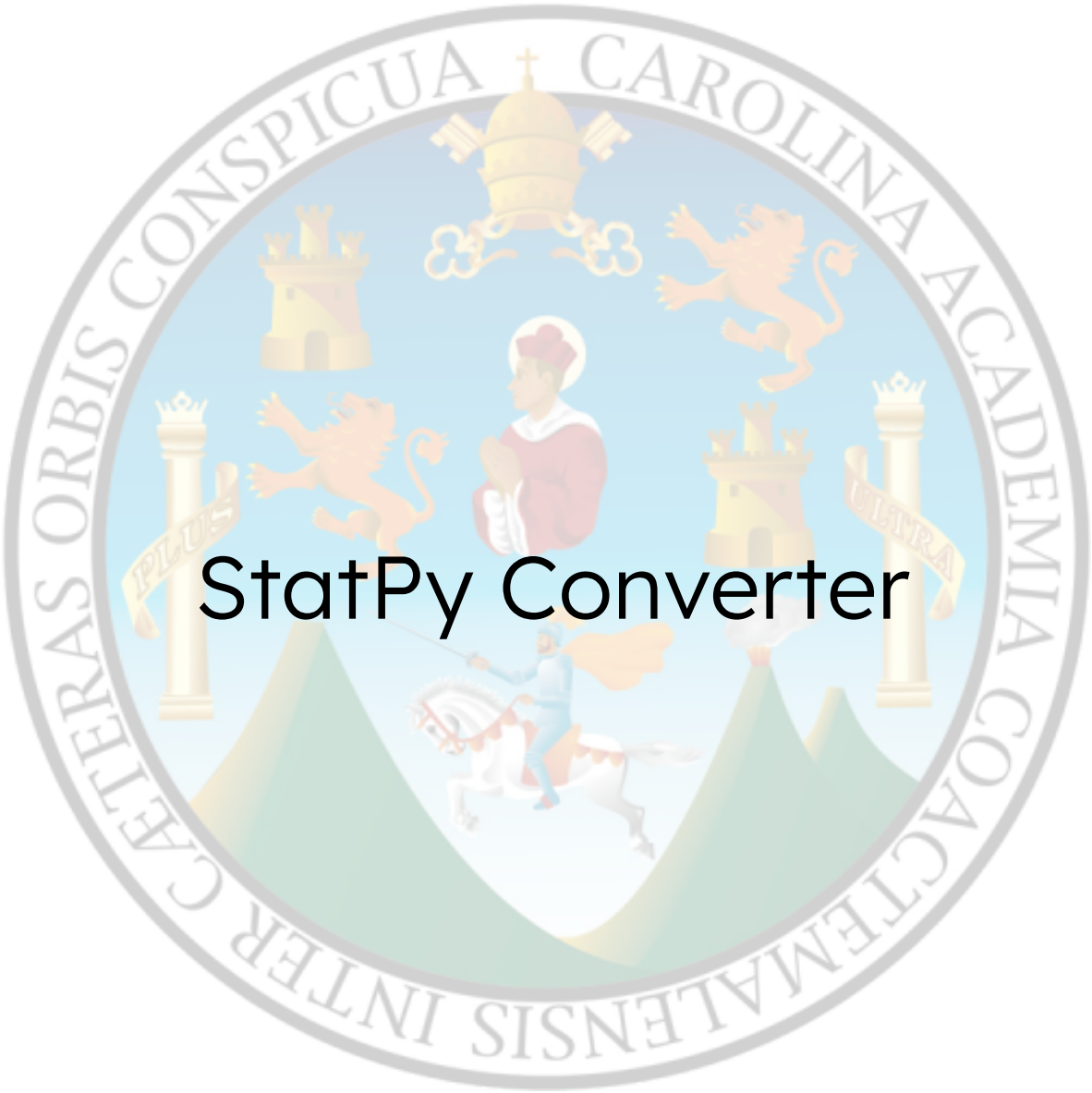
MANUAL
TECNICO



StatPy Converter

Tabla de Contenido

Introducción

El presente documento describe la serie de pasos que realizan algunos de los métodos o funciones que conforman el programa computacional (StatPy Convertor) por medio de fragmentos de código, como también se detalla los conocimientos previos que debe tener el lector de este manual para comprender de una mejor manera el funcionamiento de cada una de las partes del código que conforman el programa.

Objetivos

Instruir el uso adecuado del programa computacional, describiendo el diseño y la lógica del programa por medio de fragmentos de códigos.

Describir al usuario el funcionamiento del programa para el mejor uso de él y demostrar el proceso necesario para su ejecución.

Orientar al usuario a entender la estructura del programa, como lo son sus clases y cada uno de los métodos que componen dicha clase.

Información Destacada

El manual técnico hace referencia a la información necesaria con el fin de orientar al personal en la concepción, planteamiento análisis programación e instalación del sistema. Es de notar que la redacción propia del manual técnico está orientada a personal con conocimientos en sistemas y tecnologías de información, conocimientos de programación avanzada sobre Java, responsables del mantenimiento e instalación del programa computacional en el computador.

Conocimientos Previos

Los conocimientos mínimos que deben tener las personas que operarán las páginas y deberán utilizar este manual son:

- Conocimientos y entendimientos en logaritmos
- Conocimientos en Java
- Conocimientos en JFlex y Cup
- Programación Orientada a Objetos
- Interfaces gráficas
- Conocimiento básico de Windows

Requerimientos

El sistema puede ser instalado en cualquier sistema operativo que cumpla con los siguientes requerimientos:

- Sistema Operativo: Windows 7 o superior
- Procesador mínimo Intel Pentium (800 MHz Intel Pentium)
- Mínimo 1GB en RAM
- IDE Apache Netbeans, o compatible con Java
- Exploradores: Internet Explorer 9 y superior

Instalación y Configuración

Para el proceso de instalación de esta aplicación únicamente es necesario tener instalado un IDE que sea compatible con el lenguaje Java para ejecutar la aplicación desde la terminal de este.

No es necesario tener alguna configuración ya que la configuración que trae por determinado el IDE es la necesaria para que el funcionamiento del programa sea posible.

Fragmentos de Código

En esta sección, se proporciona una explicación detallada de los métodos y funciones clave que componen el código del programa. El propósito principal es facilitar la comprensión de cómo funciona el programa, para que las personas que lo utilicen puedan brindar un mejor soporte y tener una comprensión más clara de la lógica subyacente en el software. En otras palabras, se busca documentar y explicar de manera exhaustiva las partes esenciales del código para hacer que su mantenimiento y uso sean más accesibles y comprensibles.

1. Código del método “abrir”:

```
private void btnOpenActionPerformed(java.awt.event.ActionEvent evt) {
    JFileChooser file_chooser = new JFileChooser();
    FileNameExtensionFilter filter = new FileNameExtensionFilter( description: "Valid Files",  extensions: "sp",  extensions: "json");
    file_chooser.setFileFilter(filter);
    int result = file_chooser.showOpenDialog( parent: this);

    if (result == JFileChooser.APPROVE_OPTION) {
        File fileSelected = file_chooser.getSelectedFile();
        String content = "";
        try {
            BufferedReader reader = new BufferedReader(new FileReader( file: fileSelected));
            String line;
            while ((line = reader.readLine()) != null) {
                content += line + "\n";
            }
            reader.close();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
        this.areaCode.setText( t: content);
        path_file = String.valueOf( obj: fileSelected);
        this.lexErrorsStatPy = null;
        this.sintaxErrorsStatPy = null;
    }

}
```

Este método inicia su proceso verificando si el usuario ha elegido un archivo. En caso afirmativo, guarda la ubicación del archivo en la variable llamada "path_file". Si el usuario no ha seleccionado ningún archivo, la función no realiza ninguna acción adicional.

Una vez que se ha almacenado la ruta del archivo, se procede a comprobar si la extensión del archivo seleccionado es la correcta, es decir, si es ".sp" o ".json". Si la extensión no es la adecuada, se mostrará un mensaje de error en la pantalla para informar al usuario del problema.

Si la extensión del archivo es correcta, el método continúa leyendo el contenido del archivo línea por línea y almacena toda esta información en la variable llamada

"content". Posteriormente, se mostrará el contenido completo en el área de texto de la aplicación para que el usuario pueda visualizarlo.

2. Código del método “guardar”:

```
private void btnSaveActionPerformed(java.awt.event.ActionEvent evt) {
    String content = this.areaCode.getText();
    if (path_file != "") {
        try {
            FileWriter writer = new FileWriter( fileName: path_file,  append: false);
            writer.write( str: content);
            writer.close();
            JOptionPane.showMessageDialog( parentComponent: this,  message: "¡Archivo guardado exitosamente!",  title: "Informacion",  messageType:
        } catch (IOException e) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "¡Error al guardar el archivo!",  title: "Error",  messageType: JOptionP
        }
    } else {
        JFileChooser file_chooser = new JFileChooser();
        int result = file_chooser.showSaveDialog( parent: this);

        if (result == JFileChooser.APPROVE_OPTION) {
            File file = file_chooser.getSelectedFile();

            try {
                String pathFile = String.valueOf( obj: file);
                FileWriter writer = new FileWriter( fileName: pathFile);
                writer.write( str: content);
                writer.close();
                JOptionPane.showMessageDialog( parentComponent: this,  message: "¡Archivo creado exitosamente!",  title: "Informacion",  messageTyp
            } catch (IOException e) {
                JOptionPane.showMessageDialog( parentComponent: this,  message: "¡Error al crear el archivo!",  title: "Error",  messageType: JOptio
            }
            path_file = String.valueOf( obj: file);
        }
    }
}
```

Este método es llamado cuando el usuario presiona el botón “guardar” que se muestra en la interfaz gráfica de la aplicación y funciona de la siguiente manera:

Guarda todo el contenido que tiene el área de texto de la aplicación en una variable, luego verifica si el contenido pertenece a un archivo abierto por medio de la aplicación y de ser así realiza los cambios que el usuario ingresó en el archivo. En caso de que no haya un archivo abierto el programa le preguntará al usuario si desea guardar el contenido en un archivo nuevo.

3. Codigo del metodo “analizador”

```
private void btnStatPyActionPerformed(java.awt.event.ActionEvent evt) {
    this.currentAnalyzer = "StatPy";
    this.lblCurrentAnalyzer.setText( text: this.currentAnalyzer);
}
```

El método "analizador" tiene la responsabilidad de actualizar el tipo de análisis que se aplicará al texto ingresado en el área de texto, el cual proviene de un archivo con extensión ".sp" o ".json". En otras palabras, este método cambia el enfoque de análisis que se utilizará en función del tipo de archivo que se haya leído previamente.

4. Código del método “ejecutar”

```
AnalyzerResult result = Utils.analyzerFileStatPy( input: content, variables_json);
this.ast = result.ast;
this.lexErrorsStatPy = result.lexErrors;
this.sintaxErrorsStatPy = result.sintaxErrors;
this.tokensStatPy = result.tokens;

if (!lexErrorsStatPy.isEmpty() || !sintaxErrorsStatPy.isEmpty()) {
    JOptionPane.showMessageDialog( parentComponent: this,  message: "Se han encontrado errores en la entrada",  title: "Error",  messageType: JOptionP

    ErrorsReport lexicalErrorsReport = new ErrorsReport();
    String content_html_errors = lexicalErrorsReport.generateErrorsReport( lexErrors: lexErrorsStatPy);

    try {
        FileWriter writer = new FileWriter( fileName: "./src/main/java/docs/erroresStatPy.html");
        writer.write( str: content_html_errors);
        writer.close();
    } catch (IOException e) {
        JOptionPane.showMessageDialog( parentComponent: this,  message: "¡Error al crear el archivo de reporte!",  title: "Error",  messageType: JOpti
    }

} else {
    this.areaResult.insert( str: Utils.translatePython(ast),  pos: 0);
    JOptionPane.showMessageDialog( parentComponent: this,  message: "¡Analisis realizado exitosamente!",  title: "Informacion",  messageType: JOptionP

}
```

Este método se encarga de examinar el texto ingresado en el área de texto. En primer lugar, verifica si hay errores en el código. Si encuentra errores, se mostrará un mensaje de error en la pantalla para informar al usuario sobre la presencia de problemas.

Si no se detectan errores, el método procede a traducir el código ingresado a lenguaje Python. Además, si el código contiene funciones, este método generará gráficos, ya sea de barras o de pie, con los valores correspondientes. En resumen, se encarga de validar y procesar el código, generando gráficos en caso de que se requieran.

5. Código del método “errores”:

```
private void btnErrorsJSONActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        File fileHTML = new File( pathname: "src/main/java/docs/erroresJSON.html");

        if (fileHTML.exists()) {
            Desktop desktop = Desktop.getDesktop();
            if (desktop.isSupported( action: Desktop.Action.BROWSE)) {
                desktop.browse( uri: fileHTML.toURI());
            } else {
                JOptionPane.showMessageDialog( parentComponent: this,  message: "La funcionalidad de navegación no es compatible.",  title: "Adver
            }
        } else {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "¡Error el archivo no existe!",  title: "Error",  messageType: JOptionPane
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Este método tiene la función de presentar los errores identificados durante el análisis léxico del archivo, ya sea en formato ".sp" o ".json", en un documento PDF. En otras palabras, se encarga de crear un archivo PDF que contiene una lista detallada de los errores que se encontraron durante el proceso de análisis léxico del archivo. Esto permite al usuario tener una referencia clara de los problemas encontrados en el código.