Lenguajes de Programación
PRACTICA 3, FASE C

Alonso Cifuentes, Roberto
Lobo Mata, Carlos
Muñoz Gozalo, Sergio
Urbón Domínguez, Fernando

Gramatica usada:

Para llegar a esta solución nos hemos basado en la solución propuesta para la práctica 3B. ANTLR4 permite transcribir la gramática a codigo ejecutable con gran facilidad. Añadiendo unas líneas de código al principio del programa para inicializar variables en las que guardar las llamadas encontradas (con un par de ArrayList) y con guardar la ID de las llamadas al encontrar un paréntesis en la regla código y más tarde filtrar por aquellas que sean métodos de la clase actual conseguimos lo propuesto.

Definición Dirigida por la Sintaxis, sobre gramática LL(1):

reglas sintácticas — acciones semánticas

prog → idspc CLASS ID { clase }

idspc → ID idspc | ; idspc | ε

clase → ID ID ids resto // retener los dos últimos id1 = id_1.lexema; id2 = id_2.lexema

| ε

resto → (params) def

| ; clase

def → { codigo } clase // recoger los dos últimos guarda(id1, id2)

| ; clase

ids → ID ids id1 = id2; id2 = $ID.text

| ε

codigo → ID codigo id3 = $ID.text;

| ; codigo

| (codigo) codigo //recoge el ID anterior guarda(id3)

| { codigo } codigo

| ε

params -> ID params

| ε

Tambien hemos añadido el analizador lexico correspondiente para poder encontrar CLASS e ID, asi como para borrar los comentarios de una o varias lineas (sacado de las gramaticas de https://github.com/antlr/grammars-v4), para borrar los saltos de linea y para borrar otros caracteres que no nos interesan.

Tablas de Análisis Sintáctico Predictivo:

FIRST	prog	idspc	clase	resto	def	ids	codigo	params
		ID ; ε	ID ε	(;	{ ;	ID ε	ID ; ({ ε	ID ε

FOLLOW	prog	idspc	clase	resto	def	ids	codigo	params
	$	CLASS	}			(;	})	)
				}	}			

TASP	ID	;	CLASS	(	)	{	}	$
prog	1	1						
idspc	1	2	ε					

clase	1					ε		
resto		2		1				
def		2				1		
ids	1	ε		ε				
codigo	1	2		3	ε	4	ε	
params	1				ε			

El número corresponde al numero de regla de cada auxiliar.

Ejecucion y aclaraciones:

Para ejecutar se debe ejecutar ANTLR4 para generar los .java, después compilar y después usar el comando grun (proporcionado por ANTLR4) redireccionando la entrada estándar al fichero a analizar:

```
$antlr4 Practica3c.g4                    //Generas .java
$javac Practica3c*.java                  //Compilas
$grun Practica3c prog < archivo.java     //Analiza el fichero
```

El programa devuelve los métodos y las llamadas de la forma:

```
metodo1
        llamada1
        llamada2
        llamada3
metodo2
        llamada1
        llamada2
        llamada3
        llamada4
metodo3
        llamada1
metodo4
...
```

También escribe en el fichero funciones.txt lo mismo.