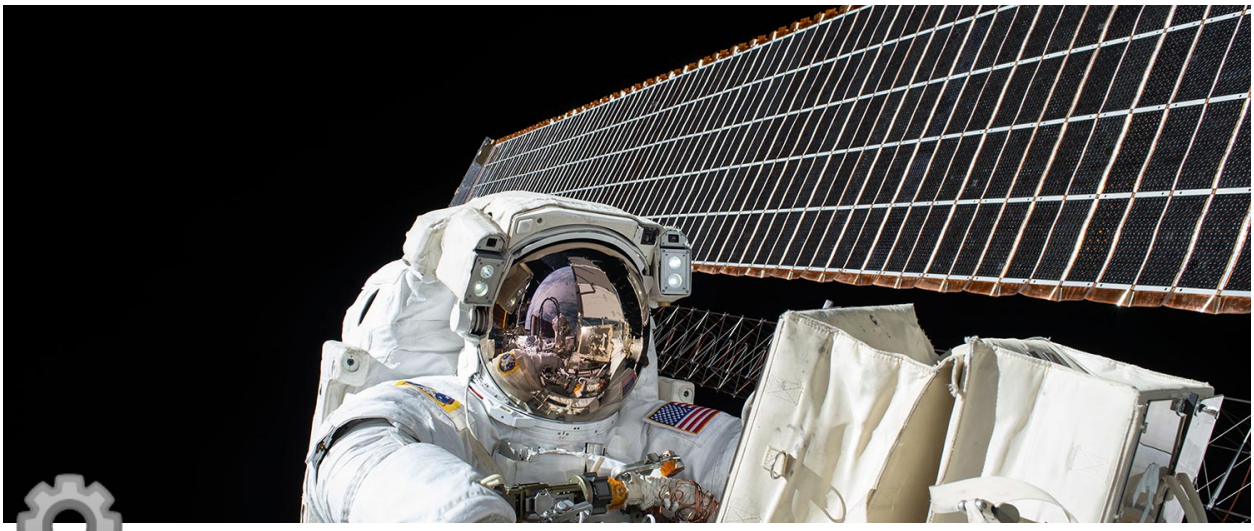

# Manual Técnico

Copy Analyzer es un programa desarrollado en Java la cual despliega una interfaz grafica de java swing. En la carpeta principal proyecto se encuentran todos los paquetes utilizados a lo largo del proyecto, así como las librerías que fueron utilizadas. Se utilizó la versión 8 de java y como IDE se utilizó NetBeans 11.2. Copy Analyzer es un proyecto desarrollado en java ant. La clase principal es la clase llamada javaant en el paquete principal, la cual tiene el método main.

## Librerías

- JFlex

  Se utilizó la librería JFlex para llevar a cabo los analizadores léxicos del programa. Su función principal es definir los tokens que los distintos lenguajes pueden recibir y almacenarlos según se localizan en la entrada. JFlex utiliza expresiones regulares para definir cada token y todos los archivos con extensión jflex contienen todos los tokens definidos para los lenguajes que juegan un papel importante en el programa. También es útil para determinar la linea y la columna en la que se encuentran los tokens.  Si la entrada contiene símbolos desconocidos que no pertenecen al lenguaje, JFlex se encarga de manejarlos como puede y luego informar al usuario de todos los errores léxicos encontrados.

- JCup

  Se utilizó la librería JCup para llevar a cabo los analizadores sintácticos del programa. Su función principal es definir la sintaxis que los distintos lenguajes pueden aceptar por medio de símbolos terminales y símbolos no terminales. Todos los archivos .cup en el proyecto contienen las gramáticas que los lenguajes manejan junto con la declaración de todas las variables terminales y no terminales. JCup también es útil para guardar información importante que llega por las entradas y sirve para ejecutar más código en procesos posteriores. Si la entrada contienen errores de sintaxis, JCup se encarga de manejarlos como puede y luego informar al usuario de todos los errores sintácticos encontrados.

- JFreeChart

  Se utilizó JFreeChart para generar las gráficas del reporte estadístico. En general, solo hay 3 tipos de gráficas de JFreeChart que se utilizan en todo el proyecto: gráfica de barras, gráfica pie, gráfica de líneas. Cada una de estas gráficas contienen información que se especifica en los archivos con extensión fca de reportería los cuales son analizados por JFlex y JCup, por lo que las gráficas dependen de la información recuadada por los analizadores léxicos y sintácticos.

# Paquetes

- Javaant

  En el paquete Javaant se encuentra la clase Javaant la cual contiene la clase main en donde inicia el código de todo el proyecto. La única instrucción en el método main es mostrar el jframe menu.

  La clase LectorProyectos también se encuentra en Javaant. LectorProyectos se encarga de analizar los archivos de los dos proyectos declarados en el archivo fca de reportería. Al final, LectorProyectos devuelve dos arreglos con las rutas de todos los archivos con nombres repetidos de los dos proyectos enteros, para que se manden al analizador en pares de rutas.

  La clase Comparador también se encuentra en el paquete Javaant. Comparador es el motor del proyecto para comparar dos archivos, ya que en comparador se analiza las repeticiones de clases, métodos, variables y comentarios. También se

encarga de guardar toda la información de cada archivo para luego utilizarla en los reportes.

La clase Total también se encuentra en Javaant. Total se encarga de obtener el puntaje general de toda la comparación. En su constructor recibe como parámetros todas las comparaciones de los archivos repetidos.

La clase VerVariables también se encuentra en Javaant. VerVariables se encarga de guardar todas las variables encontradas en los archivos fca para utilizarlas en el momento en que se desee generar una gráfica.

La clase VerGraficas también se encuentra en Javaant. VerGraficas se encarga de generar las gráficas declaradas en el archivo fca. Para ello, los analizadores léxico y sintáctico tuvieron que recolectar toda la información necesaria para generar las gráficas.

▼ Reportería

La clase LexerReporteria se encuentra en el paquete Reportería. Esta es la clase que genera JFlex al ejecutar el archivo ReporteriaLexer.jflex y se encarga de hacer el análisis léxico del lenguaje de reportería.

El archivo ReporteriaLexer.jflex también se encuentra en Reportería. Este archivo se encarga de declarar todos los tokens que el analizador léxico puede aceptar como entrada.

La clase ParserReporteria también se encuentra en Reportería. Esta es la clase que genera JCup al ejecutar el archivo ReporteriaParser.cup y se encarga de hacer el análisis sintáctico del lenguaje de reportería.

El archivo ReporteriaParser.cup también se encuentra en Reportería. En este archivo se encuentra toda la gramática utilizada en el analizador sintáctico.

▼ LenguajeCopias

La clase LexerCopias se encuentra en el paquete LenguajeCopias. Esta es la clase que genera JFlex al ejecutar el archivo CopiasLexer.jflex y se encarga de hacer el análisis léxico del lenguaje JavaScript.

El archivo CopiasLexer.jflex también se encuentra en LenguajeCopias. Este archivo se encarga de declarar todos los tokens que el analizador léxico puede aceptar como entrada.

La clase ParserCopias también se encuentra en LenguajeCopias. Esta es la clase que genera JCup al ejecutar el archivo CopiasParser.cup y se encarga de hacer el análisis sintáctico del lenguaje JavaScript.

El archivo CopiasParser.cup también se encuentra en LenguajeCopias. En este archivo se encuentra toda la gramática utilizada en el analizador sintáctico.

▼ Models

El paquete Models contiene todas las clases de modelos bastante útiles a lo largo del desarrollo del proyecto. Cada una de estas clases contienen propiedades que le dan sentido a la clase. Los modelos declarados en este son:

- Archivo.java
- Clase.java
- Error.java
- Grafica.java
- Metodo.java
- Puntaje.java
- PuntajeEspecifico.java
- PuntajeGeneral.java
- SetArchivo.java
- Token.java
- Variable

▼ Utils

En el paquete Utils se encuentran los java enum utilizados para los nombre de los tokens de cada analizador. En este caso son dos analizadores léxicos, por lo que los archivos son:

- TokensReporteria.java
- TokensJS.java

▼ Vistas

En este paquete se encuentra toda la parte visual del programa. Para manejar toda la interfaz gráfica del programa se utilizó java swing. El único archivo en este paquete es el archivo menu.java el cual es el que se manda a llamar en el método main del proyecto. En este archivo se encuentra la clase Menu y aquí es donde se declara toda la interfaz del proyecto. Menu maneja muchos paneles, según la cantidad de pestañas generadas por el usuario y todas las características de de los analizadores dependen de la pestaña activa. Cada pestaña tiene su propio JTextArea en el cual se escribe todo el lenguaje de reportería. El botón que inicia el proceso de análisis es el de "ejecutar" el cual ejecuta el código de reportería del panel activo y la consola (otro JTextArea) se va actualizando con mensajes dependiendo de lo que se va ejecutando. También se encuentran los botones para generar reportes, los cuales hacen un nuevo objeto reporte con la información de la pestaña activa. Cabe mencionar que los botones de reportes no funcionaran si no se ejecuta el codigo de reportes primero.

▼ Reportes

La clase ReporteErrores se encuentra en el paquete Reportes. Esta clase se encarga de generar el reporte de errores en un archivo html con un listado de todos los errores léxicos y sintácticos encontrados por los analizadores de los dos lenguajes del proyecto.

La clase ReporteEstadistico también se encuentra en Reportes. Esta clase se encarga de generar el reporte estadístico del panel activo en un archivo html con todas las clases, métodos, variables y comentarios encontrados en ambos proyectos, además de las gráficas declaradas en el archivo fca.

La clase ReporteTokens también se encuentra en Reportes. Esta clase se encarga de generar un reporte de tokens del panel activo en un archivo html con todos los tokens encontrados por los analizadores léxicos del proyecto.

La clase ReporteJson también se encuentra en Reportes. Esta clase se encarga de generar un reporte en formato json con el puntaje general y todos los puntajes específicos detectados por los comparadores.