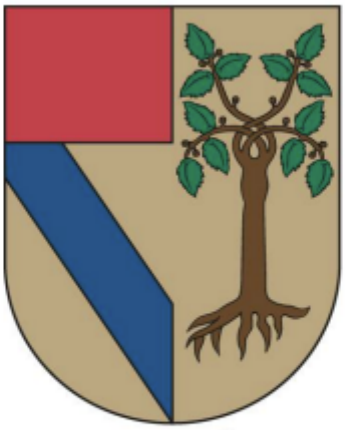


UNIVERSIDAD
PANAMERICANA

Syntax Diagrams Parser

Second Partial

Students:

Renata Calderón Mercado

Regina Ochoa Gispert

Miguel Angel Velez Olide

Professor:

Javier Gonzales Sanchez

Date of Delivery:

Sunday, March 30, 2024

Diagrama de sintaxis fue hecho en esta página:
https://jacquev6.github.io/DrawGrammar/

Código para hacer el diagrama:

Program = { GlobalVariableDeclaration }, ClassDeclaration;

GlobalVariableDeclaration = DataType, "id", ["=", Expression], ";";

ClassDeclaration = "class", "id", "{", {MethodDeclaration}, "}";

MethodDeclaration = Modifiers, DataType, "id", "(", [ParameterList], ")", MethodBody;

Modifiers = {"public" | "private" | "static" | "final" };

DataType = Primitive | "void" ;

Primitive = "KEYWORD" | "int" | "boolean" | "char" | "double" | "float" | "byte" | "short" | "long" | "void";

Value = "Integer" | "Bool" | "String" | "Float" | "Char" | "Octal" | "Hex" ;

ParameterList = Parameter, { "," , Parameter};

Parameter = DataType, "id";

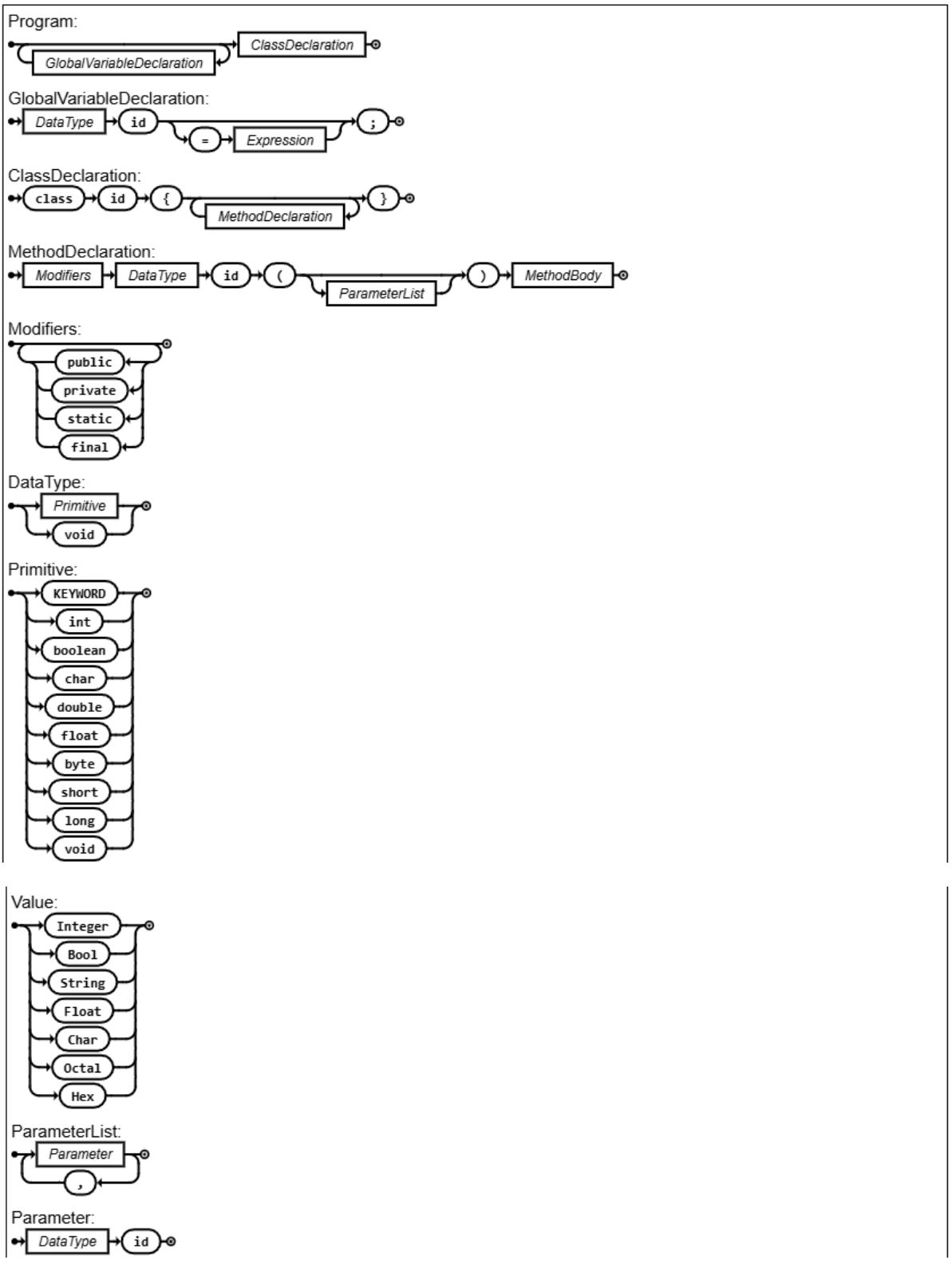
MethodBody = "{", { Statement }, "}" | Statement;

Statement = VariableDeclaration | ReturnStatement | IfStatement | WhileStatement | DoWhileStatement | ForStatement | SwitchStatement | CallMethod |

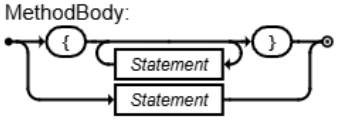
```
AssignmentStatement | ExpressionStatement;

VariableDeclaration = DataType, "id", ["=", Expression], ";";

ReturnStatement = "return", [Expression], ";";

AssignmentStatement = "id", "=", Expression, ";";

ExpressionStatement = Expression, ";";

IfStatement = "if", "(", Expression, ")",
("{", { Statement }, "}" | Statement),
["else", ("{", { Statement }, "}" | IfStatement | Statement)];

WhileStatement = "while", "(", Expression, ")",
("{", { Statement }, "}" | Statement);
DoWhileStatement = "do", ("{", { Statement }, "}" | Statement),
"while", "(", Expression, ")", ";";

ForStatement = "for", "(",
[TypedVar | Assignment | Expression], ";",
[Expression], ";",
[Assignment | Expression],
")", ("{", { Statement }, "}" | Statement);

TypedVar = DataType, "id", ["=", Expression];

Assignment = "id", "=", Expression;

SwitchStatement = "switch", "(", Expression, ")", "{",
{ CaseBlock },
```

```
"}";

CaseBlock = ("case", ("INTEGER" | "CHAR" | "STRING" | "IDENTIFIER"), ":" |
"default", ":"),
{ Statement },
["break", ";"];

CallMethod = "id", "(", [ParamValues], ")", ";";

ParamValues = Expression, { ",", Expression };

Expression = X, { "||", X };

X = Y, { "&&", Y };

Y = { "!" }, R;

R = E, { ("<" | ">" | "==" | "!=" | "<=" | ">="), E };

E = A, { ("+" | "-"), A };

A = B, { ("*" | "/"), B };

B = ["-"], C;

C = "id", ["(", [Arguments], ")"] |
("INTEGER" | "FLOAT" | "CHAR" | "STRING" | "BINARY" | "OCTAL" |
"HEXADECIMAL") |
"(", Expression, ")";

Arguments = Expression, { ",", Expression };
```

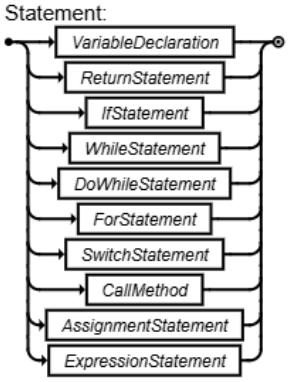
Imagen del diagrama:



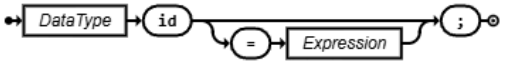
MethodBody:



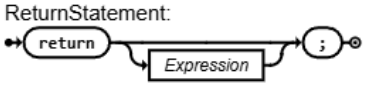
Statement:



VariableDeclaration:



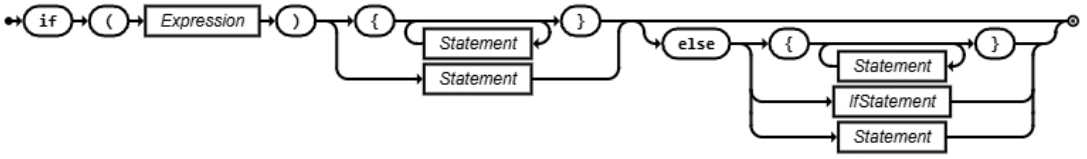
ReturnStatement:



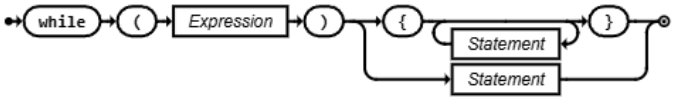
AssignmentStatement:

ExpressionStatement:

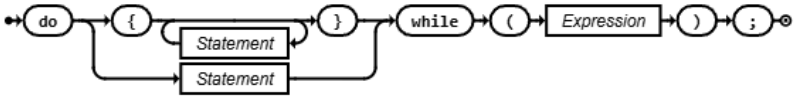
IfStatement:



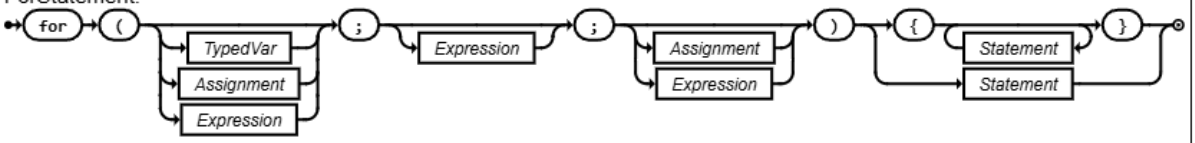
WhileStatement:



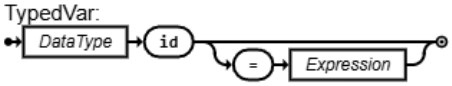
DoWhileStatement:



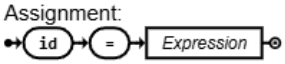
ForStatement:



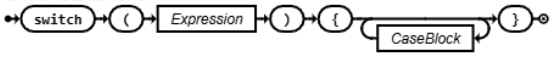
TypedVar:



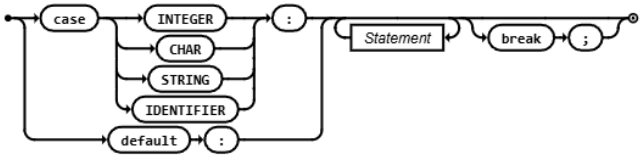
Assignment:



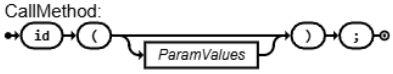
SwitchStatement:



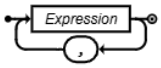
CaseBlock:



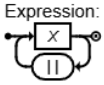
CallMethod:



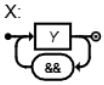
ParamValues:



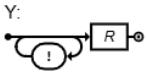
Expression:



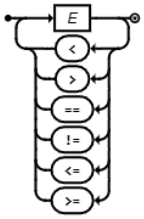
X:



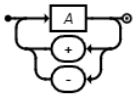
Y:



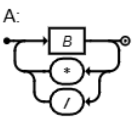
R:



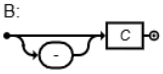
E:



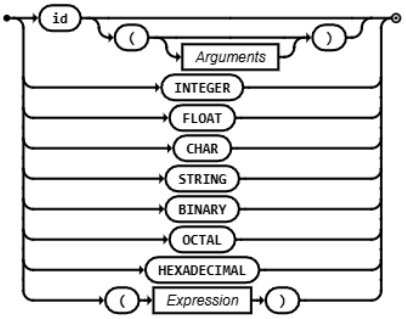
A:



B:



C:



Arguments: