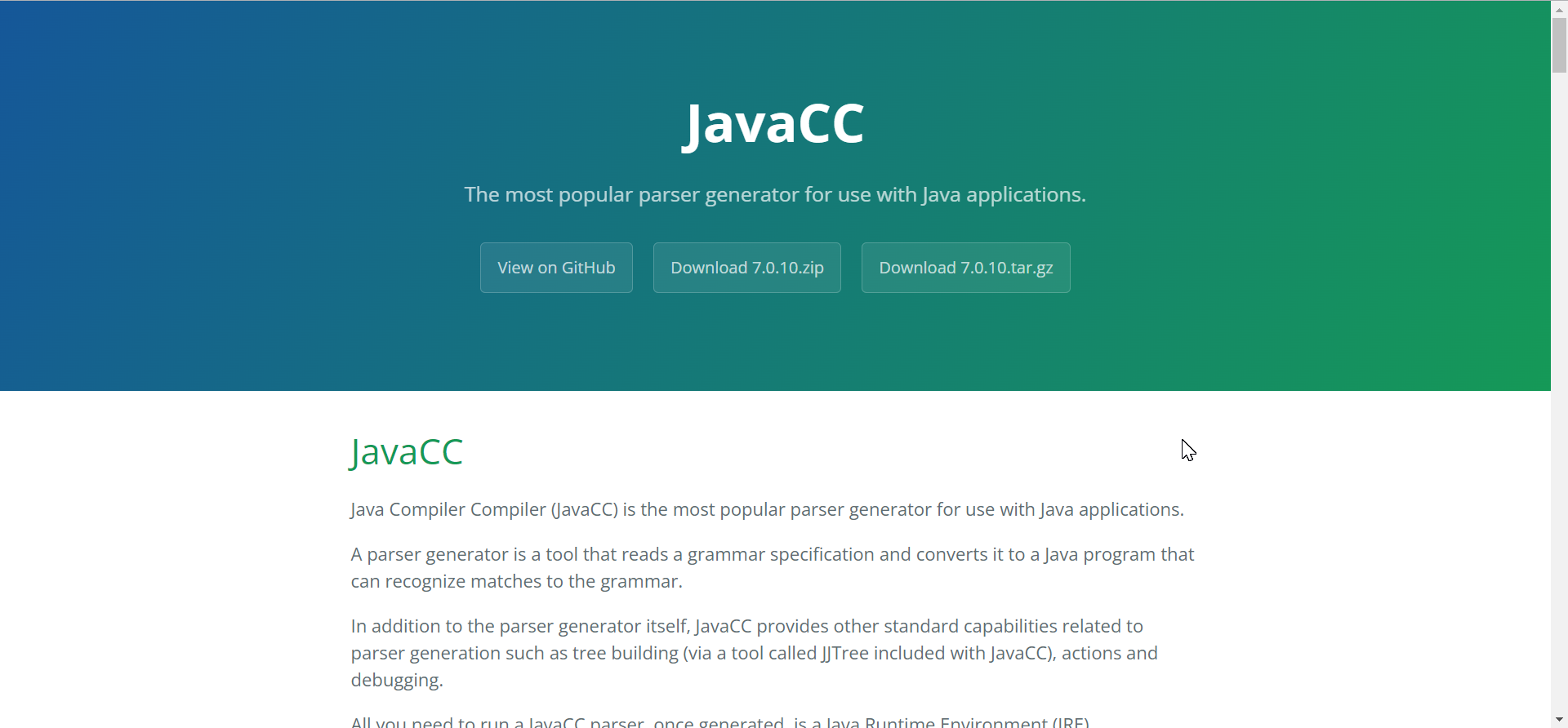
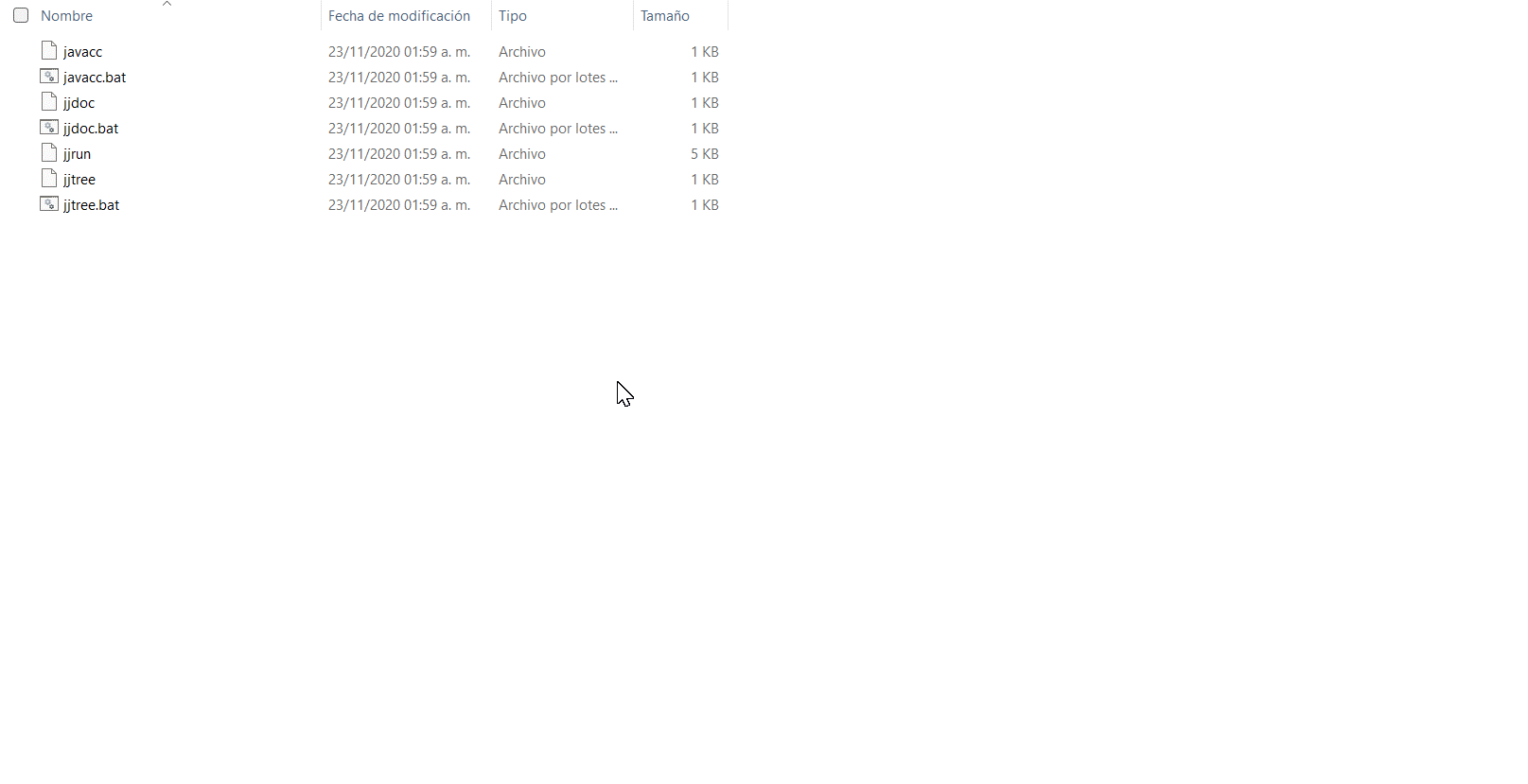
# Requerimientos (Instalación de JavaCC).



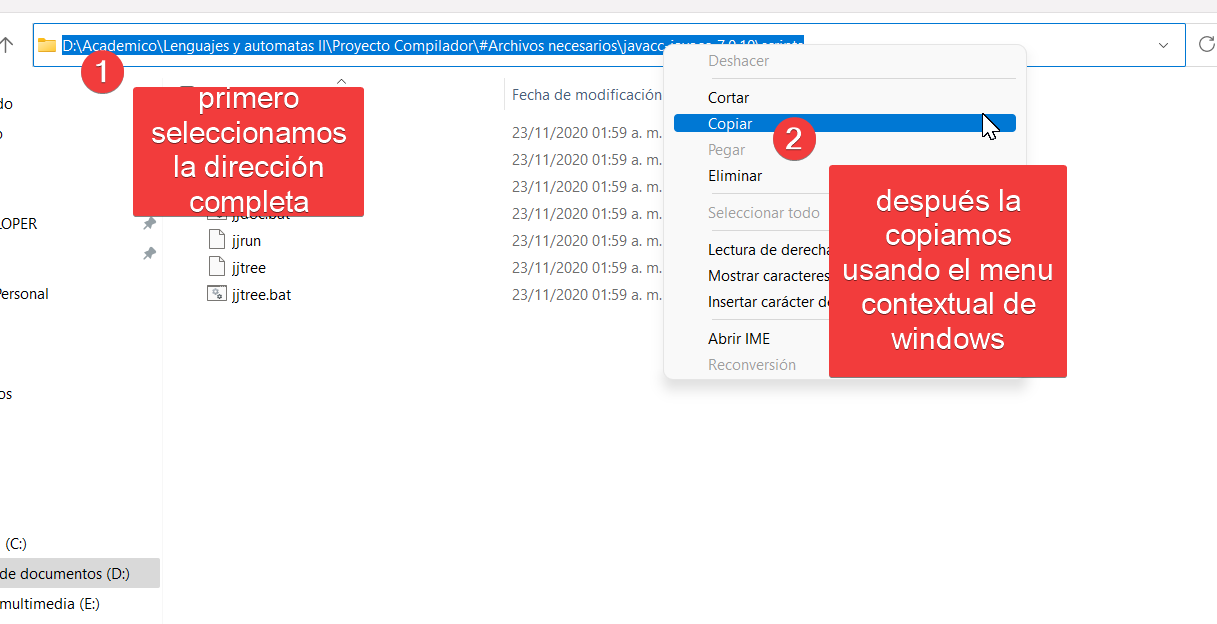
Captura de pantalla . - Página oficial de JavaCC

Para comenzar ésta práctica necesitamos descargar JavaCC. Podemos descargar JavaCC desde la página oficial incluida en los anexos de éste documento. Tendremos a la vista tres botones, debemos seleccionar la segunda opción, la que nos permite descargar el archivo en formato zip. Una vez hecho eso se descargará un archivo comprimido que debemos descomprimir en cualquier dirección de nuestra computadora, en éste caso yo lo descomprimí en el directorio maestro del proyecto compilador para la materia lenguajes y automatas II. En la versión 7 de JavaCC tendremos que ubicar la carpeta “scripts” que contenga los archivos como se aprecian en la imagen.



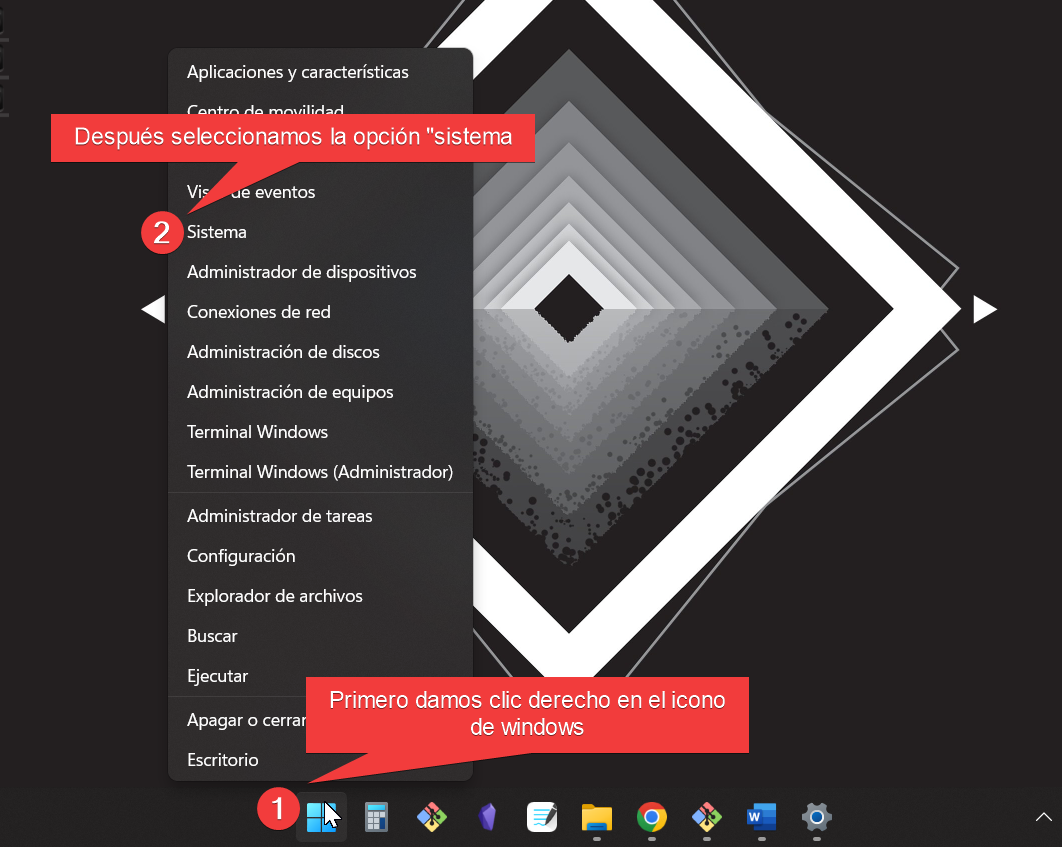
Captura de pantalla . - Carpeta para añadir a las variables de entorno

Una vez ubicada esa carpeta entonces podemos añadirla a las variables de entorno de nuestro sistema operativo. Es importante no mover la carpeta de la dirección que hayamos seleccionado para evitar errores.



Captura de pantalla . - Primer paso para añadir variables de entorno

Tenemos que usar la dirección completa donde se encuentran los ejecutables para nuestros programas, en éste caso es la carpeta Scripts dentro de la carpeta JavaCC que descargamos. Lo último sería añadirlo a las variables de entorno, esto es relativamente fácil.



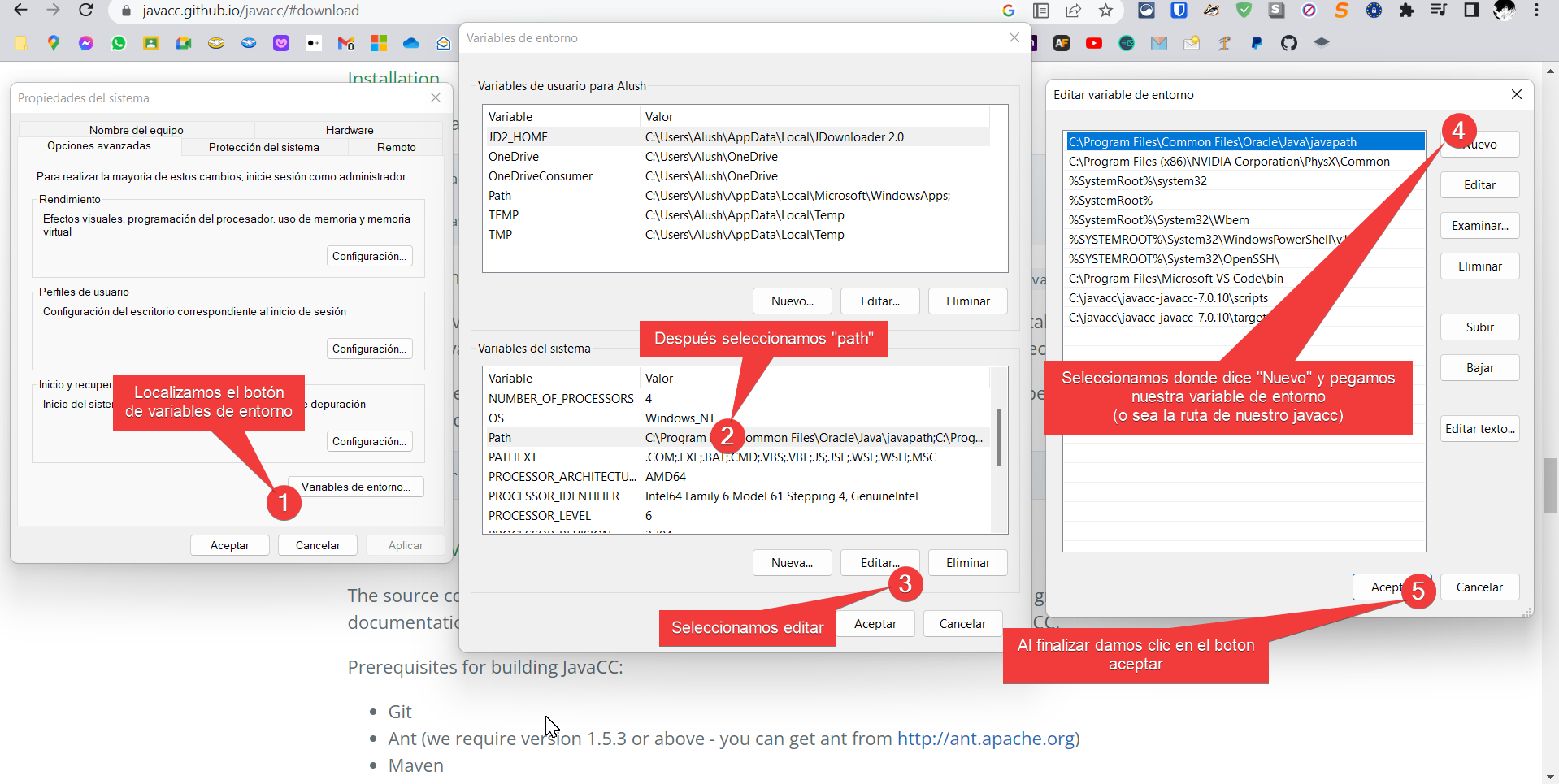
Captura de pantalla . - Segunda serie de pasos para añadir variables de entorno

Seguimos las instrucciones de la “Captura de pantalla I.D - Segunda serie de pasos para añadir variables de entorno” para poder llegar al sistema.



Captura de pantalla . Tercer paso para añadir variables de entorno

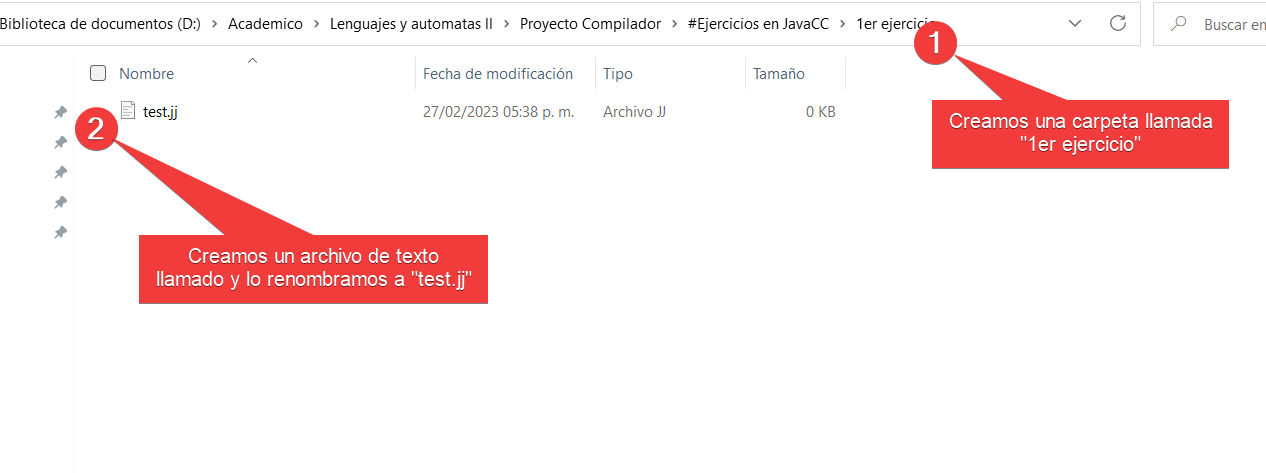
Una vez llegados al sistema seleccionamos la opción que se aprecia en la imagen. Y seguimos los pasos que se muestran en la siguiente captura de pantalla.



Captura de pantalla . Cuarta serie de pasos para añadir variables de entorno

# 1er ejercicio en JavaCC

Éste ejercicio corresponde a “tutorial 1: Sintaxis en Java”. El enlace se puede encontrar en la sección de anexos de éste documento.



Captura de pantalla . - Creación del archivo del ejercicio 1

## Código del ejercicio 1:

Editamos con cualquier editor de texto nuestro archivo “test.jj” de Captura de pantalla II.A - Creación del archivo del ejercicio 1

PARSE\_BEGIN (test)

class test

public static void main (String [] args) throws ParseException

{

try

{

test analizador== new test (System.in);

analizador.Programa ();

System.out.println ("Se ha compilado con exito");

}

catch (ParseException e)

{

System.out.println (e.getMessage());

System.out.println ("Se han encontrado errores");

}

}

PARSE\_END (test)

TOKEN:

{

<MAIN: "public static void Main ()"> {System.out.println ("MAIN" ->"+image);}

<PROGRAMA: "Programa" > {System.out.println ("PROGRAMA ->"+image);}

<IF "ien" > {System.out.println ("IF ->"+image);}

}

TOKEN:

{

<LPAREN: "(" > {System.out.println("LPAREN ->"+image);}

|<RPAREN: ")" > {System.out.println("DPAREN ->"+image);}

|<LBRACE: "{" > {System.out.println("ILLAVE ->"+image);}

|<RBRACE: "}" > {System.out.println("DLLAVE ->"+image);}

|<SEMICOLON: ";" > {System.out.println("PUNTO Y COMA ->"+image);}

}

TOKEN:

{

<ASIGNACION: "=" > {System.out.print.ln("ASIGNACION ->"+image+"\r\n");}

<MENOR: "<" > {System.out.print.ln("MENOR QUE ->"+image);}

}

TOKEN:

{

<INT: "inum" > {System.out.print.ln("ASIGNACION ->"+image+"\r\n");}

<NUMBER: (["0"-"9"])+ > {System.out.print.ln("NUMERO ->"+image+"\r\n");}

<IDENTIFIER: ["a"-"z","A"-"Z"] (["a"-"z","A"-"Z","0"-"9","\_"])\*> {System.out.print.ln("ASIGNACION ->"+image+"\r\n");}

}

SKIP:

{

" " | "\r\n" | "\t"

}

## Notas importantes

* Los archivos con formato .jj son los que podrán ser leídos por JavaCC
* El nombre de la clase debe ser igual al nombre del archivo (esto nos permitirá compilar).
* Los tokens son bloques de código, es recomendable asignar instrucciones del mismo tipo para cada bloque de código, esto por hacer buenas prácticas.
* SKIP debe ir al final, se encarga de saltar los espacios en blanco, tabulaciones y retornos. La razón de que vaya al final es evitar errores de código ya que si se ejecuta un SKIP antes podría afectar las entradas de nuestro programa.

# 2do ejercicio en JavaCC

Inicio del ejercicio

# 3er ejercicio en JavaCC