Linguagem original:

Programa --> Classe EOF
Classe --> "public" "class" ID (DeclaraVar)* (Cmd)* "end"
DeclaraVar --> TipoPrimitivo ID ";"
TipoPrimitivo --> numInt | boolean | String
Cmd --> CmdDispl | CmdAtrib
CmdDispln --> "SystemOutDispln" "(" Expressao ")" ";"
CmdAtrib --> ID "=" Expressao ";"
Expressao --> Expressao Op Expressao | ConstNumInt | ConstString
Op --> ">" | ">=" | "*" | "+" | ID

Tratando a expressão regular de Fecho de Kleene sobre o DeclaraVar e o Cmd:

Programa --> Classe EOF
Classe --> "public" "class" ID ListaDeclaraVar ListaCmd "end"
ListaDeclaraVar --> TipoPrimitivo ID ";" ListaDeclaraVar | λ
TipoPrimitivo --> numInt | boolean | String
ListaCmd --> CmdDispln ListaCmd | CmdAtrib ListaCmd | λ
CmdDispln --> "SystemOutDispln" "(" Expressao ")" ";"
CmdAtrib --> ID "=" Expressao ";"
Expressao --> Expressao Op Expressao | ConstNumInt | ConstString
Op --> ">" | ">=" | "*" | "+" | ID

Eliminando a ambiguidade sobre a precedência de operadores (Conforme o link da UFSCar):

Programa --> Classe EOF
Classe --> "public" "class" ID ListaCmd "end"
ListaDeclaraVar --> TipoPrimitivo ID ";" ListaDeclaraVar | λ
TipoPrimitivo --> numInt | boolean | String
ListaCmd --> CmdDispln ListaCmd | CmdAtrib ListaCmd | λ
CmdDispln --> "SystemOutDispln" "(" Expressao ")" ";"
CmdAtrib --> ID "=" Expressao ";"
Expressao --> Expressao ">" Expressao1 | Expressao ">=" Expressao1 | Expressao1
Expressao1 --> Expressao1 "+" Expressao2 | Expressao2
Expressao2 --> Expressao2 "*" Expressao3 | Expressao3
Expressao3 --> ConstNumInt | ConstString | ID

Eliminando a recursividade à esquerda:

Programa --> Classe EOF
Classe --> "public" "class" ID ListaDeclaraVar ListaCmd "end"
ListaDeclaraVar --> TipoPrimitivo ID ";" ListaDeclaraVar | λ
TipoPrimitivo --> numInt | boolean | String
ListaCmd --> CmdDispln ListaCmd | CmdAtrib ListaCmd | λ
CmdDispln --> "SystemOutDispln" "(" Expressao ")" ";"
CmdAtrib --> ID "=" Expressao ";"
Expressao --> Expressao1 Expressao'
Expressao' --> ">" Expressao1 Expressao' | ">=" Expressao1 Expressao' | λ
Expressao1 --> Expressao2 Expressao1'
Expressao1' --> "+" Expressao2 Expressao1' | λ
Expressao2 --> Expressao3 Expressao2'
Expressao2' --> "*" Expressao3 Expressao2' | λ
Expressao3 --> ConstNumInt | ConstString | ID

Nesse exemplo, não teve problemas de fatoração.