardo o si se guard
o)
    private final JLabel resultado;

    public InterfazRegex() {

        setTitle("Validador de Expresiones Regulares");
        setSize(400, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setLayout(new BorderLayout());

        // Panel Gneral Multiuso
        JPanel panelGeneral = new JPanel();
        panelGeneral.setLayout(new BoxLayout(panelGeneral, BoxLayout.Y_AXI
```

```
S));
        panelGeneral.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 2
0));

        archivo = new JLabel("Buscar documento");
        archivo.setFont(new Font("Arial", Font.BOLD, 15));
        panelGeneral.add(archivo);
        panelGeneral.add(Box.createRigidArea(new Dimension(0, 10)));

        JButton buscadorDeArchivosBoton = new JButton("Seleccionar archivo .t
xt");
        buscadorDeArchivosBoton.setFocusPainted(false);
        buscadorDeArchivosBoton.addActionListener(e -> seleccionarArchivo());
        panelGeneral.add(buscadorDeArchivosBoton);
        panelGeneral.add(Box.createRigidArea(new Dimension(10, 15)));

        JLabel textoInstruccion = new JLabel("Ingresa la expresión a buscar en e
l archivo seleccionado");
        panelGeneral.add(textoInstruccion);
        panelGeneral.add(Box.createRigidArea(new Dimension(0, 10)));

        //Especificaciones de la zona de la expresion regular, para que no se pier
da el usuario
        zonaDeExpresion = new JTextField("El texto va aqui...");
        zonaDeExpresion.setForeground(Color.GRAY);
        zonaDeExpresion.addFocusListener(new FocusAdapter() {
            public void focusGained(FocusEvent e) {
                if (zonaDeExpresion.getText().equals("El texto va aqui...")) {
                    zonaDeExpresion.setText("");
                    zonaDeExpresion.setForeground(Color.BLACK);
                }
            }
            public void focusLost(FocusEvent e) {
                if (zonaDeExpresion.getText().isEmpty()) {
                    zonaDeExpresion.setText("El texto va aquí...");
                    zonaDeExpresion.setForeground(Color.GRAY);
```

```
            }
        }
    });
    panelGeneral.add(zonaDeExpresion);

    panelGeneral.add(Box.createRigidArea(new Dimension(0, 15)));

    buscadorDeExpresiones = new JButton("Buscar expresión");
    buscadorDeExpresiones.setFocusPainted(false);
    buscadorDeExpresiones.setBackground(new Color(234, 133, 62));
    buscadorDeExpresiones.setForeground(Color.WHITE);
    buscadorDeExpresiones.setEnabled(false);
    buscadorDeExpresiones.addActionListener(e -> runExpressionValidation
());
    panelGeneral.add(buscadorDeExpresiones);

    panelGeneral.add(Box.createRigidArea(new Dimension(0, 15)));

    resultado = new JLabel("");
    resultado.setFont(new Font("Arial", Font.BOLD, 14));
    resultado.setForeground(new Color(234, 133, 62));
    panelGeneral.add(resultado);
    add(panelGeneral, BorderLayout.CENTER);
    setVisible(true);
  }

  private void seleccionarArchivo() {
    JFileChooser seleccionadorDeArchivo = new JFileChooser();
    seleccionadorDeArchivo.setFileFilter(new FileNameExtensionFilter("Archi
vos .txt", "txt"));
    int option = seleccionadorDeArchivo.showOpenDialog(this);

    if (option == JFileChooser.APPROVE_OPTION) {
      archivoSeleccionado = seleccionadorDeArchivo.getSelectedFile();
      archivo.setText("Archivo seleccionado: " + archivoSeleccionado.getNa
me());
```

```
            buscadorDeExpresiones.setEnabled(true);
        }
    }
    //Se ejecuta el validador de expresiones y se guarda el archivo
    private void runExpressionValidation() {
        String expresion = zonaDeExpresion.getText();

        if (ExpressionValidation.ValidSyntax(expresion)) {
            JFileChooser guardadorDeArchivos = new JFileChooser();
            guardadorDeArchivos.setDialogTitle("Guardar resultados de RegEx");
            guardadorDeArchivos.setFileFilter(new FileNameExtensionFilter("Archi
vos .txt", "txt"));

            int option = guardadorDeArchivos.showSaveDialog(this);

            if (option == JFileChooser.APPROVE_OPTION) {
                File fileToSave = guardadorDeArchivos.getSelectedFile();
                // Aseguramos que termine en .txt
                String outputPath = fileToSave.getAbsolutePath();
                if (!outputPath.toLowerCase().endsWith(".txt")) {
                    outputPath += ".txt";
                }
                // Llamada al validador y guardamos el path
                ExpressionValidation.TestExpression(
                        expresion,
                        archivoSeleccionado.getAbsolutePath(),
                        outputPath
                );

                resultado.setText("Los resultados se guardaron en: " + outputPath);
            } else {
                resultado.setText("No se guardó ningún archivo.");
            }
        } else {
            resultado.setText("La expresión regular no es válida");
        }
```

```
    }
}
```

Ejemplos de Ejecución con Resultado