**Trabajo Practico de Traductor Dirigido por la Sintaxis**

**CI.:** 5.574.180

**Nombre:** Andres

**Apellido:** Heo Kim

**Tema:** Dado una lista de precios, se pide obtener el valor correspondiente al precio promedio y la cantidad de precios que se tomaron en consideración (elementos en la lista).

Inicialmente planteamos que la lista de números está separado por comas ( , ) y que los precios son números enteros.

Ejemplo de entrada de la lista de precios valida:

10000 , 40000 , 5000

Serian 3 precios con un promedio de 18333

Empezamos el análisis de como vamos a leer la estructura de lista de precios con su separador la coma ( , ).

Tenemos un BNF inicial para leer lista de precios:

LISTA -> PRECIO , LISTA

LISTA -> PRECIO

y el BNF de precio seria un numero entero, basicamente una cadena consecutiva de digitos:

PRECIO -> DIGITO PRECIO

PRECIO -> DIGITO

DIGITO -> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Escribimos todo el BNF en una columna:

TDS -> LISTA

LISTA -> PRECIO **,** LISTA | PRECIO

PRECIO -> DIGITO PRECIO | DIGITO

DIGITO -> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Verificamos que no tenga recursión por la izq., en este caso no tenemos.

Verificamos factores comunes: en este caso tenemos en “lista” y en “precio”

LISTA -> PRECIO **,** LISTA | PRECIO LISTA -> PRECIO R1

R1 -> **,** LISTA | ε

PRECIO -> DIGITO PRECIO | DIGITO PRECIO -> DIGITO R2

R2 -> PRECIO | ε

Verificamos el conjunto primero:

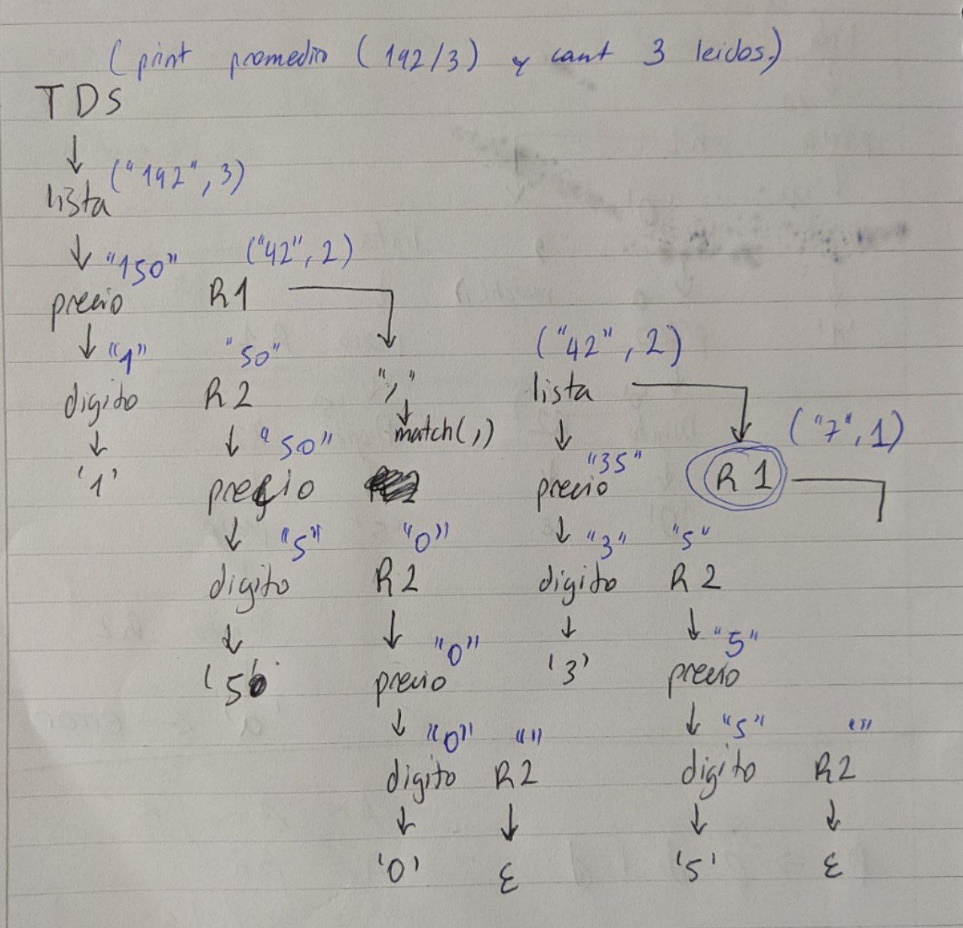
|  |  |
| --- | --- |
| **BNF** | **Conjunto Primero** |
| TDS -> LISTA | P(TDS) = P(LISTA) = { 0 1 2 3 4 5 6 7 8 9 } |
| LISTA -> PRECIO R1 | P(LISTA) = P(PRECIO R1) = { 0 1 2 3 4 5 6 7 8 9 } |
| R1 -> **,** LISTA | P(R1) = P( **,** LISTA ) U P(ε) = { **,** } U { ε } |
| PRECIO -> DIGITO R2 | P(PRECIO) = P(DIGITO R2) = {0 1 2 3 4 5 6 7 8 9 } |
| R2 -> PRECIO | ε | P(R2) = P(PRECIO) U P(ε) = {0 1 2 3 4 5 6 7 8 9 } U { ε } |
| DIGITO -> 0 | 1 | … | 9 | P(DIGITO) = {0 1 2 3 4 5 6 7 8 9} |

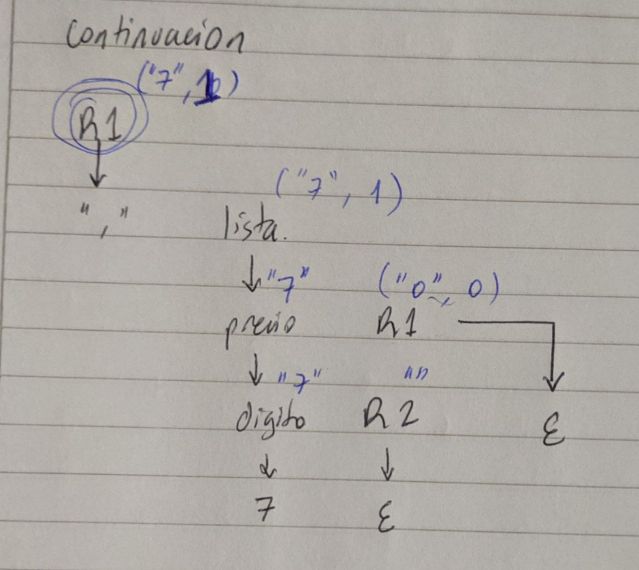
Ya que corroboramos que la gramática es predictiva (viendo solamente la recursión por la izq y conjunto primero), escribimos el BNF junto a las Reglas emanticas:

|  |  |
| --- | --- |
| **BNF** | **Reglas Semanticas** |
| TDS -> LISTA | LINEA < entrada.length() ? error() :  print(LISTA.numero / LISTA.contador, LISTA.contador) |
| LISTA -> PRECIO R1 | LISTA.numero = PRECIO.numero1  + R1.numero2 != “” ? R1.numero2 : 0  LISTA.contador = 1 + R1.contador |
| R1 -> **,** LISTA | R1.numero = LISTA.numero  R1.contador = LISTA.contador |
| R1 -> e | R1.numero= ""  R1.contador = 0 |
| PRECIO -> DIGITO R2 | PRECIO.numero = DIGITO.numero1 || R2.numero2 |
| R2 -> PRECIO | R2.numero = PRECIO.numero |
| R2 -> ε | R2.numero = "" |
| DIGITO -> 0 | DIGITO = “0” |
| … | … |
| DIGITO -> 9 | DIGITO = “9” |

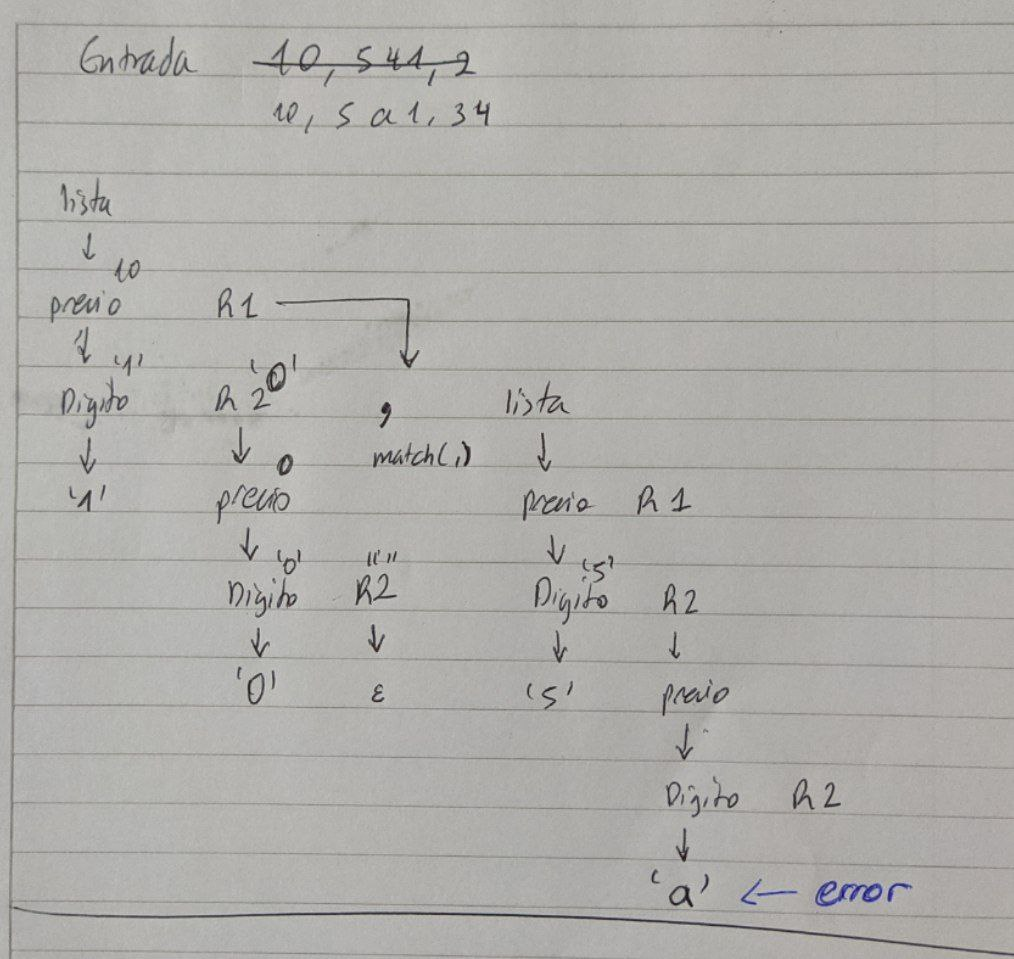
Ejemplos de Arboles Sintacticos

a. para una Entrada Valida: 150, 35, 7



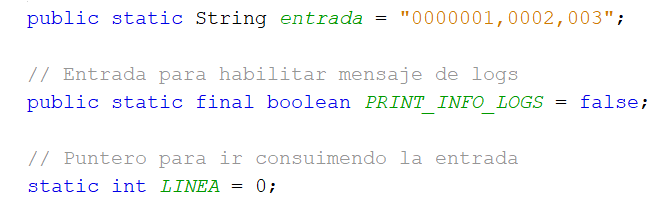


b. para una Entrada Nó Valida: 10, 5a, 90

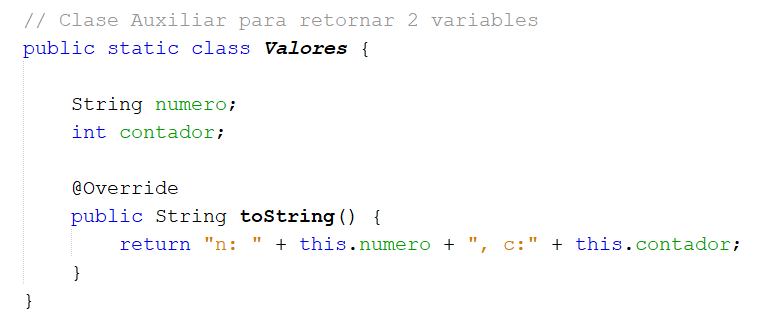


Ahora que tenemos el BNF, Conjunto Primero y las Reglas Semanticas procedemos a codificar nuestro TDS, para este caso estamos usando el lenguaje Java.

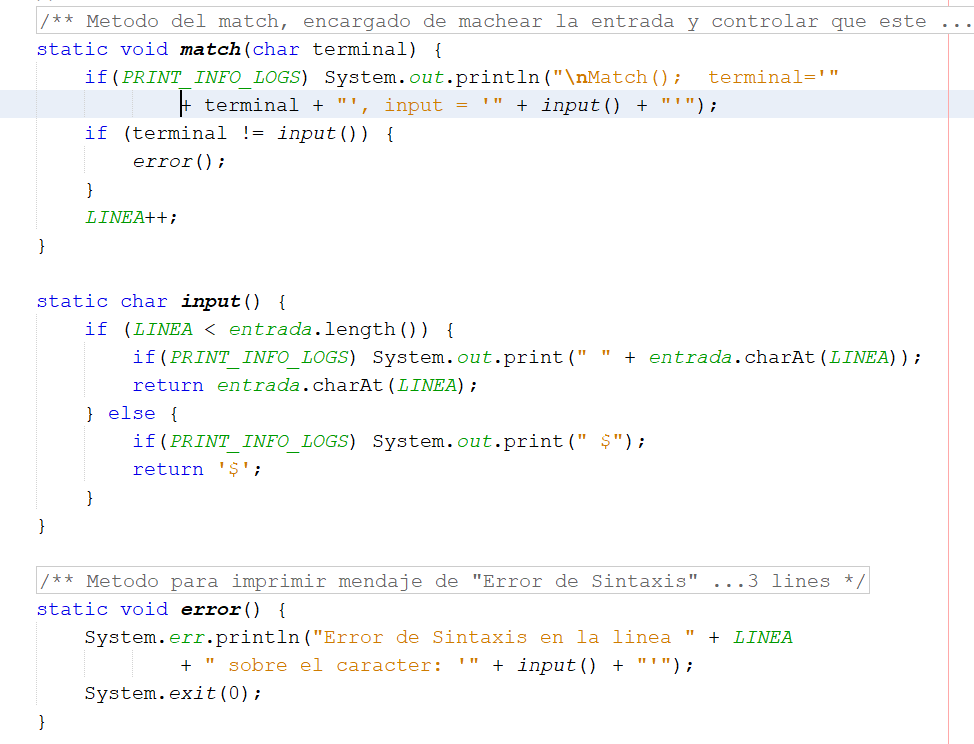
Definimos 3 variables globales, uno para guardar la entrada, otro de utilidad para imprimir LOGs y el puntero a la entrada.



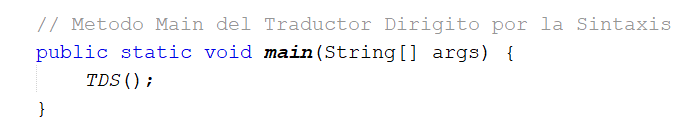
Estructura Auxiliar para hacer retorno de variables compuestas:



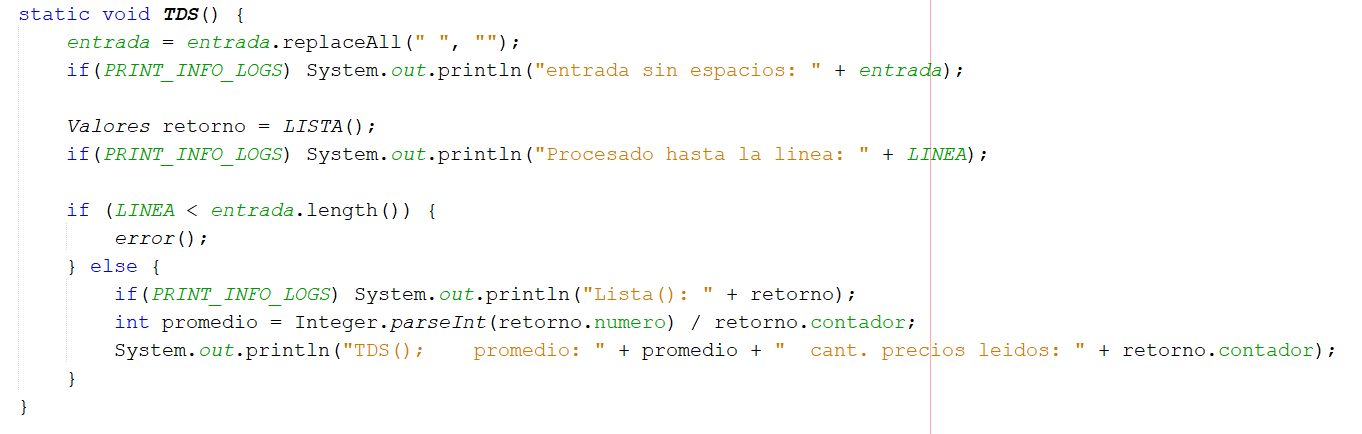
Funciones Match, Input y Error:

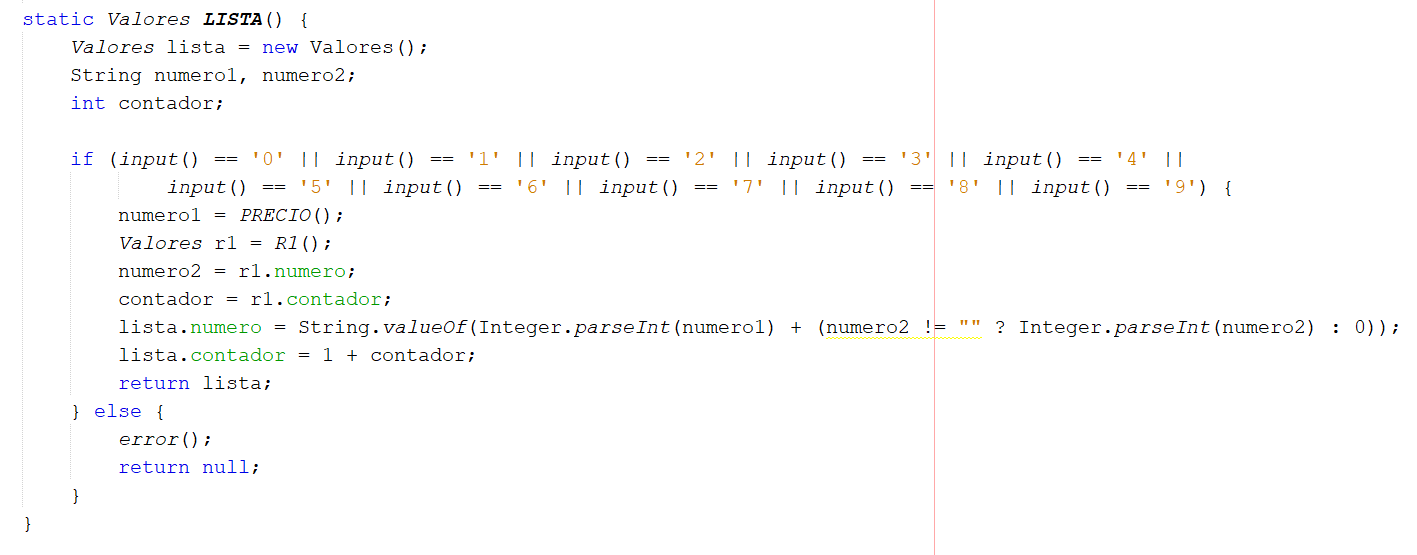


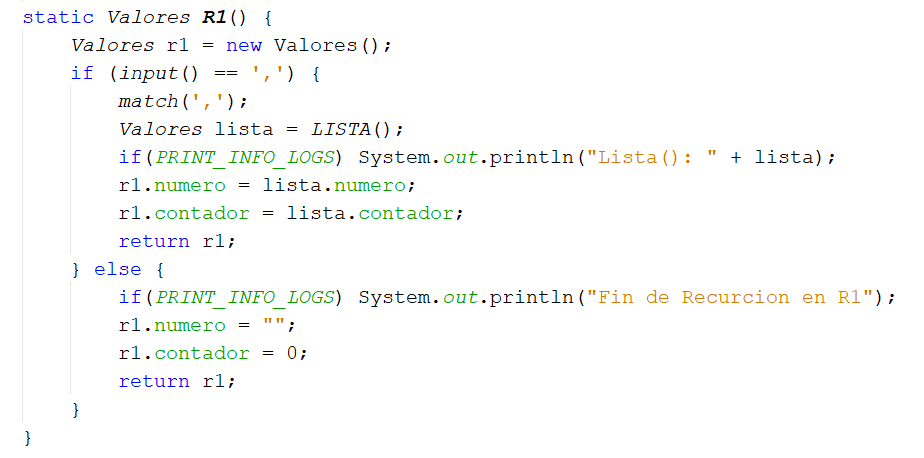
Metodo Main:

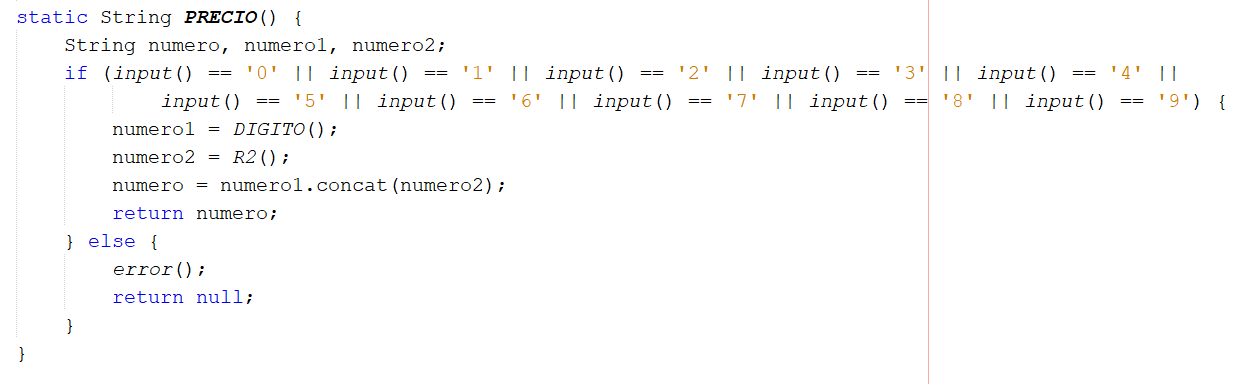


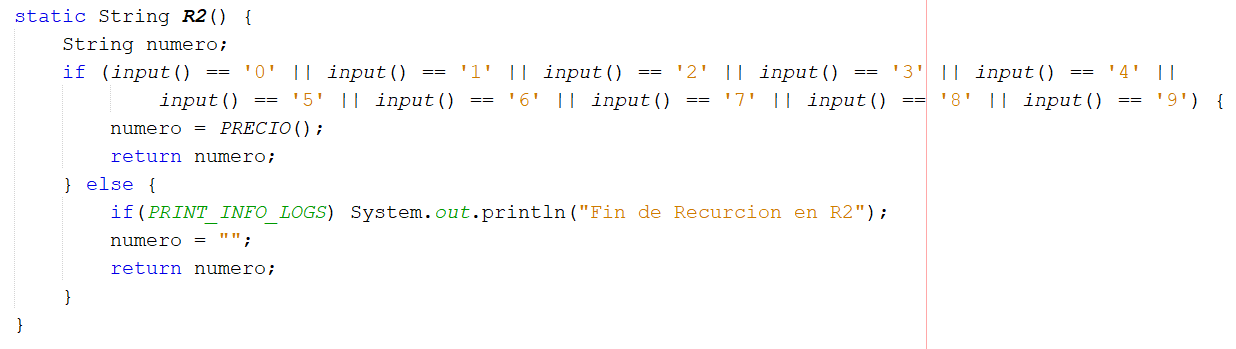
Producciones:

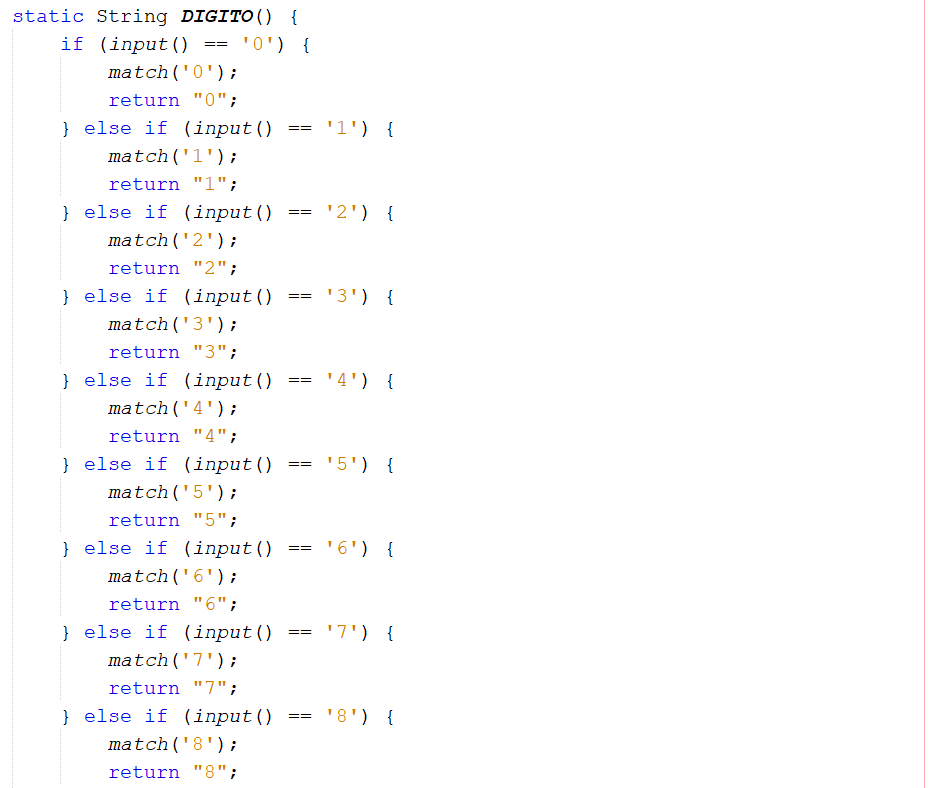


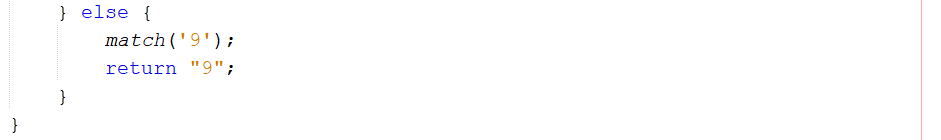












Para el caso que se quiera probar el TDS, tendra en un repositorio publico el codigo para probarlo. IDE usado seria NetBeans 8.2 sobre Java 8

Link: <https://github.com/heokim/compiladores-tp1-tds/blob/main/src/tds/TDS.java>