Um Baita Exercício Auxiliar

Linguagem original

P--> C $
C --> "public" "class" ID (CD)* "end"
CD --> "SystemOutDispln" "(" E ")" ";"
E --> E Op E | ConstNumInt | ID
Op --> ">" | ">=" | "*" | "+" |

Obs1.: ID é um token para nomes de variáveis e nome da classe
Obs2.: ConstNumInt é um token para constantes numéricas

Vamos deixar a linguagem sem recursão à esquerda, problemas com precedência ou fatoração à esquerda:

Tratando a expressão regular de Fecho de Kleene sobre o CD:
comentário: a ER (CD)* significa que CD pode aparecer zerou ou mais vezes. Então, uma gramática correspondente a essa ER é:

LC --> CD LC | ε

Portanto,

P --> C $
C --> "public" "class" ID LC "end"
LC --> CD LC | ε
CD --> "SystemOutDispln" "(" E ")" ";"
E --> E Op E | ConstNumInt | ID
Op --> ">" | ">=" | "*" | "+"

Eliminando a ambiguidade sobre a precedência de operadores:
comentário: > e >= tem menor precedência que +, que por sua vez tem menor precedência que *.

P --> C $
C --> "public" "class" ID LC "end"
LC --> CD LC | ε
CD --> "SystemOutDispln" "(" E ")" ";"
E --> E ">" E_1 | E ">=" E_1 | E_1
E1 --> E_1 "+" E_2 | E_2
E2 --> E_2 "*" E_3 | E_3
E3 --> ConstNumInt | ID

Eliminando a recursividade à esquerda:
comentário: procuramos produções do tipo

$A \rightarrow A\alpha \mid \beta$

para substituir por

$A \rightarrow \beta A'$
$A' \rightarrow \alpha A' \mid \varepsilon$

Esse tipo de produção acontece em:

E --> E ">" E_1 | E ">=" E_1 | E_1
E1 --> E_1 "+" E_2 | E_2
E2 --> E_2 "*" E_3 | E_3

Logo,

P --> C $
C --> "public" "class" ID LC "end"
LC --> CD LC | ε
CD --> "SystemOutDispln" "(" E ")" ";"
E --> E_1 E'
E' --> ">" E_1 E' | ">=" E_1 E' | ε
E1 --> E_2 E_1'
E1' --> "+" E_2 E_1' | ε
E2 --> E_3 E_2'
E2' --> "*" E_3 E_2' | ε
E3 --> ConstNumInt | ID

Obs3.: Nesse exemplo, não tivemos problemas de fatoração à esquerda.

Vamos calcular FIRST e FOLLOW:

	FIRST	FOLLOW
P	public	$
C	public	$
LC	SystemOutDispln, ε	end
CD	SystemOutDispln	SystemOutDispln, end
E	ConstNumInt, ID	)
E’	>, >=, ε	)
E_1	ConstNumInt, ID	>, >=,)
E_1’	+, ε	>, >=,)
E_2	ConstNumInt, ID	+, >, >=,)
E_2’	*, ε	+, >, >=,)
E_3	ConstNunInt, ID	*, +, >, >=,)

Vamos calcular a TP:

	public	**class**	**ID**	**end**	**SystemOutDispln**	**(**	**)**	**;**	**>**	**>=**	**+**	*****	**ConstNumInt**	**end**	**$**
P	C														
C	public class ID LC end														
LC					CD LC									ε	
CD					SystemOutDispln (E);										
E			E_1E'										E_1E'		
E'							ε		$>E_1E'$	$>=E_1E'$					
E_1			E_2E_1'										E_2E_1'		
E_1'							ε		ε	ε	$+E_2E_1'$				
E_2			E_3E_2'										E_3E_2'		
E_2'							ε		ε	ε	ε	$*E_3E_2'$			
E_3			ID										ConstNumInt		

Vamos testar a sintaxe do código:

```
public class Teste
    SystemOutDispln(0 + 42 * 1);
end$
```

PILHA	ENTRADA	AÇÂO
$P	public	C
$C	public	public class ID LC end
$end LC ID class public	public	casou terminal
$end LC ID class	class	casou terminal
$end LC ID	Teste	casou terminal
$end LC	SystemOutDispln	CD LC
$end LC CD	SystemOutDispln	SystemOutDispln (E);
$end LC ;) E (SystemOutDispln	SystemOutDispln	casou terminal
$end LC ;) E (	(	casou terminal
$end LC ;) E	0	E_1E'
$end LC ;) E' E_1	0	E_2E_1'
$end LC ;) E' E_1' E_2	0	E_3E_2'
$end LC ;) E' E_1' E_2' E_3	0	casou terminal
$end LC ;) E' E_1' E_2'	+	ε
$end LC ;) E' E_1'	+	$+E_2E_1'$
$end LC ;) E' E_1' E_2 +	+	casou terminal
$end LC ;) E' E_1' E_2	42	E_3E_2'
$end LC ;) E' E_1' E_2' E_3	42	casou terminal
$end LC ;) E' E_1' E_2'	*	$*E_3E_2'$
$end LC ;) E' E_1'E_2' E_3 *	*	casou terminal
$end LC ;) E' E_1'E_2' E_3	1	casou terminal
$end LC ;) E' E_1'E_2'	)	ε
$end LC ;) E' E_1'	)	ε
$end LC ;) E'	)	ε
$end LC ;)	)	casou terminal
$end LC ;	;	casou terminal
$end LC	end	ε
$end	end	casou terminal
$	$	