أمثلة المحلل اللغوي (Parser Examples) - مراجعة البنية

• قواعد النحو (Grammar)

> ***ملاحظة هامة حول التعليقات:*** *بناءً على طلبك، تم تعديل قواعد النحو لافتراض أن التعليقات (التي تبدأ بـ `//` وتستمر حتى نهاية السطر) يتم التعامل معها وتجاهلها بالكامل بواسطة* ***المحلل اللغوي (Lexer)****. هذا يعني أن الـ `Parser` لن يتلقى أي رموز (Tokens) تمثل التعليقات، وبالتالي يمكن كتابة التعليقات في أي مكان في الكود دون التأثير على بنية الجملة النحوية.*

هذه القواعد مستخلصة من تنفيذ المحلل اللغوي (Parser) في ملف `pasted_content_2.txt`، وتم تعديلها لتعكس البنية المطلوبة (الكلاسات والدالة الرئيسية).

البرنامج (Program)

Program → ClassList EOF

تعريف العيلة (Class Definition)

ClassDefinition → `عيلة` Identifier `{` FunctionList `}`

قائمة الكلاسات (ClassList)

ClassList → ClassDefinition ClassList | ε

قائمة الدوال (FunctionList)

FunctionList → FunctionStmt FunctionList | ε

> ***ملاحظة:*** *يجب أن تحتوي قائمة الدوال في* ***إحدى*** *الكلاسات على دالة واحدة فقط اسمها `ساموعليكم` (الدالة الرئيسية).*

تعريف الدالة (FunctionStmt)

FunctionStmt → Identifier `(` ParameterList `)` BlockStmt [Type] حيسبة [جاعد]

قائمة المعاملات (ParameterList)

ParameterList → [Parameter (`,` Parameter)*]

Parameter → Type Identifier

> ***ملاحظة:*** *في الكود البرمجي، تم افتراض أن المعاملات يجب أن تكون من نوع* `رقم` *(TokenType.Rakam) لتبسيط عملية التحليل. يمكن توسيعها لتشمل أنواعًا أخرى لاحقًا.*
>
> ***حيث:***
>
> - ***Identifier*** *هو اسم الدالة.*
> - ***الدالة الرئيسية:*** *يجب أن تكون دالة واحدة فقط باسم* `ساموعليكم` *.*
> - `جاعد` *تشير إلى* `static` *.*

الكتلة البرمجية (BlockStmt)

BlockStmt → `{` StatementList `}`

قائمة الجمل (StatementList)

StatementList → Statement StatementList | ε

> ***ملاحظة:*** *هناك تكرار (Recursion) هنا، نحتاج إلى تحديد متى يتوقف المحلل اللغوي عن قراءة الجمل. يتوقف عند نهاية الملف (EOF) أو عند إغلاق القوس المعقوف* `}` *.*

الجملة (Statement)

Statement → Declaration | Assignment | IfStmt | WhileStmt | ForStmt | PrintStmt | ReturnStmt | BlockStmt

التصريح عن متغير (Declaration)

Declaration → Type Identifier [`=` Expression] `;`

> ***حيث:***

- ***Type*** → `رقم` | `كلام` | `كسر` | `صح_غلط`

الإسناد (Assignment)

Assignment → Identifier `=` Expression `;`

جملة الشرط (IfStmt)

IfStmt → `لو` `(` Condition `)` Statement [`والا` Statement]

جملة التكرار (WhileStmt)

WhileStmt → `علطول` `(` Condition `)` Statement

جملة التكرار (ForStmt)

ForStmt → `لف` `(` [Declaration | Expression] `;` [Condition] `;` [Expression] `)` Statement

جملة الطباعة (PrintStmt)

PrintStmt → `اكتب` Expression `;`

جملة الإرجاع (ReturnStmt)

ReturnStmt → `الجوف` [Expression] `;`

الكتلة البرمجية (BlockStmt)s

التعبير (Expression)

Expression → Comparison

المقارنة (Comparison)

***Comparison** → Term (RelOp Term)

حيث:

- ***RelOp*** → `==` | `=!` | `<` | `=<` | `>` | `=>`

الحد (Term)

***Term** → Factor (AddOp Factor)

> *حيث:*
>
> - ***AddOp*** → `+` | `-`

العامل (Factor)

***Factor** → Postfix (MulOp Postfix)

> *حيث:*
>
> - ***MulOp*** → `*` | `/`

الزيادة اللاحقة (Postfix)

Postfix → Primary [`++`]

الأساسي (Primary)

Primary → Literal | Identifier | `(` Expression `)`

> *حيث:*
>
> - ***Literal*** → *NumberLiteral* | *StringLiteral*

• ملاحظات إضافية

الدالة الرئيسية (Main Function)

يجب أن يحتوي البرنامج على دالة واحدة فقط باسم `ساموعليكم` (SamoAlikom) لتمثل نقطة الدخول (Entry Point) للبرنامج، ويجب أن تكون هذه الدالة ضمن إحدى الكلاسات المعرفة.

هيكل شجرة بناء الجملة المجردة (AST Nodes)

لتمثيل البنية الجديدة متعددة الكلاسات، يجب أن تتضمن شجرة بناء الجملة المجردة (AST) العقد التالية:

- `ProgramNode` (يمثل البرنامج بالكامل، ويحتوي على قائمة الكلاسات)
- `ClassNode` (يمثل تعريف العيلة الواحدة)
- `FunctionNode`
- `VarDeclNode` (Declaration)
- `AssignmentNode`
- `PrintNode` (PrintStmt)
- `IfNode` (IfStmt)
- `WhileNode` (WhileStmt)
- `ForNode` (ForStmt)
- `ReturnNode` (ReturnStmt)
- `BlockNode` (BlockStmt)
- `BinaryOpNode` (للتعبيرات الثنائية مثل الجمع والمقارنة)
- `LiteralNode`
- `VariableNode`
- `PostfixIncrementNode`