المترجمات

دليل عملي مخصص للطلاب

باستخدام ال ANTRL

تعلم كيف تكتب لغتك الخاصة

سامي قزح

الفهرس :

المقدمة

خلال مسيرتي الجامعية واجهت الكثير من العقبات ، و المطبات الصعبة لكنها ما لبثت أن تذللت أمامي بفضل الإصرار ، و العمل الجاد بالإضافة إلى مساعدة و مساندة الأصدقاء و الزملاء الذين كانوا خير عون ، و خير سند خلال مسيرتي الدراسية ، في مروري الأخير للجامعة وجدت الكثير من الطلاب التائهين في هذه المادة بالرغم من أنها من أسهل المواد الجامعية التي قمت بدراستها و إنجاز مشروعها لذلك قررت أن أدون خبرتي بهذه المادة لأنقلها بشكلٍ يساعد من يقرأه على فهم كيفية عمل لغات البرمجة و كيف من الممكن أن نصنع لغةً بسيطة أو نضيف قواعداً جديدة ، حيث أنني سأشرح و بشكل تفصيلي المفاهيم الأساسية بالإضافة إلى طريقة كتابة الكود المصدري بالشكل الأبسط الأسلم بحيث يتناسب الشرح مع التباين في مستوى الفهم.

للقيام بإنجاز ما سلف سنحتاج إلى :

- بيئة تطوير (IDE) **Intellij** (يفضل استخدامه)
- **Java Jdk** (يمكنك استخدام أي لغة اخرى تدعم ال OOP)
- أداة **Antlr**(يمكن تثبيتها كإضافة Plugin موجودة ضمن ال IDE)
- قد تحتاج **Vpn** لتثبيت الأداة السابقة (Proton vpn كاقتراح)
- **Antlr Jar** يجب أن يكون الإصدار متوافق مع إصدار أداة Antlr (قد لا تحتاجه في حال استخدام لغة غير الجافا "تتم اضافته في ال project structure")
- الرغبة في التعلم بالإضافة إلى طول البال .
- **عدم إهمال أي فصل من فصول الكتاب .**

الفصل الأول:
أساسيات

مفاهيم نظرية أساسية

- ماهي اللغة ؟

هي مجموعة من الرموز المحكية أو المكتوبة التي يستخدمها البشر للتواصل مع بعضهم كاللغات الطبيعية (مثل اللغة العربية) أو مع الآلة كاللغات الاصطناعية (مثل لغات البرمجة) .

- فروق بين اللغة العربية و لغات البرمجة:

١ ـ اللغة العربية لغة سياقية حيث يمكن لنفس الجملة أن تعطي أكثر من معنى و جميع هذه المعاني تكون صحيحة لغوياً ، كأن تقول : (فلان وحش)

إن معنى هذه الجملة يختلف باختلاف سياق الكلام ، فقد يكون وحشاً لأنه أنجز عملاً صعباً ، أو قد يكون وحشاً لأنه عمل عملاً شريراً .

٢ـ لغات البرمجة هي لغات خارج السياق Context Free Language أي لا يختلف معنى الجملة أو النص باختلاف السياق و يتم توليد اللغة باستخدام مجموعة من القواعد Context Free Grammar حيث تعبر هذه القواعد عن جميع الجمل الممكنة في اللغة.

- ما هو المترجم Compiler ؟

هو برنامج خاص يقوم بترجمة النص البرمجي المكتوب بلغة عالية المستوى (مثل الجافا) إلى لغة الآلة أي أننا سنقوم ببناء مترجم يحول نص مكتوب بلغة عالية المستوى إلى لغة ذات مستوى أدنى .

- خطوات بناء المترجم :

١ــ ترميز مفردات اللغة

٢ـ بناء قواعد اللغة

٣ـ التحقق من الصحة الدلالية للقواعد

٤ـ توليد الكود الجديد

مراجعة للتعابير المنتظمة
Regular Expressions

على سبيل المثال نريد ترميز كلمة (وحش) ب Monster فترمز ببساطة بكتابة ; ”وحش“ : Monster

أردنا فيما بعد ترميز (كلمة ”وحش“ بالإضافة إلى أي رقم من 0 إلى اللانهاية) ب Monster

لا يمكن هنا تعريف Monster عدد لانهائي من المرات لكن يمكن تعريف الحالة باستخدام التعابير المنتظمة على الشكل :

Monster : **(’0’ .. ‘9’)+** ”وحش“ ;

+(’0’ .. ‘9’) أي سلسلة أرقام مؤلفة من رقم واحد على الأقل .

في حال أردنا ترميز (كلمة ”وحش “ بالإضافة إلى أي رقم من 0 إلى اللانهاية أو **بدون أي رقم**) ب Monster فترمز ببساطة بكتابة :

Monster : **(’0’ .. ‘9’) *** ”وحش“ ;

*(’0’ .. ‘9’) أي سلسلة أرقام مؤلفة من رقم واحد أو السلسلة الفارغة .

في حال أردنا ترميز (كلمة ”وحش “ بالإضافة إلى أي رقم من 0 إلى اللانهاية **أو أي اسم باللغة الإنجليزية أو بدون أي رقم أو اسم**) ب Monster فترمز ببساطة بكتابة :

Monster : **((’0’ .. ‘9’) *| (‘a’..’z’|’A’..’Z’)*)** ”وحش“ ;

(*(‘a’..’z’|’A’..’Z’) | *(’0’ .. ‘9’)) أي سلسلة أرقام أو أحرف مؤلفة من رقم واحد أو السلسلة الفارغة .

في حال أردنا ترميز (كلمة ”وحش “ بالإضافة إلى سلسلة من **الأرقام و الأحرف أو فقط كلمة وحش**) ب Monster فترمز ببساطة بكتابة :

Monster : **((’0’ .. ‘9’) | ‘a’..’z’|’A’..’Z’)*** ”وحش“ ;

* (‘a’..’z’|’A’..’Z’) | (’9‘ .. ’0‘)) أي سلسلة أرقام أو أحرف مؤلفة من رقم واحد أو السلسلة الفارغة .

شرح ملفات ANTLR

- Antlr هو أداة للتعرف وتحليل اللغة ، و يستخدم في الأوساط الأكاديمية و العملية لبناء جميع أنواع اللغات .

- صيغة ملفاته g4.

- و بما أنه محلل لغوي فهو يحتوي على أداتين :

١– lexer أداة للتعرف على تسلسل الأحرف بتمثيل هذا التسلسل برمز(token)

NUMBER :('0'..'9')+ ;

EQ : "=" ؛

CHARS : ('a'..'z')+;

SEMICOLEN : ';' ;

٢– parser أداة للتعرف على تسلسل الرمز (token) وتمثيله بقاعدة

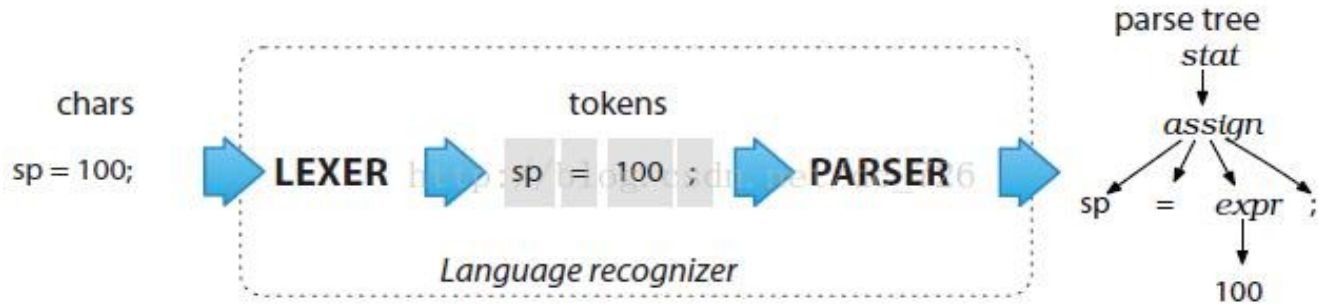
state : assign ;

expr :NUMBER;

assign : CHARS EQ expr SEMICOLEN ;

انتبه : تمثل الرموز (tokens) بحروف كبيرة بينما تمثل القواعد بحروف صغيرة .

- **تعرف الرموز ضمن ملف ال Lexer بينما تعرف القواعد ضمن ملف ال Parser الذي يستخدم الرموز التي عرفت مسبقاً ضمن ملف ال Lexer**



مسألة : نريد تحليل النص التالي :

Function (0) ;

الحل :

- أولا نقوم بتحليل النص إلى مجموعة من ال tokens

- function - - (- - 0 - -) - - ; -

- ثانياً نقوم بإنشاء LexerFile.g4 نعرف بداخله ال tokens :

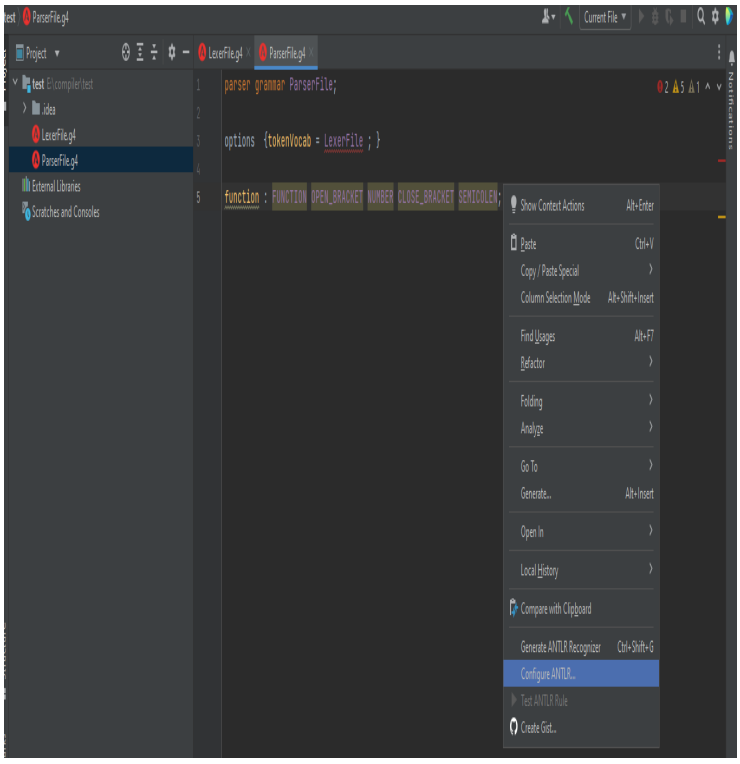
```
lexer grammar LexerFile;
NUMBER  :('0'..'9')+ ;
FUNCTION : 'function';
OPEN_BRACKET: '(';
CLOSE_BRACKET:')';
SEMICOLEN : ';'  ;
```

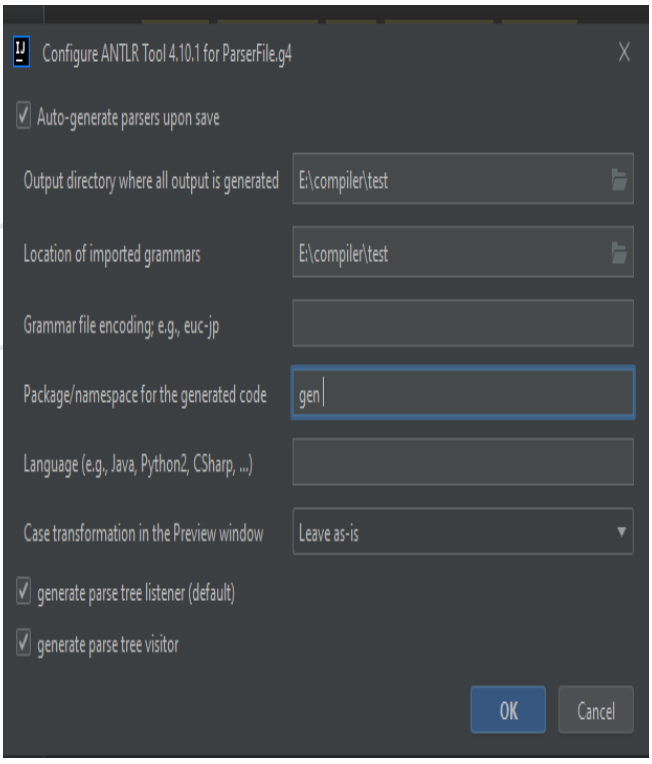
-ثالثاً نقوم بإنشاء ParserFile.g4 لكتابة القاعدة :

```
parser grammar ParserFile;
options  {tokenVocab = LexerFile ; }
function : FUNCTION OPEN_BRACKET NUMBER CLOSE_BRACKET SEMICOLEN;
```

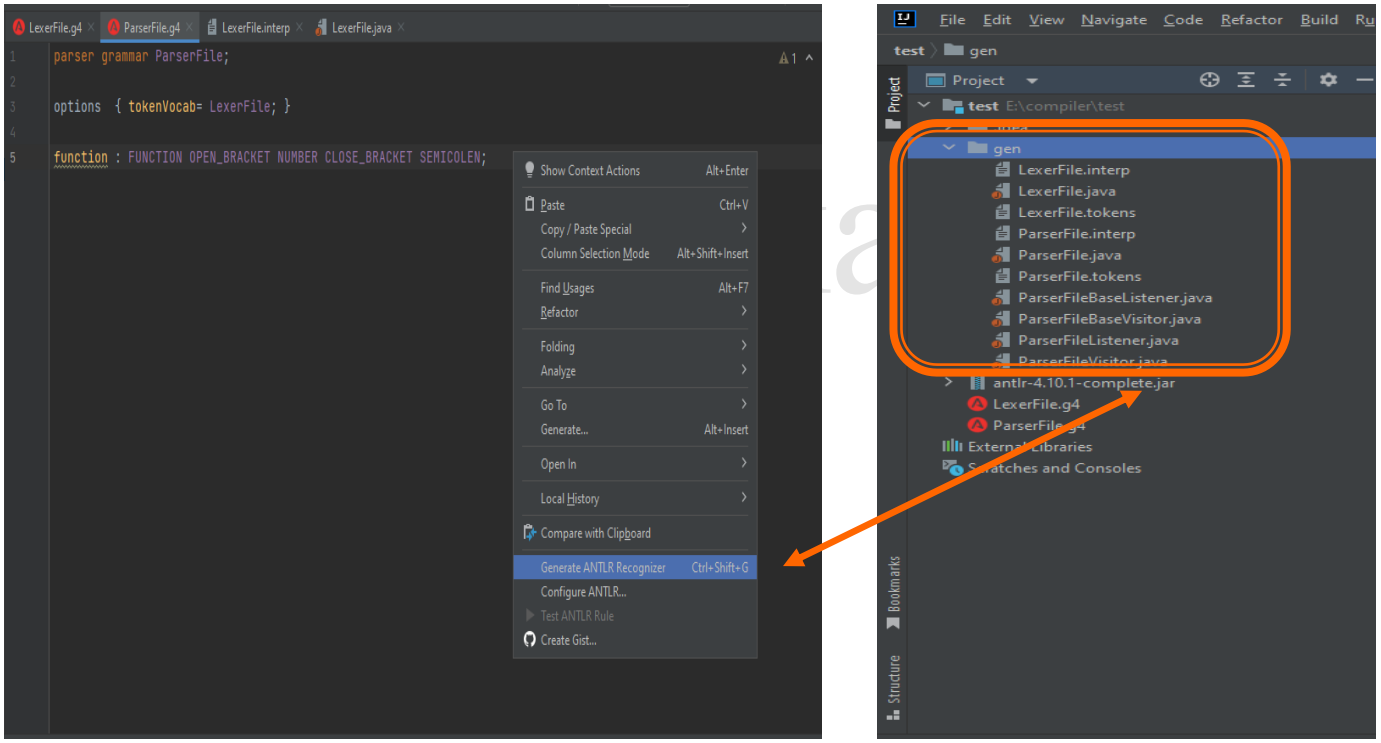
- **لاحظ أن Antlr يمثل ال Tokens باللون البنفسجي بينما يمثل القواعد باللون الأصفر .**

- رابعاً نقوم بتهيئة البرنامج لتوليد ال tokens

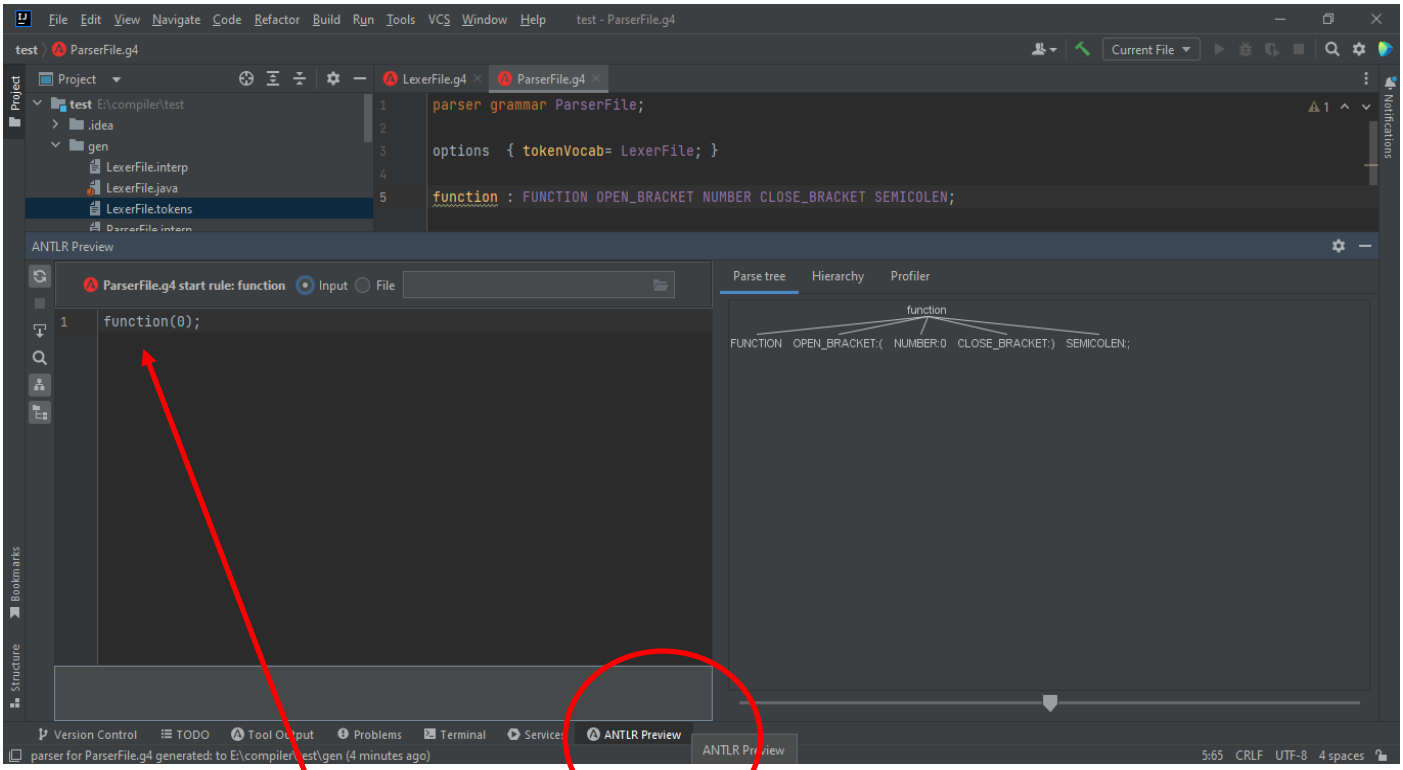




- خامساً نقوم بتوليد الملفات

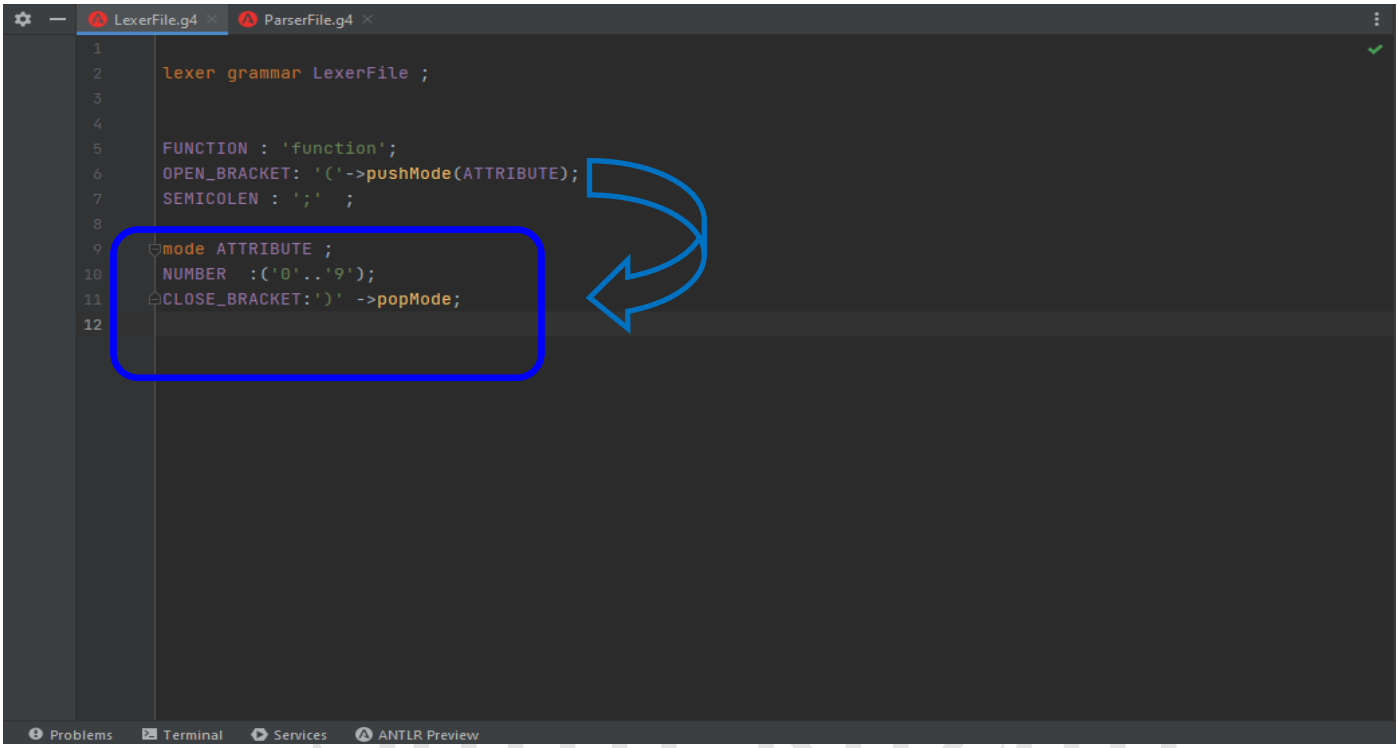


- سادساً نقوم بتوليد الشجرة



نضغط هنا ثم نكتب النص المراد تجربته

- يمكن تعريف ال Tokens ضمن ملف ال lexer بطريقة اخرى باستخدام ال mode الذي يقوم بصنع مكدس تكون عناصره منفصلة عن بقية العناصر الأخرى الموجودة بالملف و في مثالنا هنا يكون أول عنصر فيه هو "(" و آخر عنصر فيه هو ")" .



بمجرد كتابة القوس المفتوح "(" يدخل ضمن mode Attribute فلا يستطيع ال parser قراءة أي شيء موجود خارجه حتى الخروج منه باستخدام القوس المغلق عندها يعود إلى default mode

انتبه :

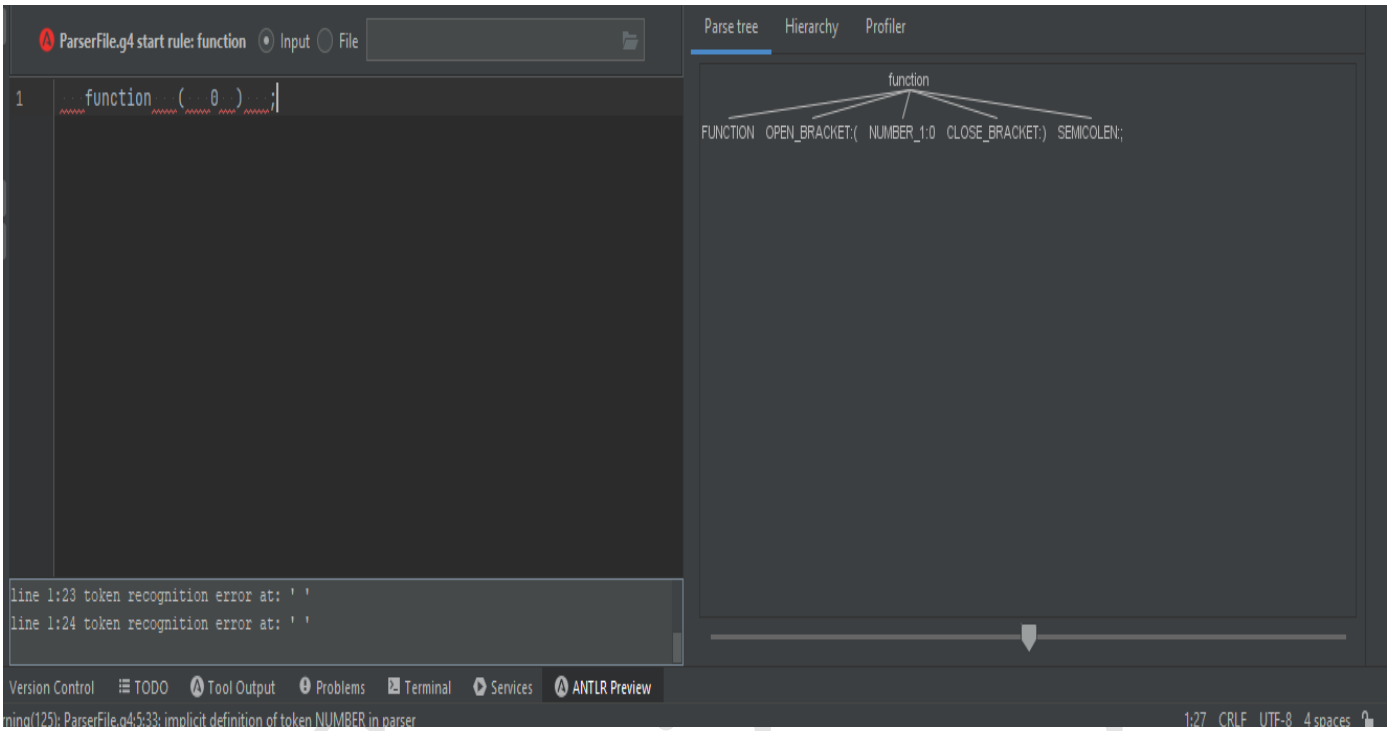
في حال كنت في ال default mode فأنت لا تستطيع الوصول إلى العناصر الموجودة في غيره ففي مثالنا السابق **لن تستطيع الوصول** إلى ال Number **بدون الدخول إلى mode attribute** .

فيكون الحل بإعادة تعريف ال Number في ال default mode على الشكل التالي :

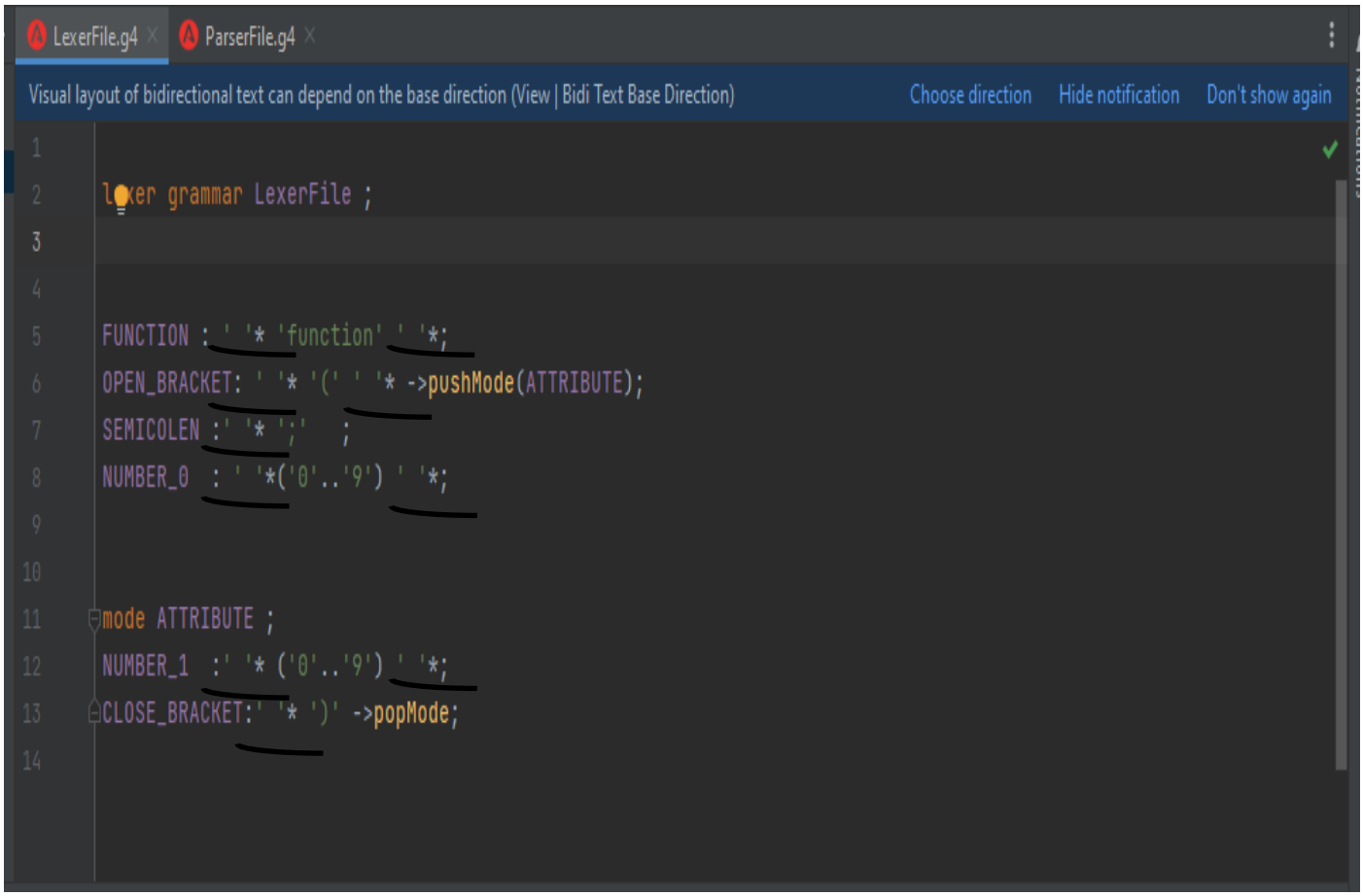
```
lexer grammar LexerFile ;

FUNCTION : 'function';
OPEN_BRACKET: '('->pushMode(ATTRIBUTE);
SEMICOLEN : ';' ;
NUMBER_0 :('0'..'9');

mode ATTRIBUTE ;
NUMBER_1 :('0'..'9');
CLOSE_BRACKET:')' ->popMode;
```

الأخطاء الشائعة (١)

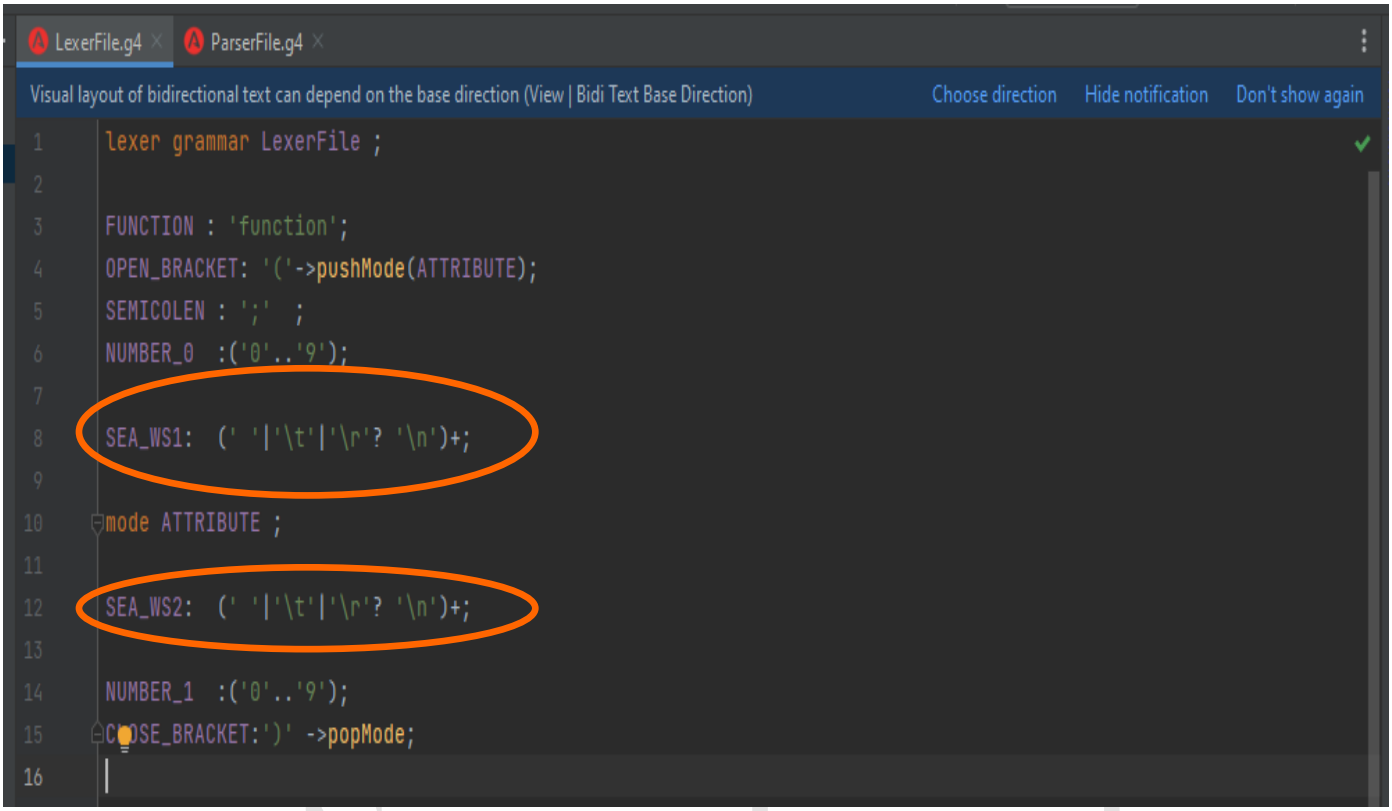
ـ عدم أخذ المسافات الفارغة بعين الاعتبار يؤدي إلى العديد من الأخطاء

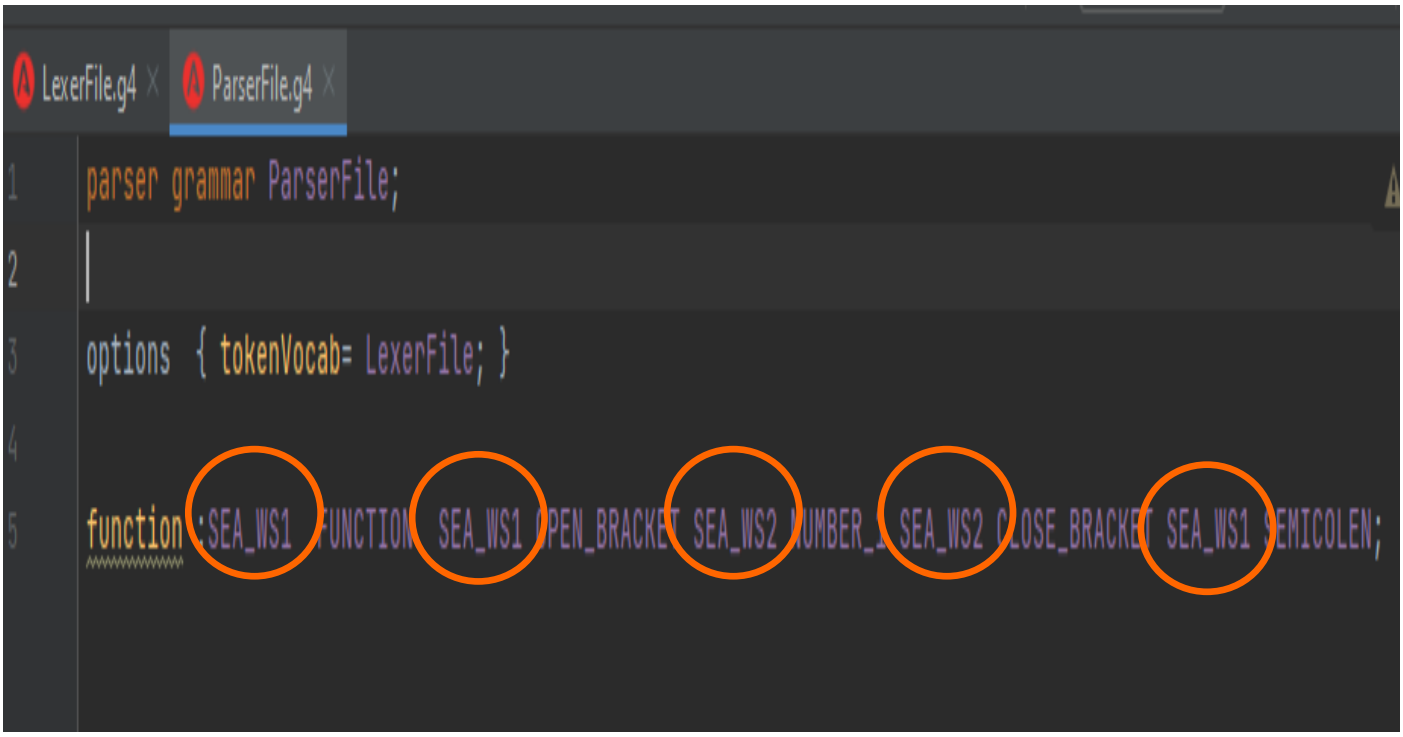


السبب لأن المسافات الفارغة هي tokens و منه يجب تعريفها ضمن ال Parser أو Lexer

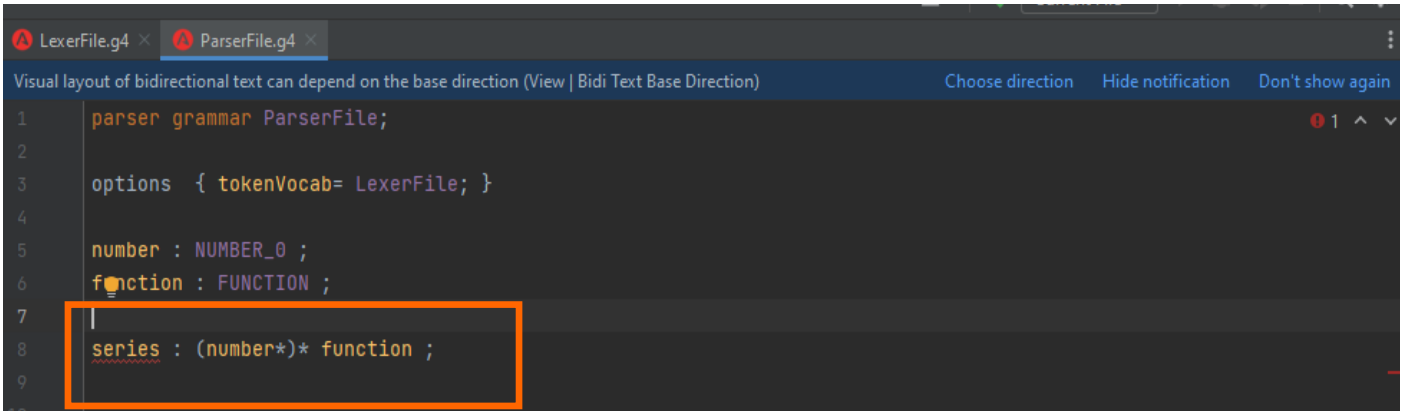


أو بال parser



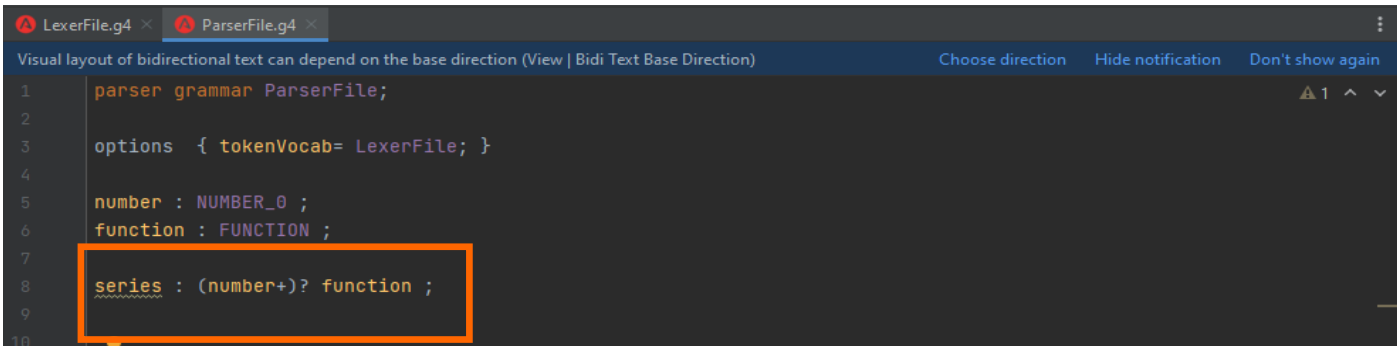


- السلسلة الفارغة غير مسموحة في ال Parser



في حال أردت أن تضع شرط ليتأكد من إمكانية ورود سلسلة لمرة واحدة فقط أو عدم ورودها إطلاقاً

يمكنك استخدام " ? " بالشكل التالي :



- عدم مراعاة أهمية ترتيب القواعد

لن تستطيع ابداً استخدام ال VAR لأن STRING تعبر عن نفس السلسلة بالإضافة إلى أن **لها أولوية الترتيب** .

- **عدم عمل popMode بعد الدخول إلى mode قد يمنعك من الوصول إلى باقي ال modes**

```
lexer grammar LexerFile ;

OPEN_BRACKET: '('->pushMode(ATTRIBUTE);

FUNCTION : 'function';
SEMICOLEN : ';'  ;
VAR : ('a'..'z')+;
QOUT: ' '* '\'' ' '*;
SEA_WS1:  (' '|'\t'|'\r'? '\n')+;



mode ATTRIBUTE ;
SEA_WS2:  (' '|'\t'|'\r'? '\n')+;
NUMBER_1  :('0'..'9');
CLOSE_BRACKET:')';
```

في المثال بمجرد دخولك إلى mode ATTRIBUTE عبر pushMode(ATTRIBUTE)

لن تستطيع الوصول سوى إلى العناصر الموجودة ضمن ال mode ATTRIBUTE فلكي تصل إلى

العناصر السابقة يجب وضع إمكانية الخروج من ال mode بعين الاعتبار.


```
lexer grammar LexerFile ;

OPEN_BRACKET: '('->pushMode(ATTRIBUTE);

FUNCTION : 'function';
SEMICOLEN : ';'  ;
VAR : ('a'..'z')+;
QOUT: ' '* '\'' ' '*;
SEA_WS1:  (' '|'\t'|'\r'? '\n')+;



mode ATTRIBUTE ;
SEA_WS2:  (' '|'\t'|'\r'? '\n')+;
NUMBER_1  :('0'..'9');
CLOSE_BRACKET:')'->popMode;
```

بحيث في حال أردت استخدام الرموز الموجودة في default mode بعد ولوجك إلى ATTRIBUTE

يمكنك الخروج من mode ATTRIBUE باستخدام CLOSE_BRACKET مما يعيدك إلى

default mode

الفصل الثاني:
بناء المترجم (١)

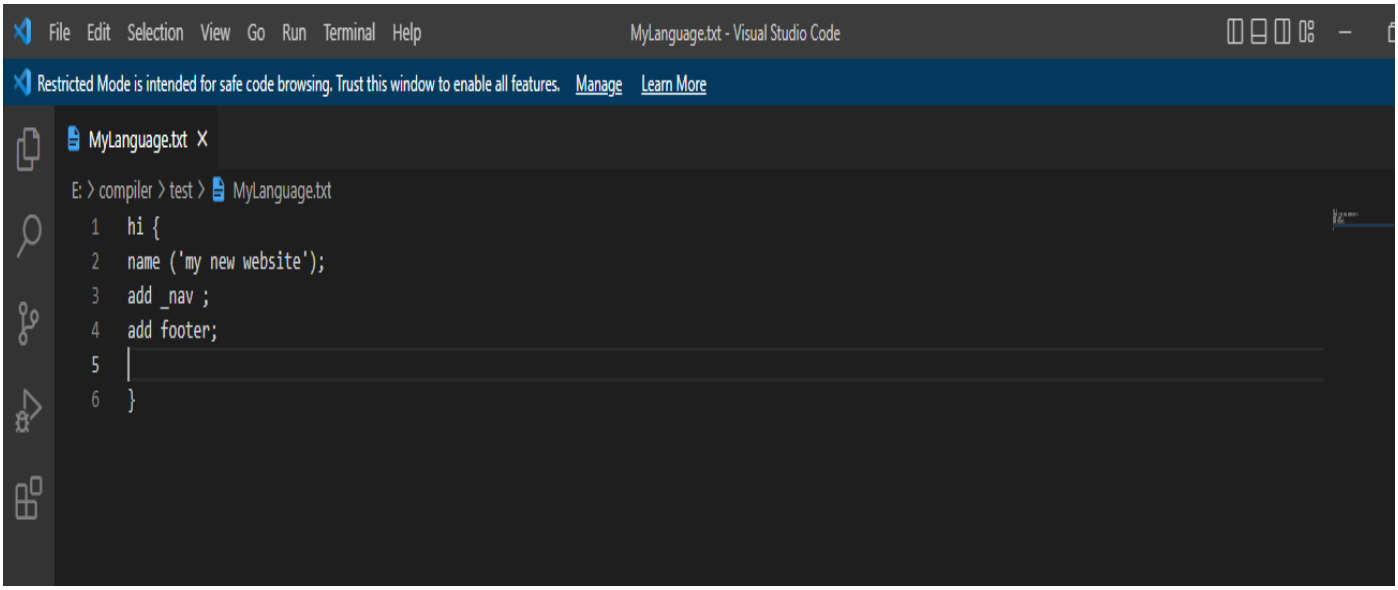
الإضافة على لغة أم كتابة لغة جديدة ؟

- يمكنك تحميل ملفات قواعد بعض اللغات أو النصوص مثل ال html / dart / java script

https://github.com/antlr/grammars-v4

حيث يمكنك مثلاً تحميل قواعد نصوص html و بعدها تقوم بابتكار و إضافة قواعد جديدة لعمل شيء يشبه ال Angular مثلاً .

نحن سنقوم في هذا الكتاب بعمل مترجم للغة جديدة تقوم بتوليد واجهات لصفحة ويب ، حيث أنها ستكون على الشكل التالي :



حيث hi هي صفحة ويب ، يتم ادخال اسمها ضمن ال name او تأخذ home في حال عدم ادخال أي اسم، ويمكن أن يتم إضافة navigation bar إليها بكتابة add nav ، و يمكن إضافة footer إليها بإضافة add footer.

- الآن نقوم بتجهيز البيئة (علماً أنه تم تحميل Antlr plugin مسبقاً):

١- نقوم بإنشاء مجلد فارغ نسميه MyNewLanguage

٢- نقوم بإنشاء ملفين MylanguageParser.g4 MylanguageLexer.g4

٣- نقوم بوضع ملف ال Antlr jar ضمن المجلد

٤- نفتح مجلد MyNewLanguage باستخدام برنامج Intellij >> نضغط على file >> project structure >> modules >> dependencies >> **إشارة** + >> نضيف ملف Antlr jar

الآن نقوم بتعريف ال tokens ضمن ملف ال Lexer:

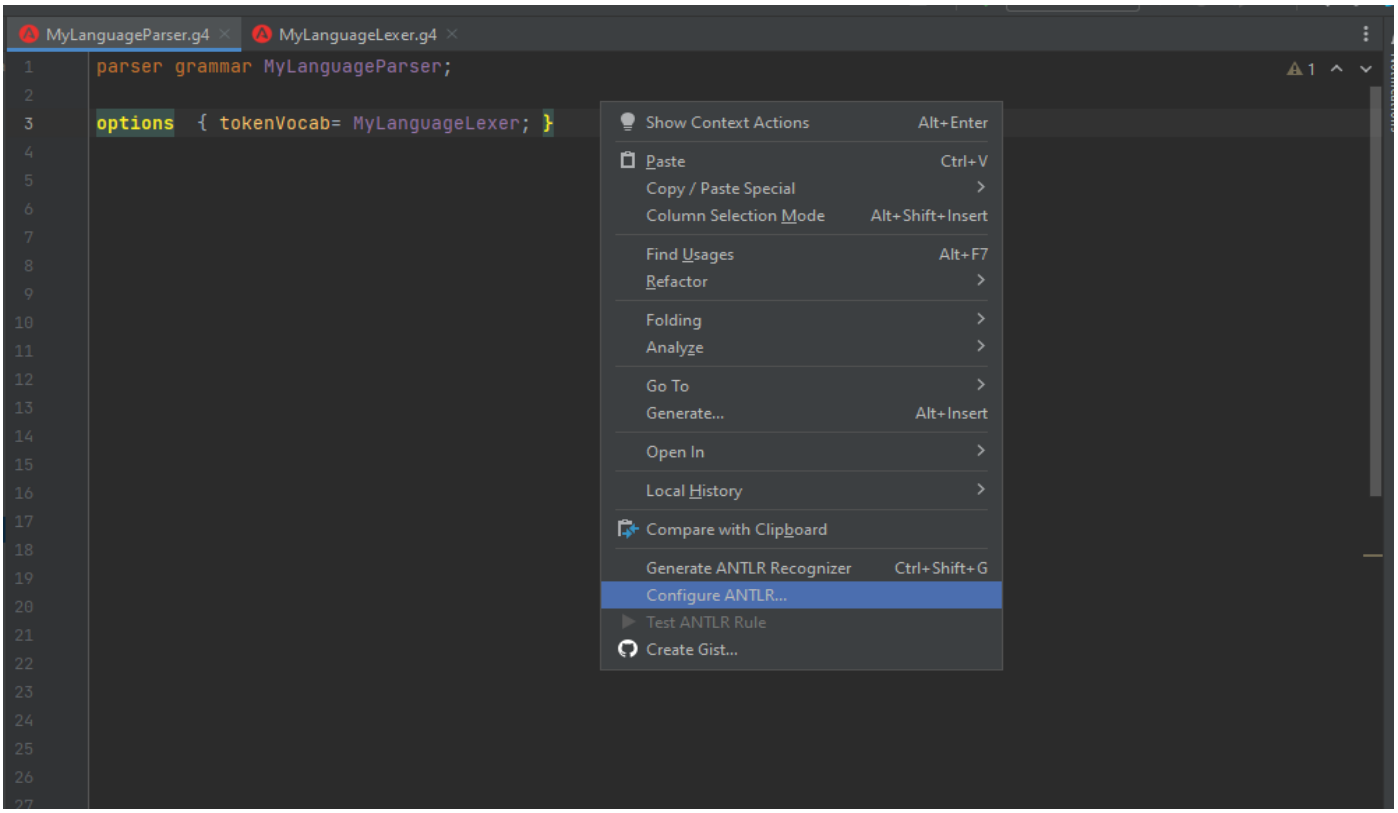
MyLanguageParser.g4 | MyLanguageLexer.g4

```
lexer grammar MyLanguageLexer ;

NEW_PAGE : ' '* 'hi' ' '* '{' ' '* ->pushMode(PAGE);

mode PAGE;
CLOSE_PAGE : ' '* '}' ' '* ->popMode;
NAME: ' '* 'name' ' '* ;
NAV :  ' '* 'add nav'  ' '*;
FOOTER :' '* 'add footer' ' '* ;
OPEN_BRACKET : ' '*'(' ' '* ->pushMode(ATTRIBUTES);
SEIMIE_QO : ' '* ';' ' '*;

mode ATTRIBUTES;
CLOSE_BRACKET:' '*')' ' '*->popMode;
QOUT: ' '* '\'' ' '*;
STRING: ('a'..'z' ' '*)+ ;
```

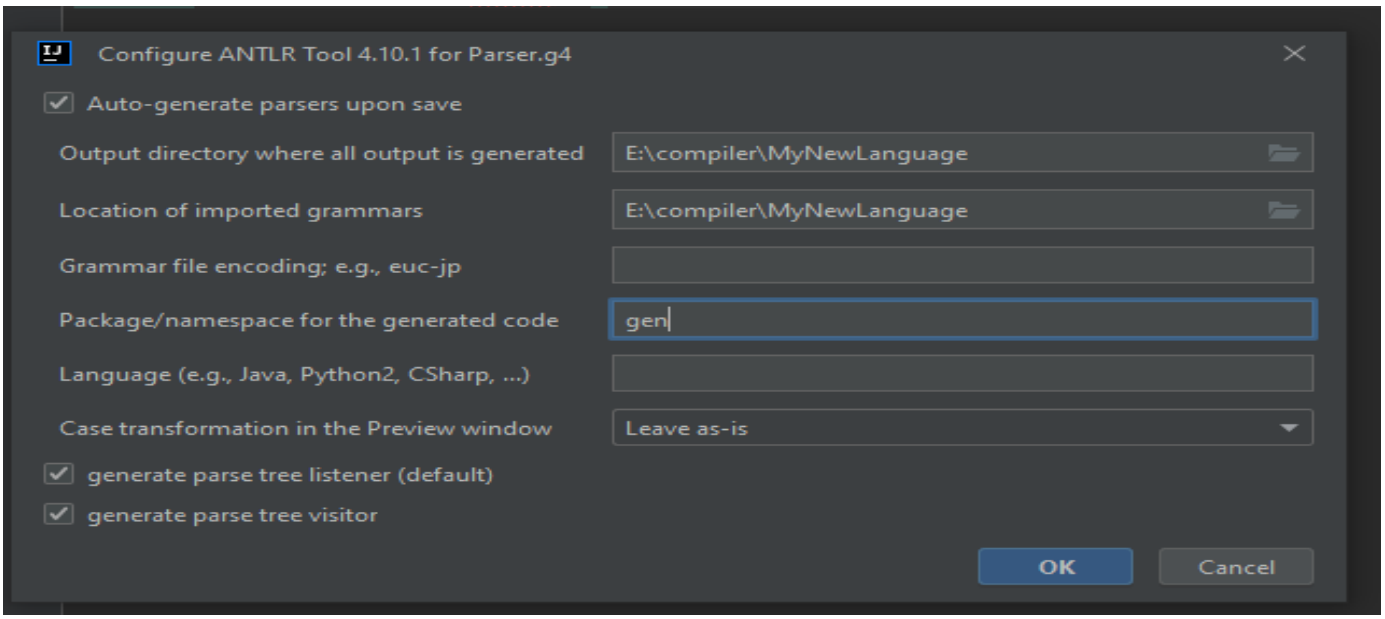
وضعت NEW_PAGE في default mode لأنها أول ما سيتم ترميزه .

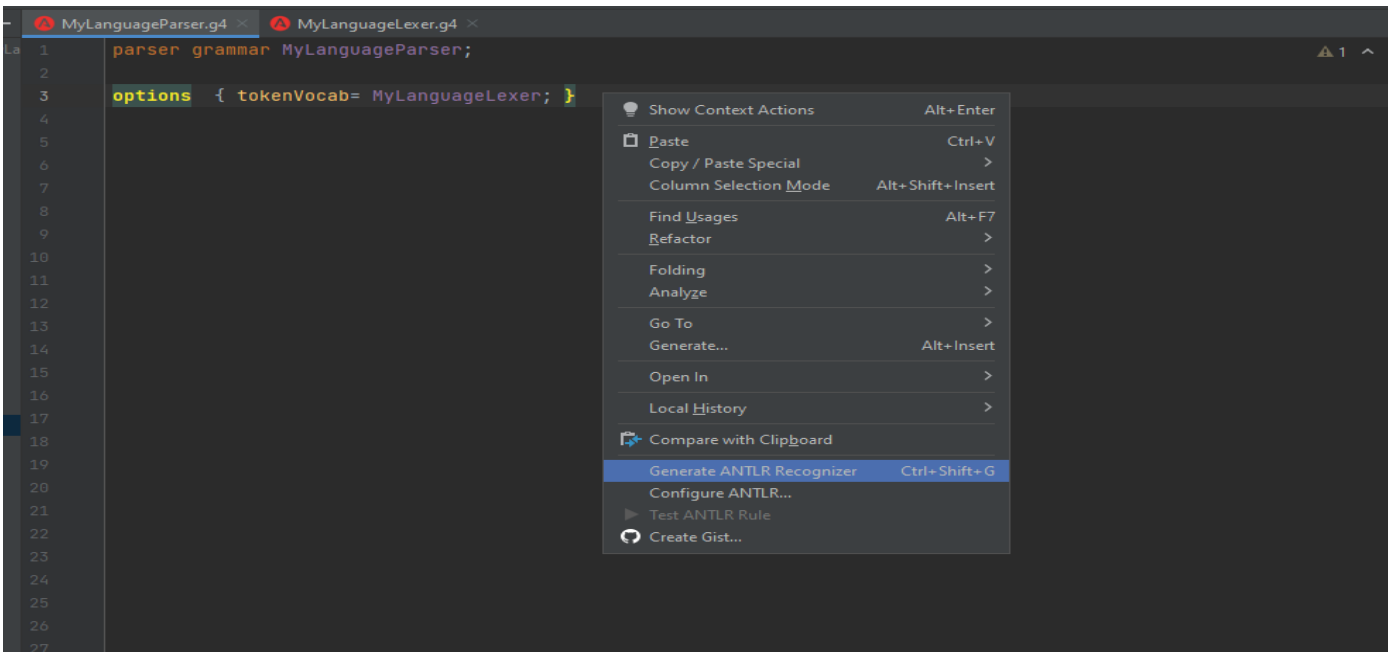
- بعدها قمت بعمل pushMode(PAGE) لفصل الرموز الخاصة بالصفحة عن باقي الرموز .
- قمت بترميز المفردات الخاصة بالصفحة ضمن ال mode PAGE
- قمت بعمل mode ATTRIBUTES لفصل الرموز عن ال mode PAGE

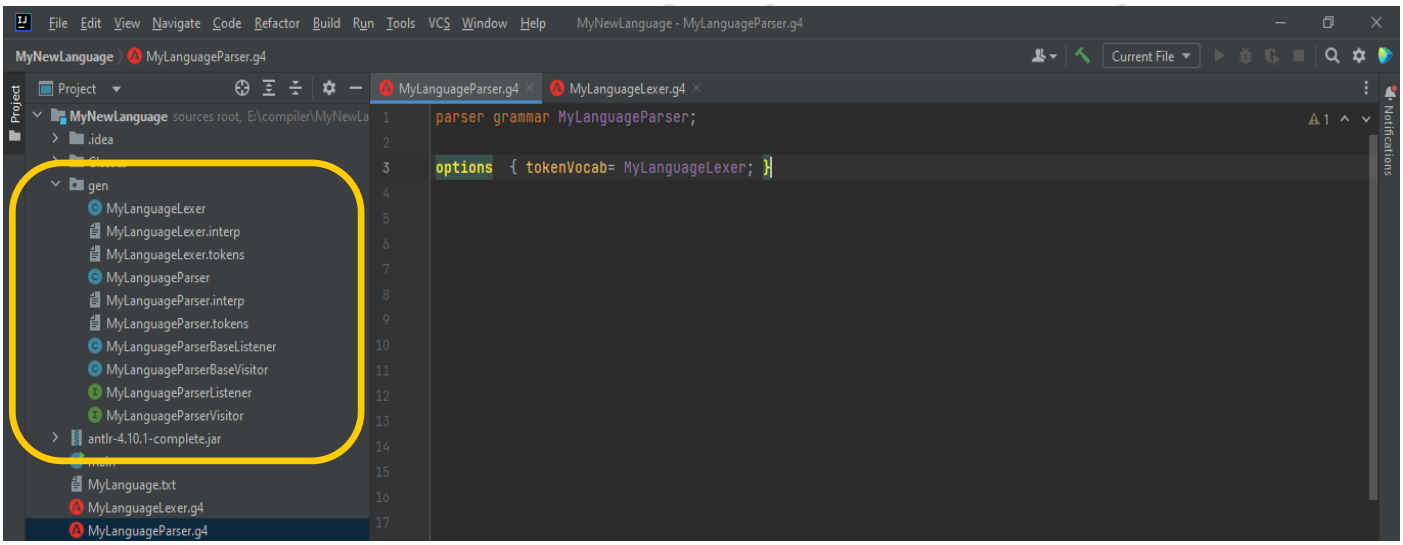
-الآن نقوم بتهيئة البرنامج لتوليد الرموز :



- الآن نقوم بتوليد الرموز







إضافة قواعد اللغة

القاعدة الأولى page

- تحتوي على { سلسلة ذات عنصر وحيد على الأقل من ال page_style }hi

القاعدة الثانية page_style

- تحتوي على القاعدة name أو navbar أو footer بالإضافة إلى الفاصلة المنقوطة

name; | nav; | footer ;

القاعدة الثالثة name

- تحتوي على كلمة name بالإضافة إلى القوس المفتوح و بعده الفارزة وبعد القاعدة page_name مع إمكانية عدم ورودها و بعدها فارزة يليها القوس المغلق .

القاعدة الرابعة page_name

- تحتوي على رمز STRING التي تعبر عن أي سلسلة من الأحرف من a ...z

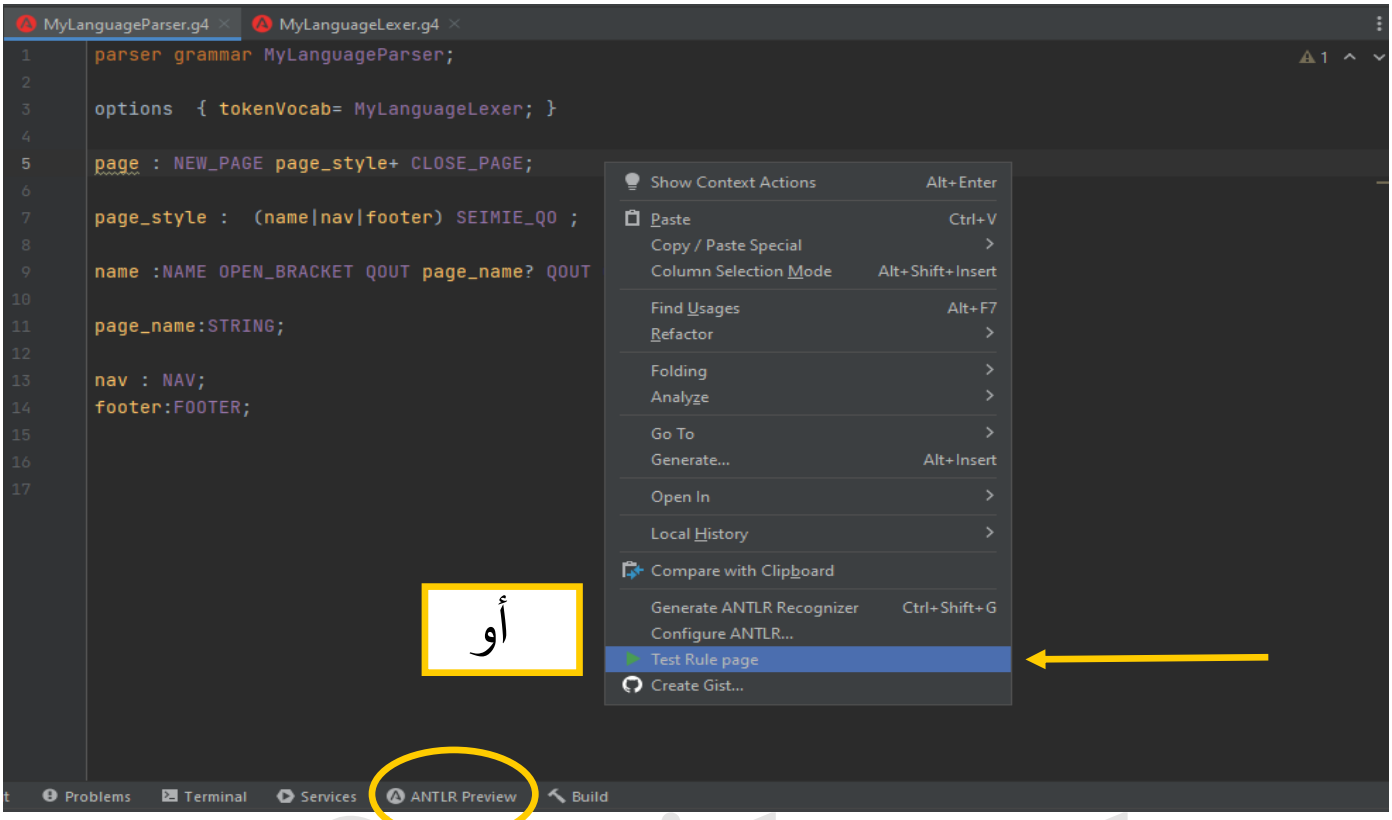
القاعدة الخامسة nav

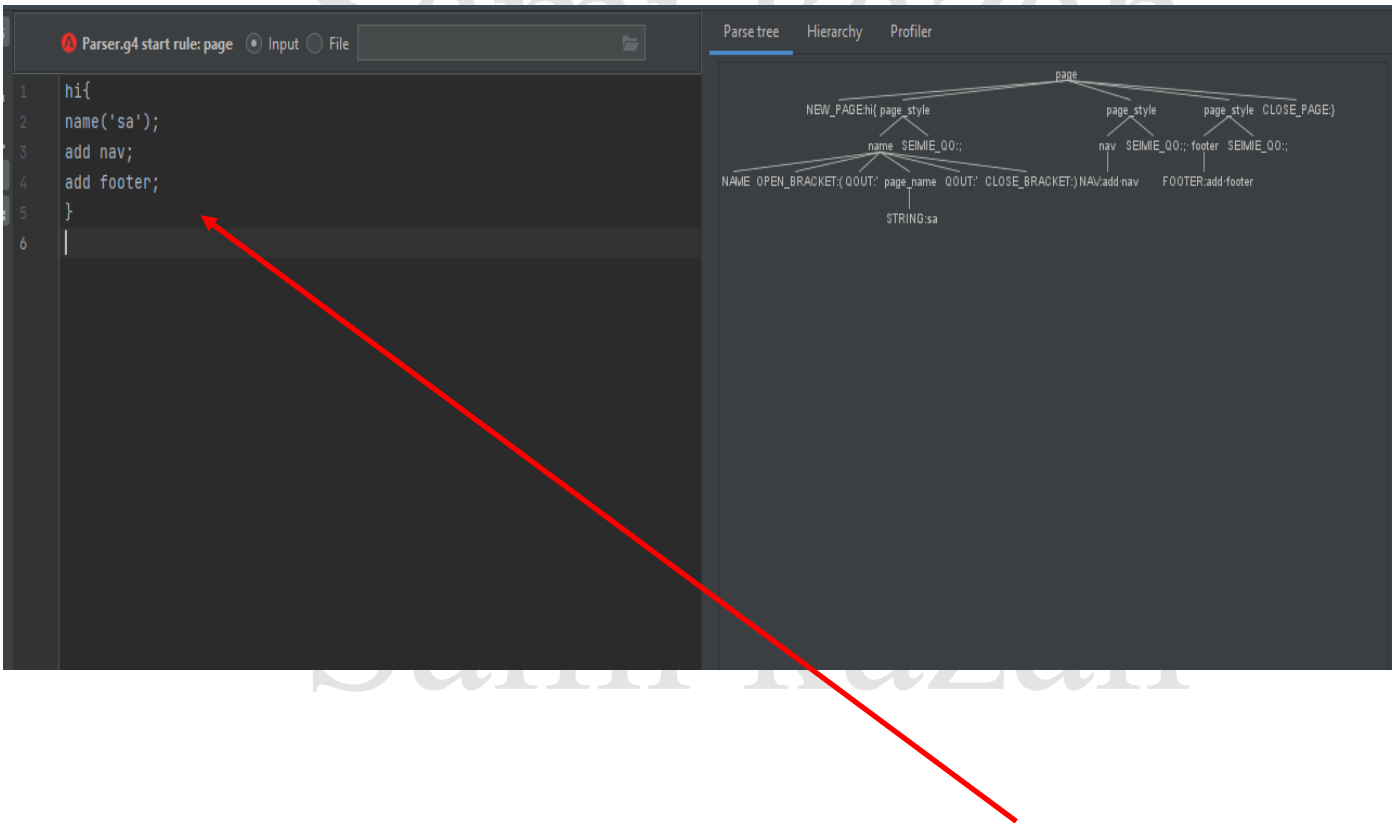
- تحتوي على رمز NAV الذي يعبر عن add nav

القاعدة السادسة footer

- تحتوي على رمز FOOTER الذي يعبر عن add footer

توليد شجرة ال Parse Tree





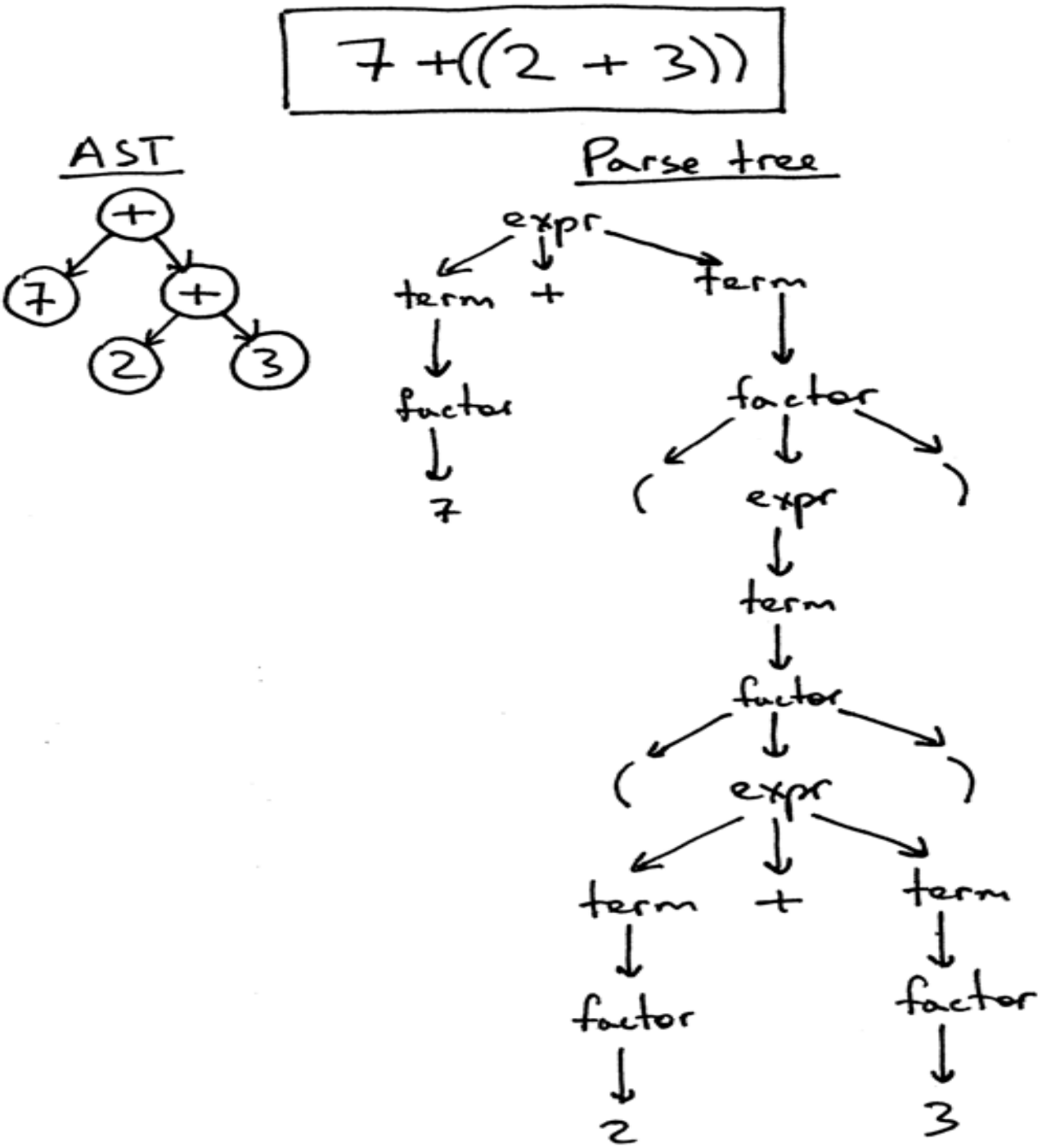
اكتب الكود المراد رسم شجرته

الفصل الثالث :
بناء المترجم (٢)

ما هي شجرة ال AST؟

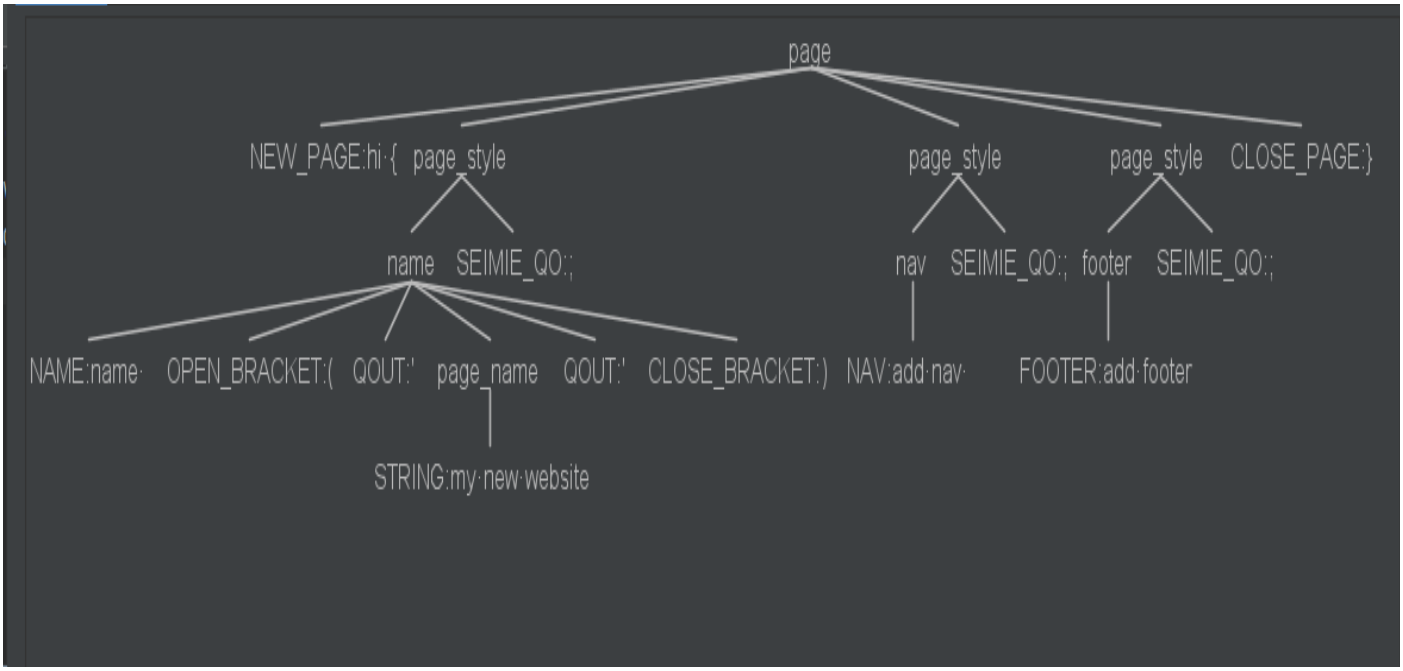
- **Abstract Syntax tree (شجرة البناء المجردة) :** هي بنية تستخدم للتمثيل جملة لغة البرمجة بشكل هرمي ، بحيث يتم استخدام هذه البنية لإنشاء ال Symbol table الذي يستخدم لاحقاً في عملية التحقق الدلالي Semantic check بالإضافة إلى توليد الكود Code Generation.

- تمثل ال AST جميع العناصر النحوية للغة البرمجة على غرار ال Parse Tree إلا أنها تركز على القواعد بدلاً عن العناصر مثل الأقواس و الفواصل و الفواصل المنقوطة .

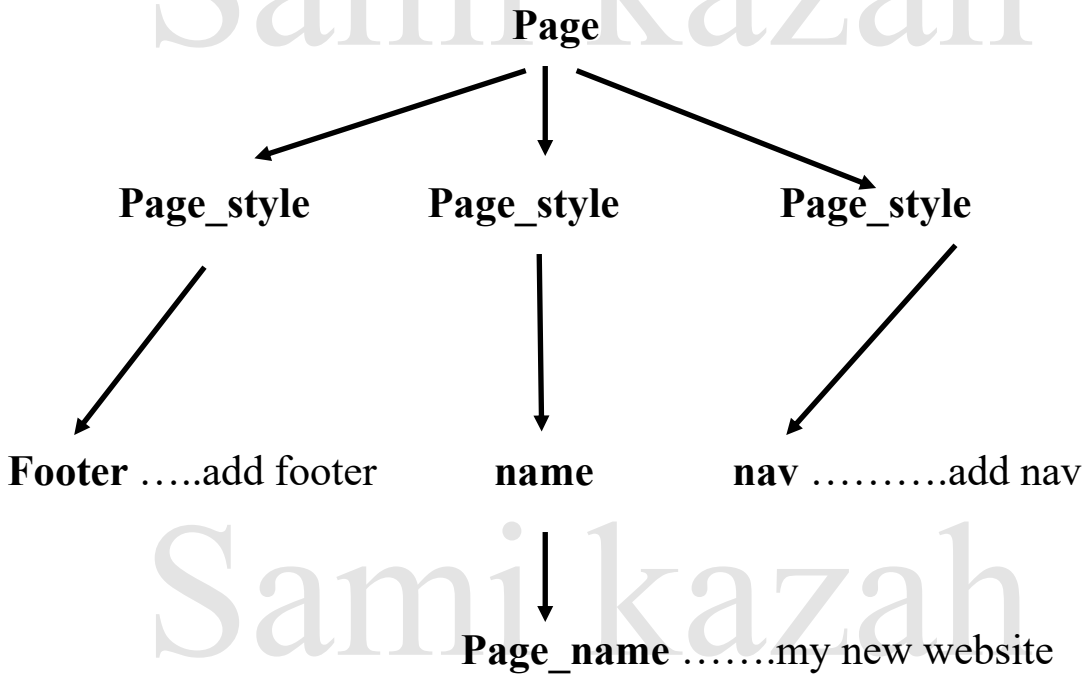


مقارنة بين ال AST و ال Parse Tree

في اللغة الجديدة التي قمت ببنائها في المرحلة السابقة كانت ال Parse tree الموافقة لها من الشكل :



بينما ال AST المقابلة لها ستكون من الشكل :



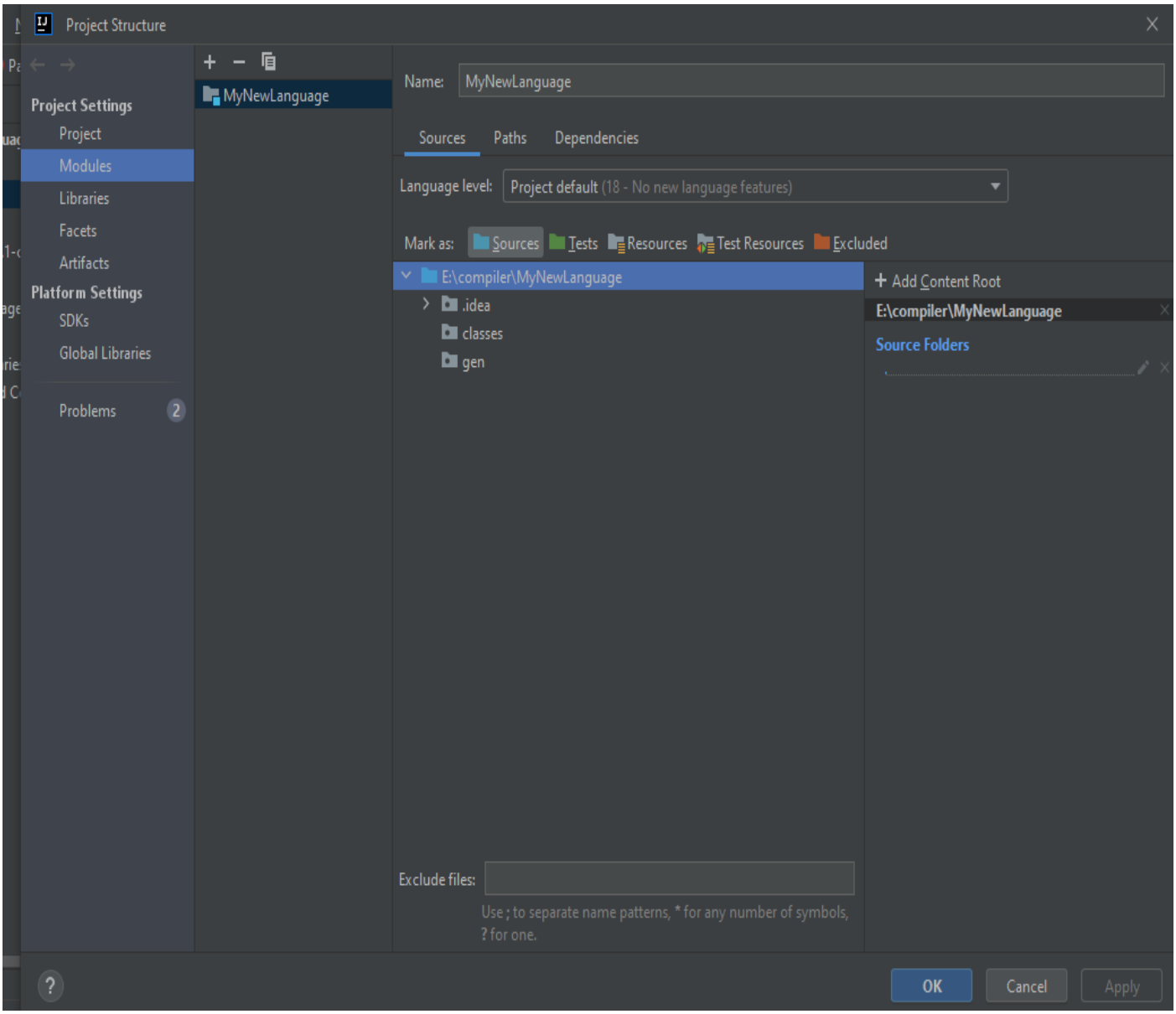
حيث تحتوي شجرة ال ASTعلى قواعد اللغة بالإضافة للرموز الرئيسية فقط (التي تمثل قيمة عددية أو تعطي معنى منطقي) .

بناء شجرة ال AST

بما أن ال AST تحتوي على قواعد اللغة ، و قواعد اللغة تحتوي على رموز tokens بالإضافة إلى قواعد اخرى فأفضل طريقة لتمثيل قاعدة هي باعتبارها كائن object و اعتبار الرمز token سلسلة String، و اعتبار السلسلة من القواعد سلسلة من الكائنات list of objects و هكذا .....

نقوم بتهيئة البيئة حسب لغة البرمجة التي نريد استخدامها (سنستخدم ال java) :

نضغط على file <<<<< project structure <<<<<<< modules <<<<<<< sources

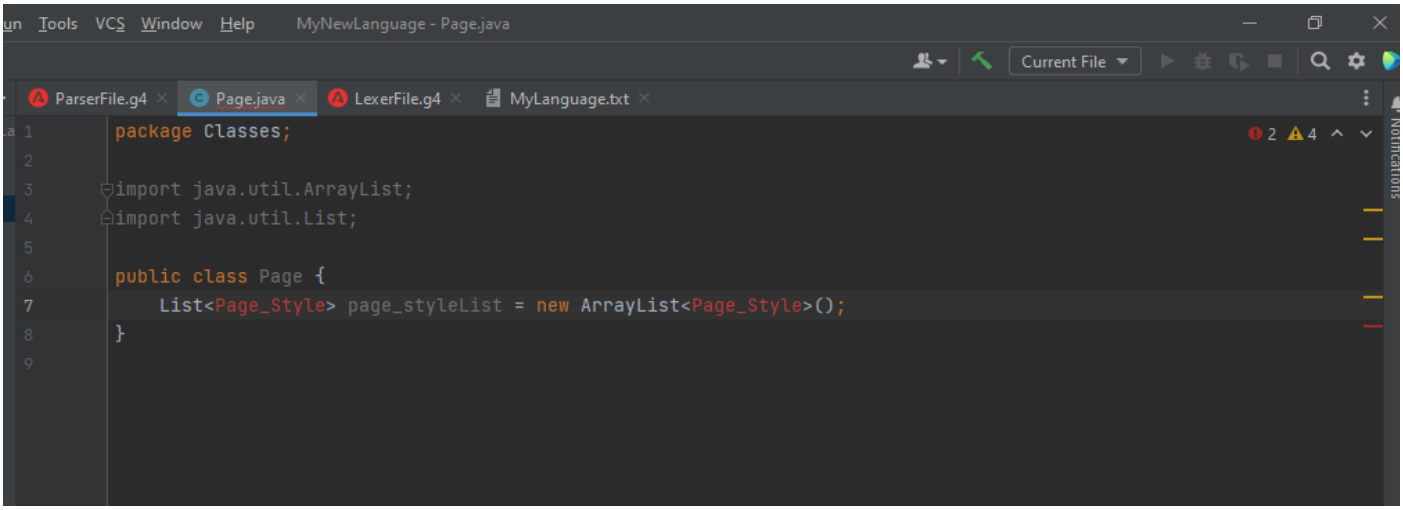


حيث نضغط على مجلد المشروع ثم نضغط على Sources

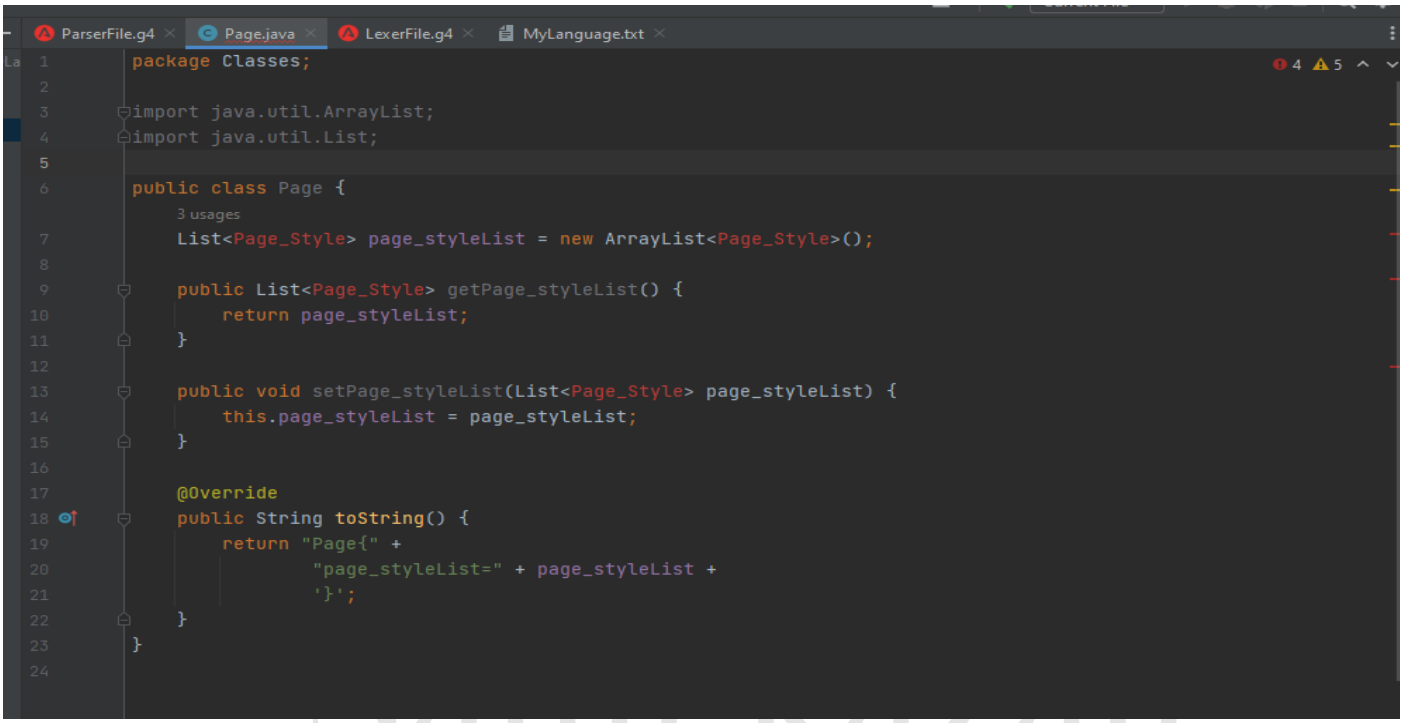
إنشاء صفوف القواعد :

بداية نقوم بإنشاء مجلد نسميه CLASSES سنضع داخله صفوف القواعد التي سنقوم بإنشائها .

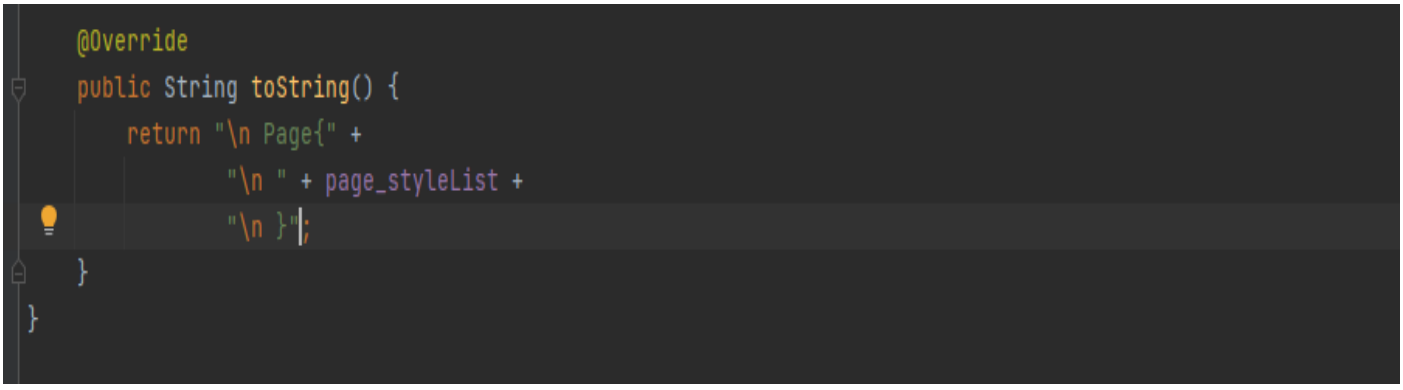
نقوم بإنشاء الصف Page:



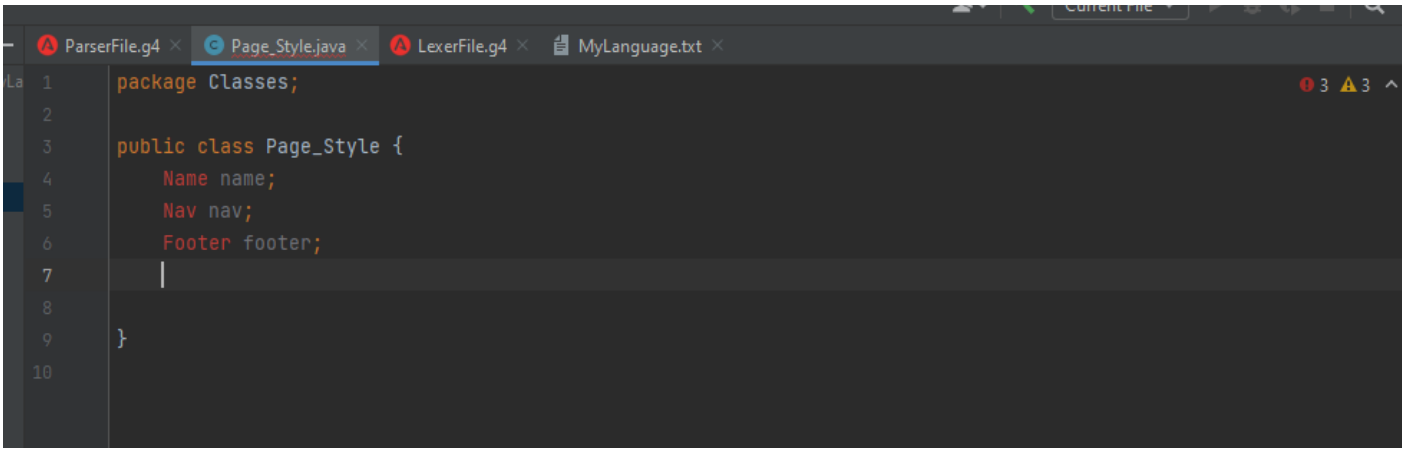
نضغط alt + insert و نضيف getter and setters ثم نضغط مرة اخرى ونضيف toString



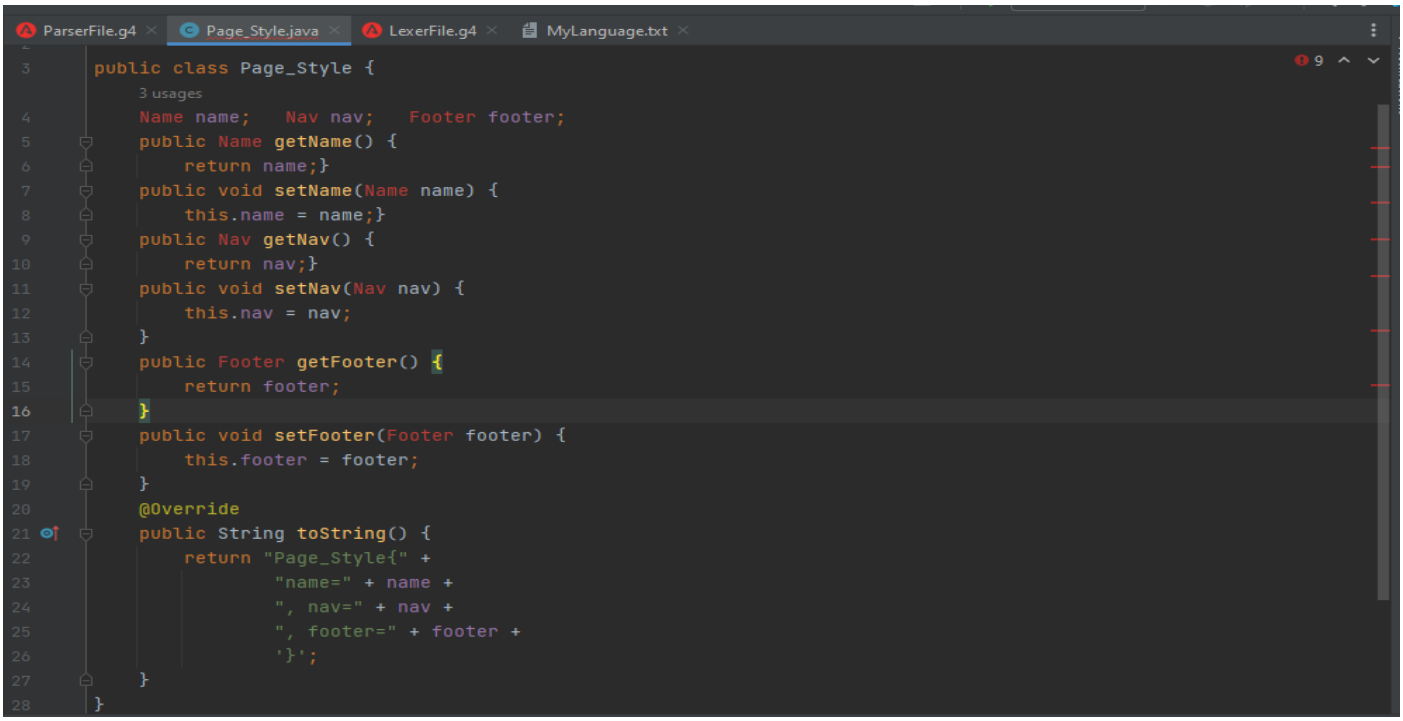
نعدل قليلا على تابع ال toString لتحسين شكل الطباعة



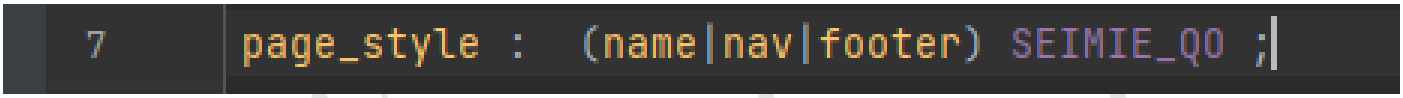
نقوم بإنشاء الصف Page_Style



نضغط alt + insert و نضيف getter and setters ثم نضغط مرة اخرى ونضيف toString



انتبه إن القاعدة page_style تحتوي على قاعدة واحدة فقط قد تكون name / nav / footer

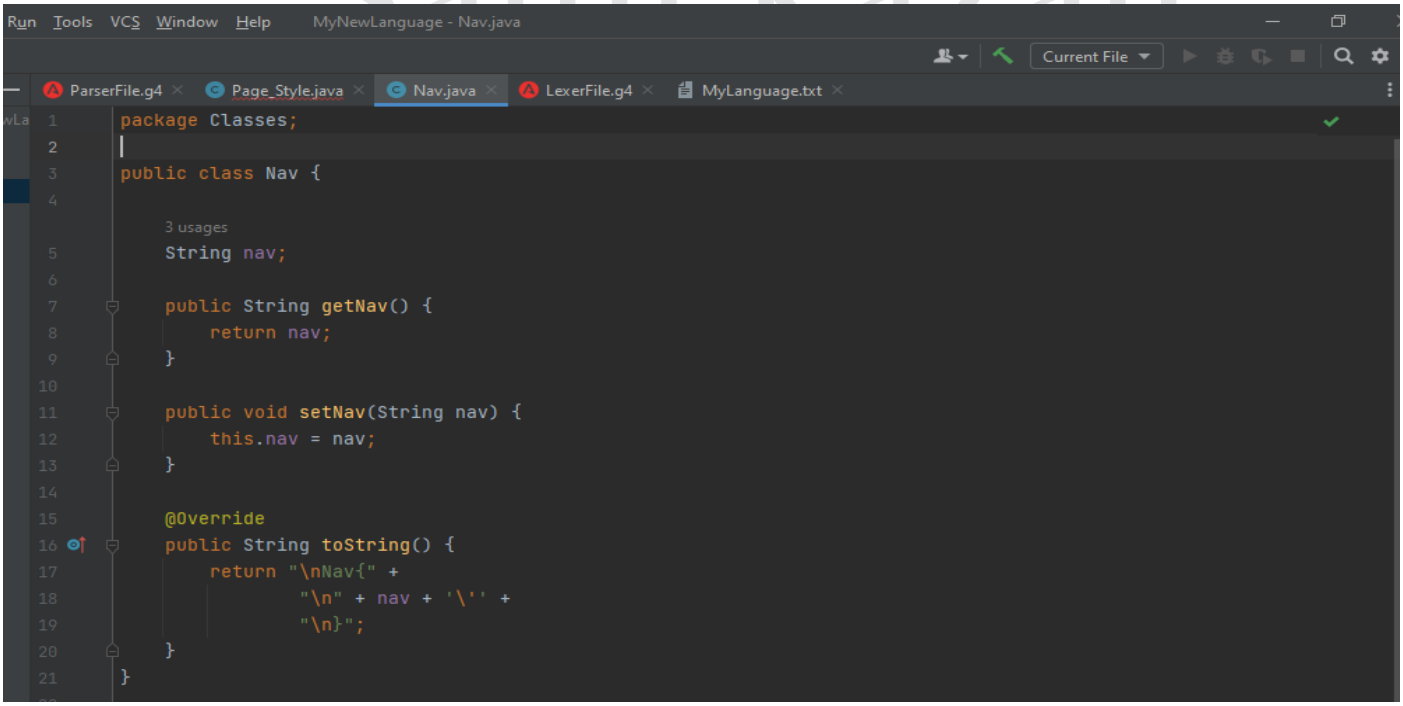


لذلك ستكون قيمة القاعدتين الباقيتين null فلتتجنب الأخطاء يجب وضع شروط للتحقق من وجود القيمة في التابع toString

وفق التالي :

```
@Override
public String toString() {
        if(name!=null)
        return "\n Page_Style{" +
                name +
               "\n}";
        else if(nav !=null)
        return "\nPage_Style{" +
                nav +
                "\n}";
        return "\nPage_Style{" +
                footer +
                "\n}";
}
```

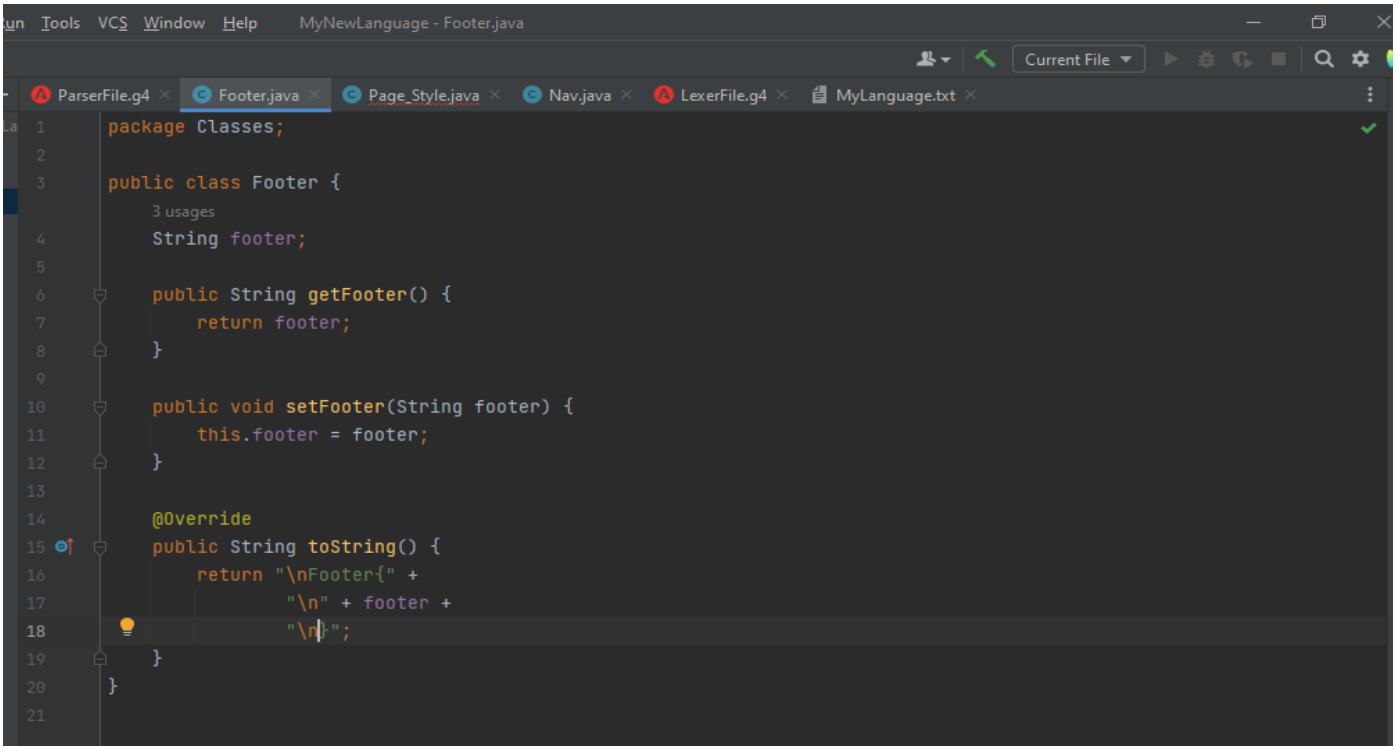
- الآن نقوم بإنشاء الصف Nav



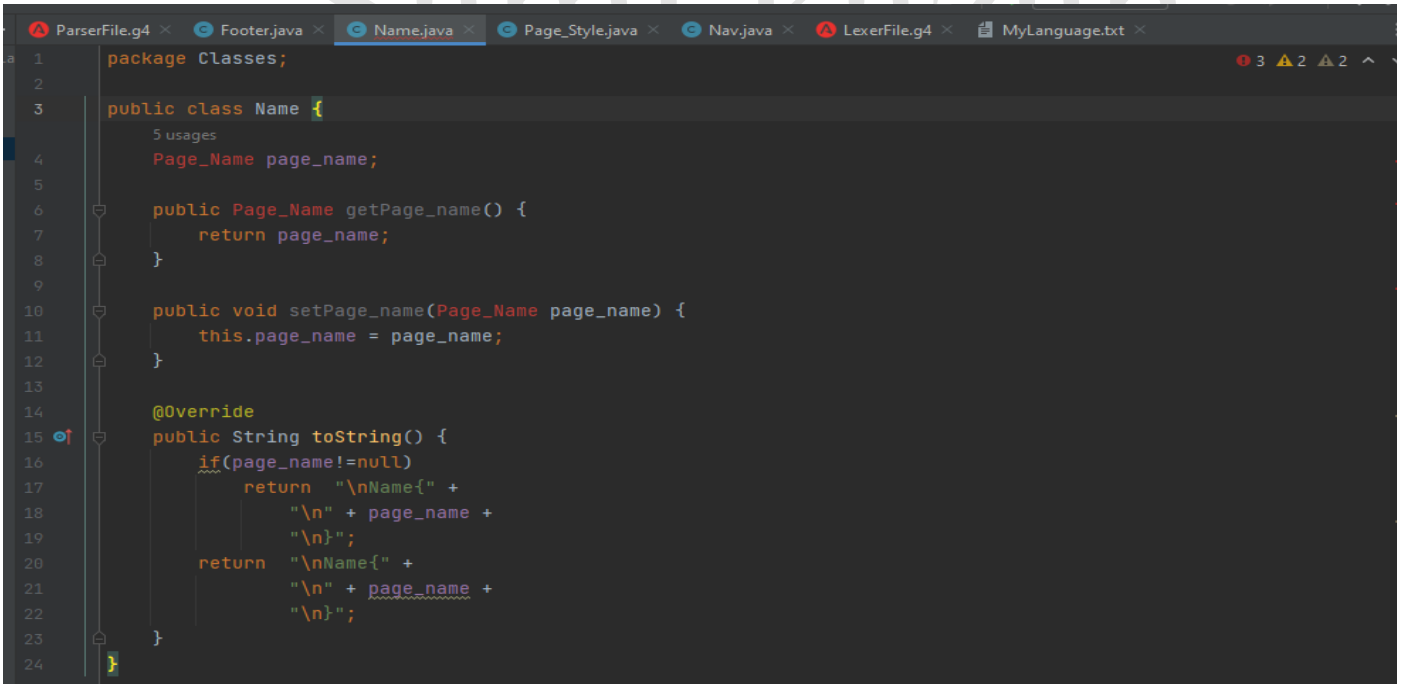
انتبه إن القاعدة nav تحتوي فقط على token لذلك تم تعريف الرمز على أنه String

```
nav : NAV;
```

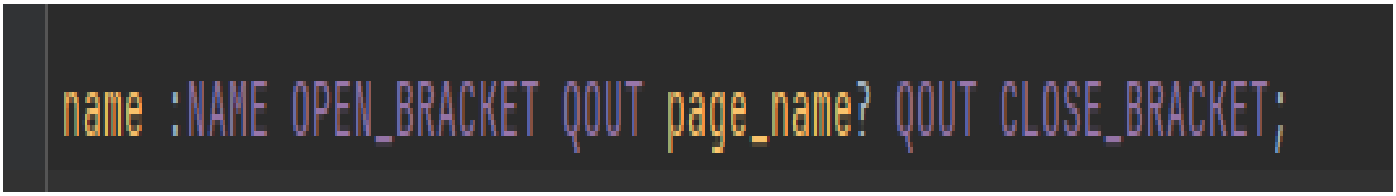
- الآن نقوم بإنشاء الصف Footer



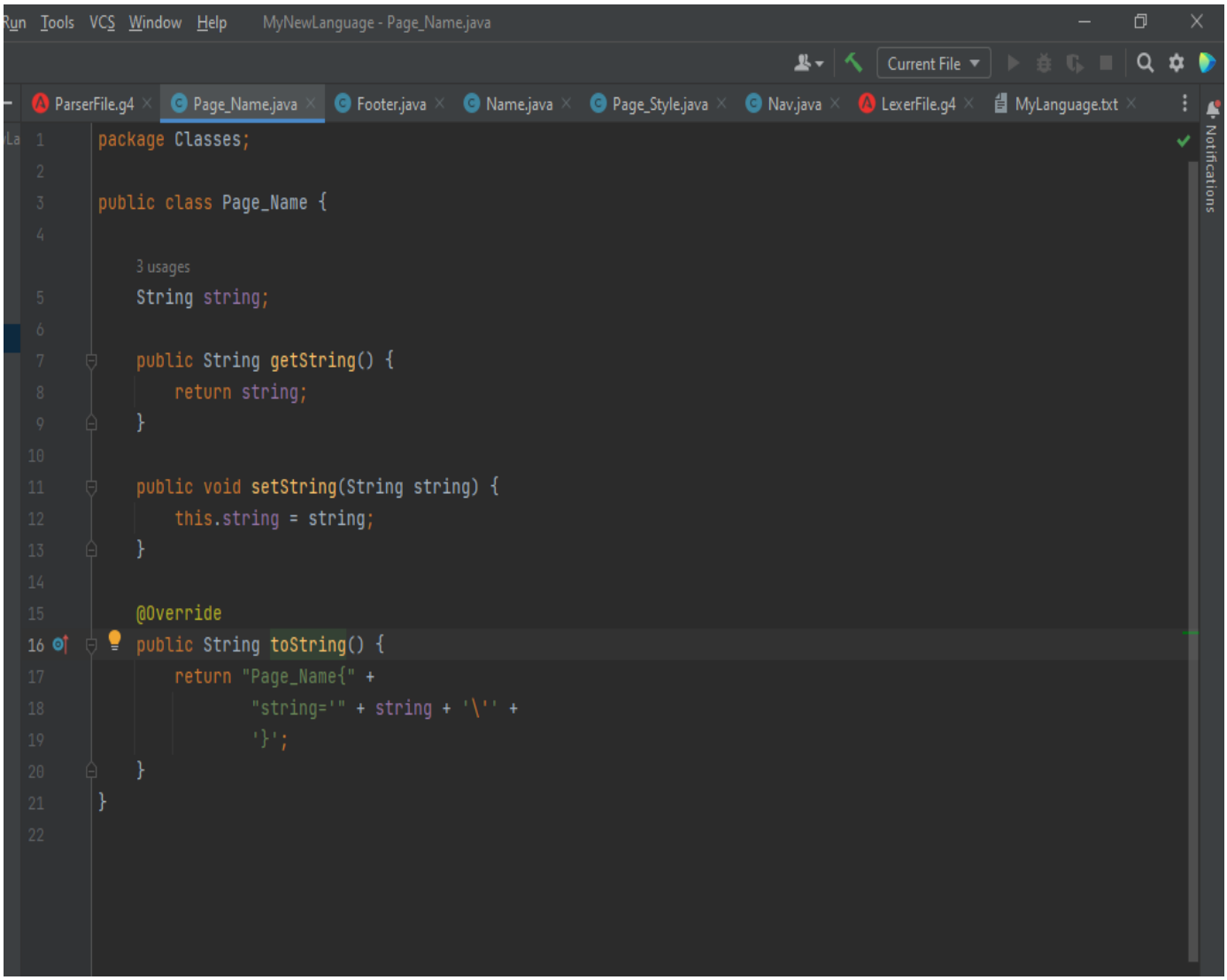
- الآن نقوم بإنشاء الصف Name



انتبه إلى أن القاعدة page_name قد لا تكون موجودة ضمن القاعدة name



- الآن نقوم بإنشاء صف Page_Name



بعد انتهائنا من إنشاء الصفوف نقوم بإنشاء صف main ضمن مجلد المشروع .

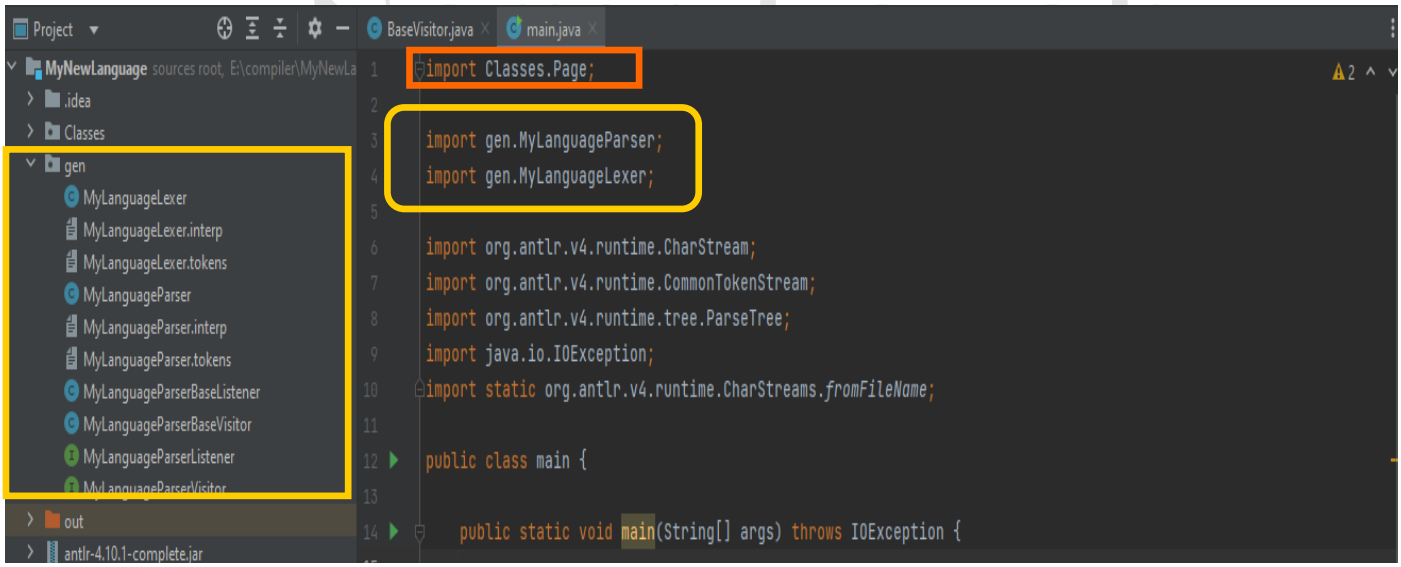
```
public class main {

    public static void main(String[] args) {

    }

    }
```

بعدها نذهب إلى ملف MyLanguageParser.g4 و نقوم بعمل generate للقواعد .

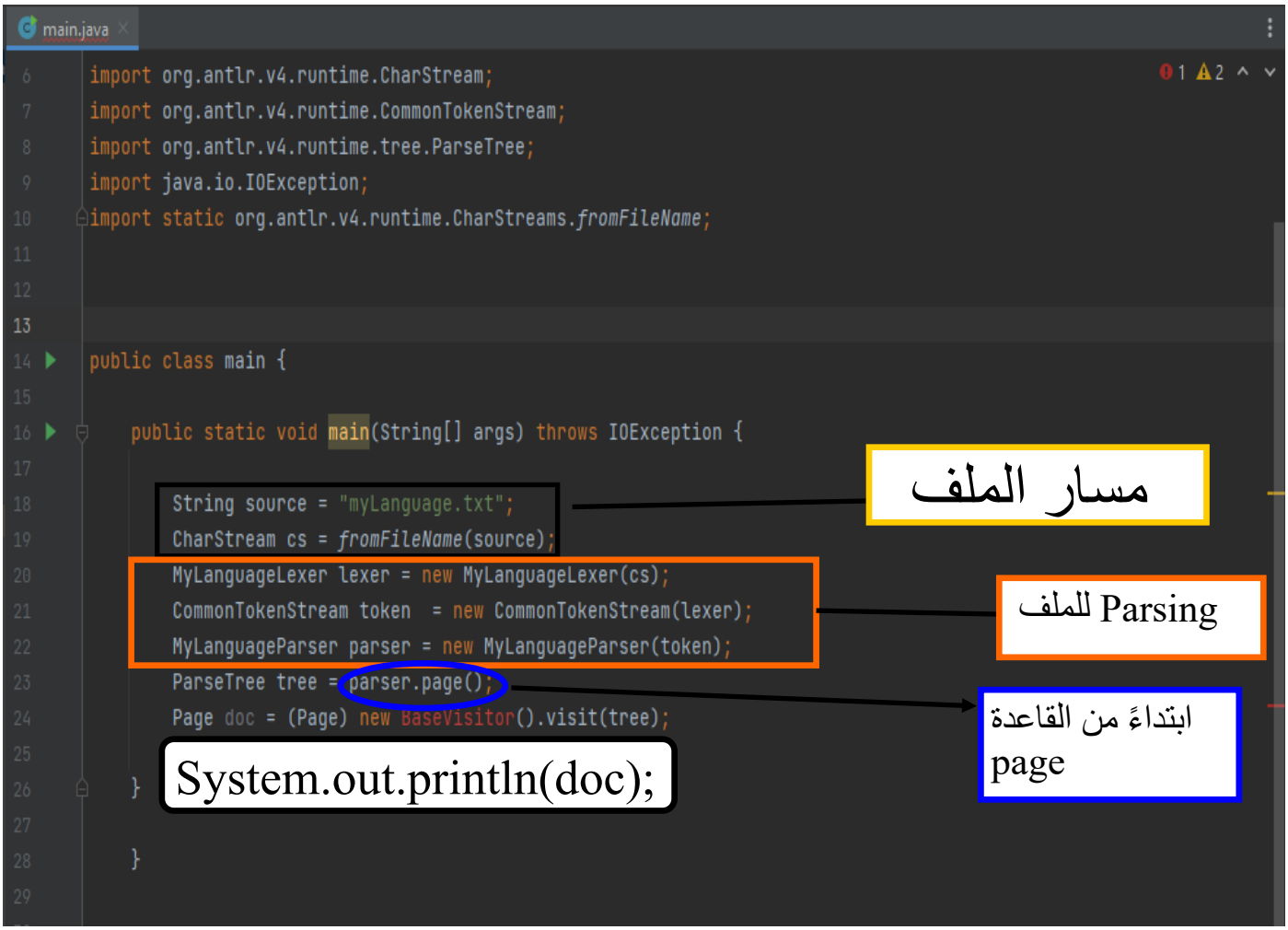
ملاحظة : عند القيام بإجراء أي تعديل على ملف ال Parser أو على ملف ال Lexer يجب عندها القيام بعمل generate للرموز و القواعد من ملف ال Parser

نقوم بعمل Import للصفوف ال MylanguageLexer , Mylanguage Parser التي قام ال Antlr بتوليدها تلقائياً عندما قمنا بعملية ال generate في ملف Mylanguage Parser.g4

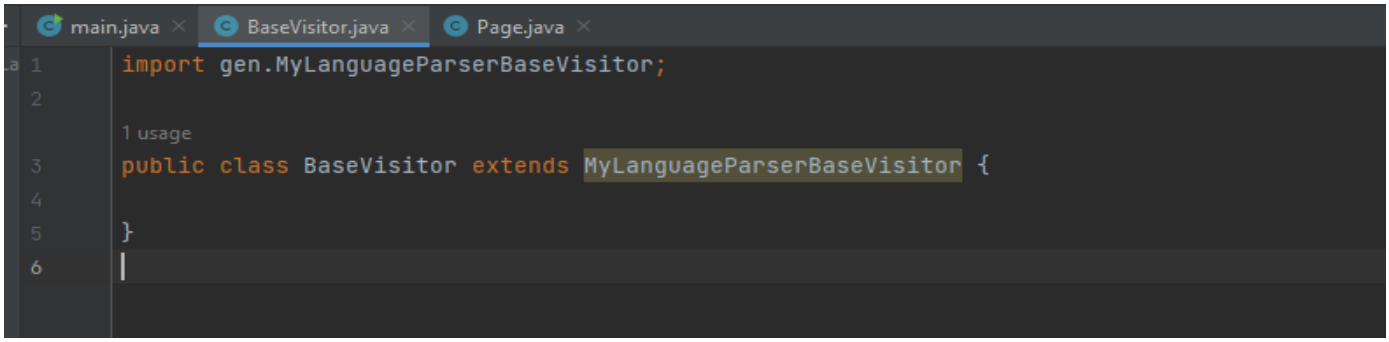
كما نقوم ايضاً بعمل import لصف القاعدة الأولى في لغتنا Page



بعد ذلك نقوم بإنشاء ملف MyLanguage.txt نكتب فيه الكود الخاص بلغتنا .

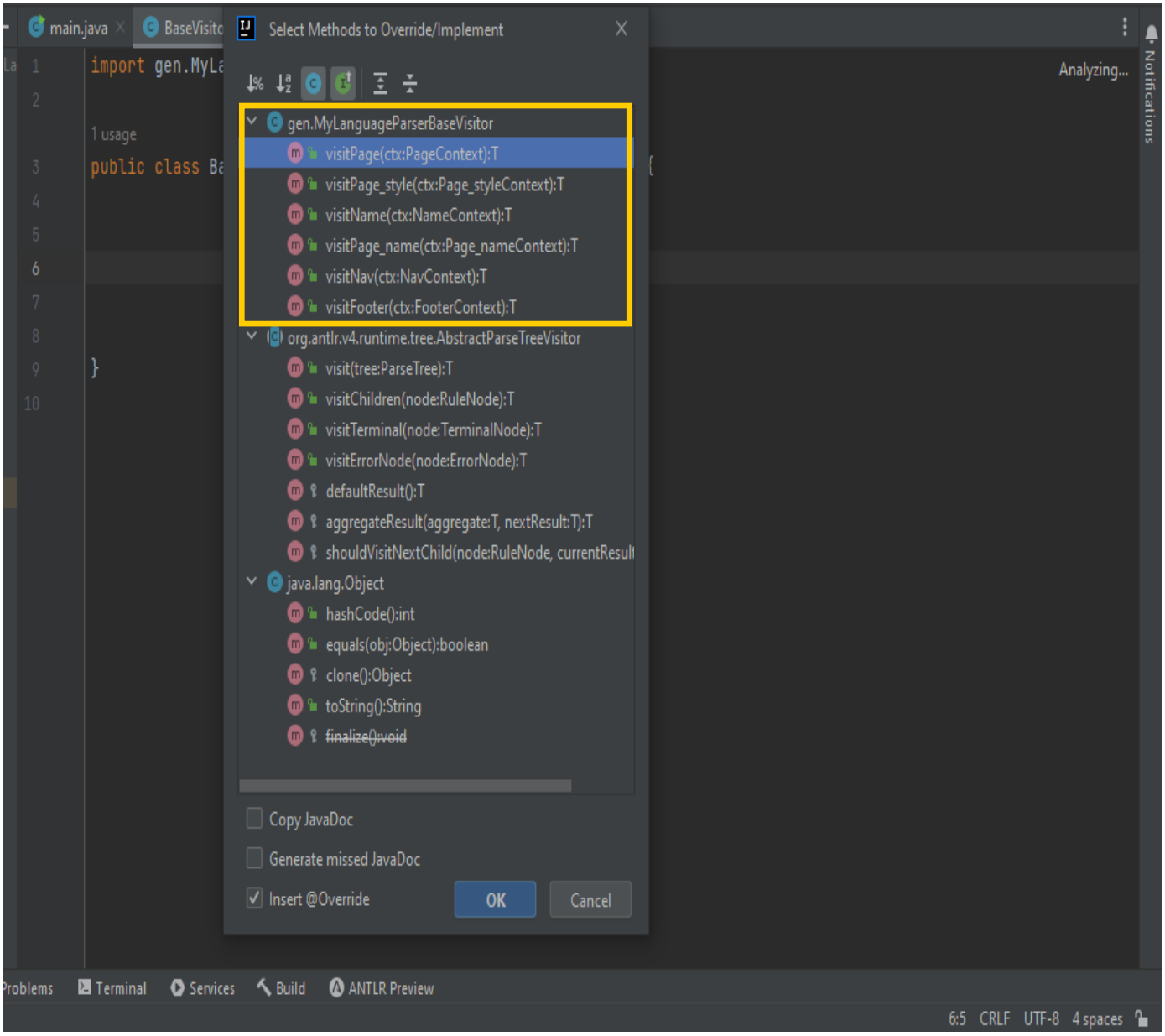


الان نقوم بإنشاء class جديد نسميه ال BaseVisitor ثم نقوم بعمل extends للصف MyLanguageParserBaseVisitor الذي تولد تلقائيا بعملية ال generate



حيث أن هذا الصف سيرث visit لجميع قواعد اللغة من MyLanguageParserBaseVisitor

بمجرد أن تضغط على CTRL + O ستظهر لك ال metods



- تكمن فائدة ال visit بأنه عند المرور بقاعدة ما نستطيع تخزين قيم ال tokens ضمن Symbol Table ثم نستخدمها فيما بعد في عملية التحقق المنطقي و تولد الكود .

مثال توضيحي : عند كتابة hi{} في ملف MyLanguage.txt نجد إن hi{} واقعة ضمن القاعدة page في ملف ال Parser فإذا احتجنا لتخزين كلمة hi للتأكد من ورودها ، أو لاستخدامها لاحقاً في عملية توليد الكود ، نقوم بعمل override للتابع visitpage الذي نستطيع من خلاله الحصول على قيم ال Token الموجودة ضمن القاعدة page ومنها الرمز المعبر عن ال hi و الذي هو في مشروعنا NEW_PAGE بكتابة ctx.NEW_PAGE كما يمكننا تخزنيه على شكل string بكتابة

ctx.NEW_PAGE.getText();

```
public class BaseVisitor extends MyLanguageParserBaseVisitor {

    1 usage
    @Override
    public Page visitPage(MyLanguageParser.PageContext ctx) {
        Page page =new Page();
        String hi = ctx.NEW_PAGE().getText();
        return page;
    }
}
```

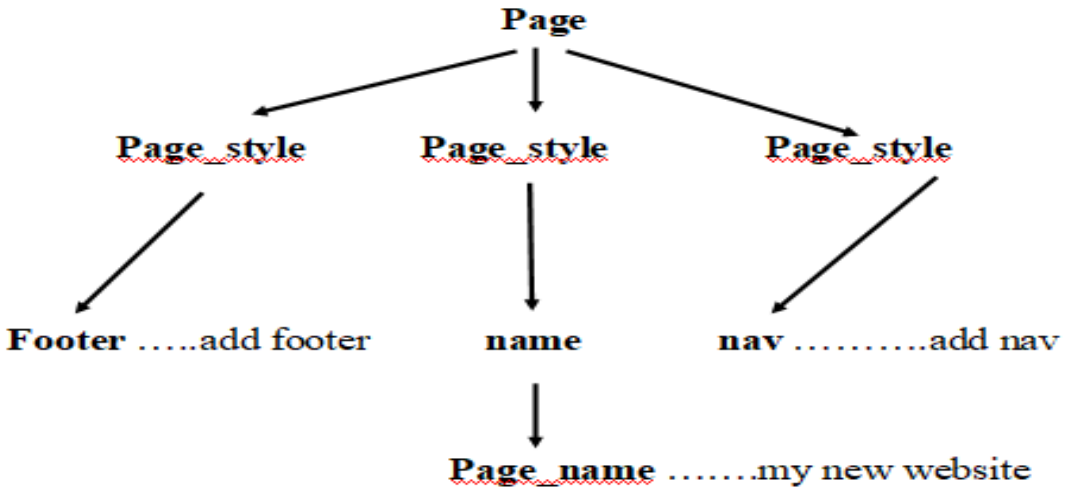
لاحظ بأن ctx تمكنك من الوصول لجميع ال tokens بالإضافة للقواعد الاخرى الموجودة ضمن القاعدة page

```
public class BaseVisitor extends MyLanguageParserBaseVisitor {

    1 usage
    @Override
    public Page visitPage(MyLanguageParser.PageContext ctx) {
        Page page =new Page();
        String hi = ctx.NEW_PAGE().getText();
        ctx.
    }
}
```

CLOSE_PAGE() TerminalNode
NEW_PAGE() TerminalNode
exception RecognitionException
enterRule(ParseTreeListener listener) void
page_style() List<Page_styleContext>
children List<ParseTree>
invokingState int
parent RuleContext
start Token
stop Token
exitRule(ParseTreeListener listener) void

Ctrl+Down and Ctrl+Up will move caret down and up in the editor Next Tip

بالعودة إلى مشروعنا فنحن نحتاج لعمل visit للقواعد التالية و تخزين ال tokens المعرفة أدناه:



- نبدأ من ال page نضغط CTRL + O و نقوم بعمل OverRide للتابع VisitPage
- بما أننا قمنا بتعريف الصف Page لتخزين ArrayList من ال page_style نجعل ال تابع visitPage من النوع Page ليرد لنا كائن Page .
- بما أننا نستطيع الوصول إلى pageStyle باستخدام visitPage_styleو نحتاج لتخزين جميع ال page_style الموجودة ضمن القاعدة page نقوم بعمل حلقة تقوم عند كل عملية مرور بال pageStyle بعملvisitPage _style التي سترد لنا فيما بعد كائن page Style نقوم بإضافته إلى ال arrayList المعرفة بالصف page.

```
1 usage
public class BaseVisitor extends MyLanguageParserBaseVisitor {


    1 usage
    @Override
    public Page visitPage(MyLanguageParser.PageContext ctx) {
        Page page =new Page();
        for (int i = 0; i < ctx.page_style().size(); i++) {
            if(ctx.page_style(i)!=null)
            {
                page.getPage_styleList().add(visitPage_style(ctx.page_style(i)));
            }
        }
        return page;
    }


}
```

- نضغط CTRL + O و نقوم بعمل OverRide للتابع visitPage_style

- نقوم بتغيير نوع القيمة التي يردها التابع إلى كائن من نوع Page_Style ثم نقوم بإنشاء كائن page_style و عمل return له .

- ننتبه إلى أن القاعدة page_style ترد قيمة واحدة فقط لأحد القواعد الثلاث name/nav/footer


```
2 usages
@Override
public Page_Style visitPage_style(MyLanguageParser.Page_styleContext ctx) {
    Page_Style page_style = new Page_Style();
    if(ctx.name()!=null)
    page_style.setName(visitName(ctx.name()));
    if(ctx.nav()!=null)
    page_style.setNav(visitNav(ctx.nav()));
    if(ctx.footer()!=null)
    page_style.setFooter(visitFooter(ctx.footer()));
    return page_style;
}

}
```


- نضغط CTRL + O و نقوم بعمل OverRide للتابع visitName

- نقوم بتغيير نوع القيمة التي يردها التابع إلى كائن من نوع Name ثم نقوم بإنشاء كائن name و عمل return له .

- ننتبه إلى أن القاعدة name ترد قيمة واحدة فقط و هي القاعدة page_name مع احتمال عدم ورودها

```
2 usages
@Override
public Name visitName(MyLanguageParser.NameContext ctx) {
    Name name =new Name();
    if(ctx.page_name()!=null)
        name.setPage_name(visitPage_name(ctx.page_name()));
    return name;
}
}
```

- نضغط CTRL + O و نقوم بعمل OverRide للتابع visitPage_name

- نقوم بتغيير نوع القيمة التي يردها التابع إلى كائن من نوع Page_Name ثم نقوم بإنشاء كائن page_name و عمل return له .

- ننتبه إلى أن القاعدة page_name ترد قيمة واحدة فقط وهي رمز .

```
2 usages
@Override
public Page_Name visitPage_name(MyLanguageParser.Page_nameContext ctx) {
    Page_Name page_name =new Page_Name();
    page_name.setString(ctx.STRING().getText());
    return page_name;
}
}
```

- نضغط CTRL + O و نقوم بعمل OverRide للتابع visitNav

- نقوم بتغيير نوع القيمة التي يردها التابع إلى كائن من نوع NAV ثم نقوم بإنشاء كائن nav و عمل return له .

- ننتبه إلى أن القاعدة nav ترد قيمة واحدة فقط وهي رمز .

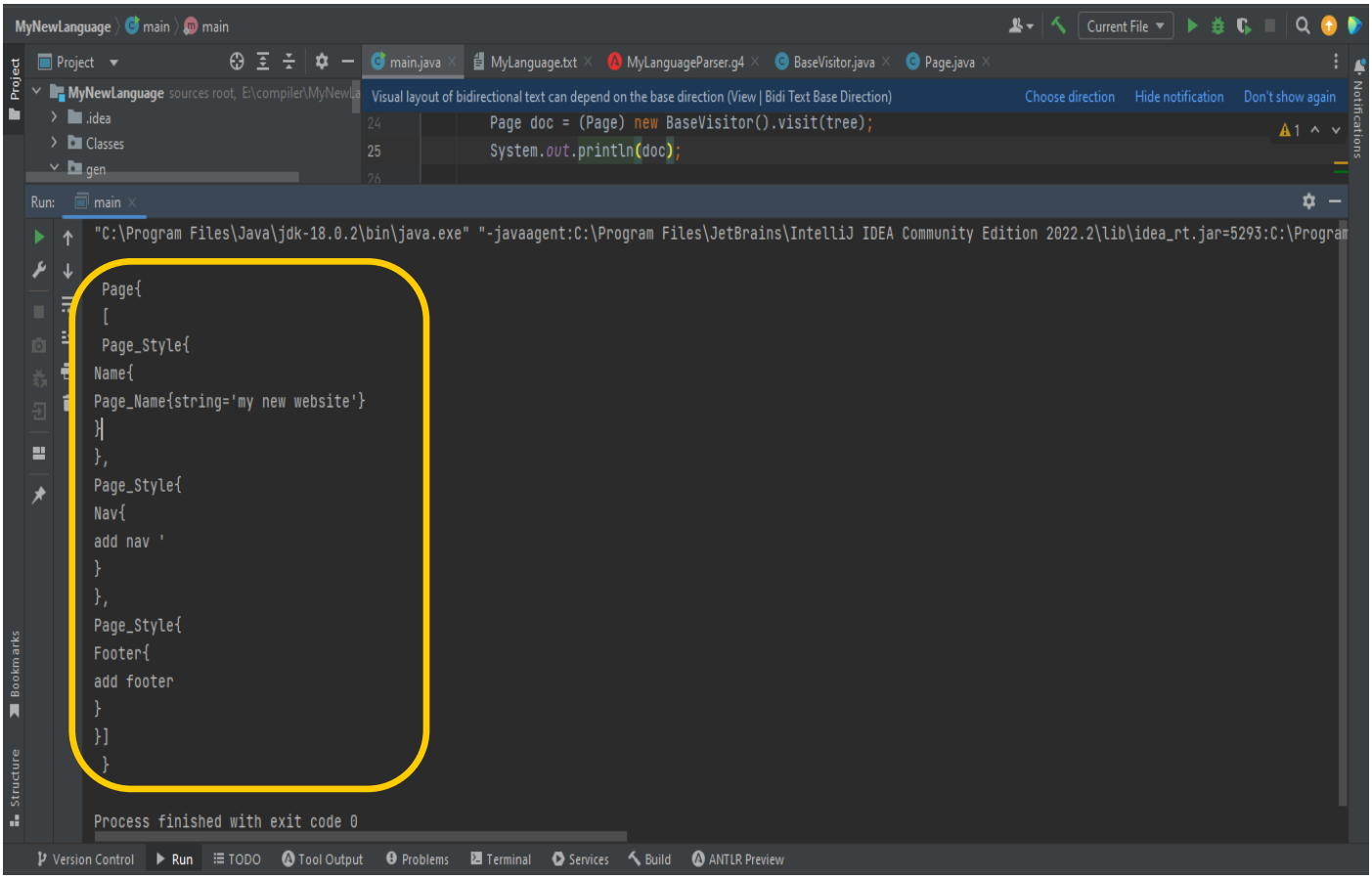
```
2 usages
@Override
public Nav visitNav(MyLanguageParser.NavContext ctx) {
    Nav nav = new Nav();
    nav.setNav(ctx.NAV().getText());
    return nav;
}
}
```

نكرر نفس الخطوات السابقة بالنسبة إلى footer

```
2 usages
@Override
public Footer visitFooter(MyLanguageParser.FooterContext ctx) {
    Footer footer =new Footer();
    footer.setFooter(ctx.FOOTER().getText());
    return footer;
}
```

توليد شجرة ال AST

نقوم بعمل run لملف ال Main



لتحسين شكل الخرج نعدل على توابع ال toString() ال الموجودة ضمن الصفوف التي قمنا بإنشائها سابقاً .

بعض الأخطاء الشائعة (٢)

- إهمال شروط التحقق من وجود القيمة

```
public class BaseVisitor extends MyLanguageParserBaseVisitor {


    1 usage
    @Override
    public Page visitPage(MyLanguageParser.PageContext ctx) {
        Page page =new Page();
        for (int i = 0; i < ctx.page_style().size(); i++) {
            if(ctx.page_style(i)!=null)
            {
                page.getPage_styleList().add(visitPage_style(ctx.page_style(i)));
            }
        }
        return page;
    }
```

- محاولة الوصول إلى قاعدة بدون عمل visit

```
{
    name.setPage_name(visitPage_name(ctx.page_name()));
    name.setPage_name(ctx.page_name());
}
```

- محاولة تخزين token بدون استخدام getText()

```
footer.setFooter(ctx.FOOTER().getText());
footer.setFooter(ctx.FOOTER());
```

- استخدام ال set بدلاً من get. Add لإضافة قيمة إلى ArrayList

```
page.getPage_styleList().add(visitPage_style(ctx.page_style(i)));
page.setPage_styleList().add(visitPage_style(ctx.page_style(i)));
```

الفصل الرابع:
بناء المترجم (٣)

ما هو ال Symbol Table ؟

هو بنية معطيات Data Structure لمعرفة دلالة المتغيرات ، كل رمز و معناه أو القيمة التي يحملها

يستخدم في عملية التحقق الدلالي Semantic Check و توليد الكود Code Generation

- في لغتنا التي قمنا بإنشائها بالمراحل السابقة سيكون جدول الرموز المتصور الخاص بها من الشكل :

Name	MyWebPageName / Home
Nav	Add nav
Footer	Add footer

- إن كل عنصر مخزن في جدول الرموز له أهمية في عملية توليد الكود و التحقق المنطقي .

- ما يتم تخزينه بال Symbol Table :

١ - المتغيرات و الثوابت

٢- التوابع و الإجرائيات

٣- الثوابت و السلاسل

٤- التسميات بلغة المصدر

- يمكن أن نبينه بالطريقة التي نراها مناسبة و التي قد تساعدنا لاحقاً بعملية توليد الكود و التحقق المنطقي

قد يكون :

١– List

٢– Linked List

٣– Hash Table

٤– Binary Search Tree

٥– Object

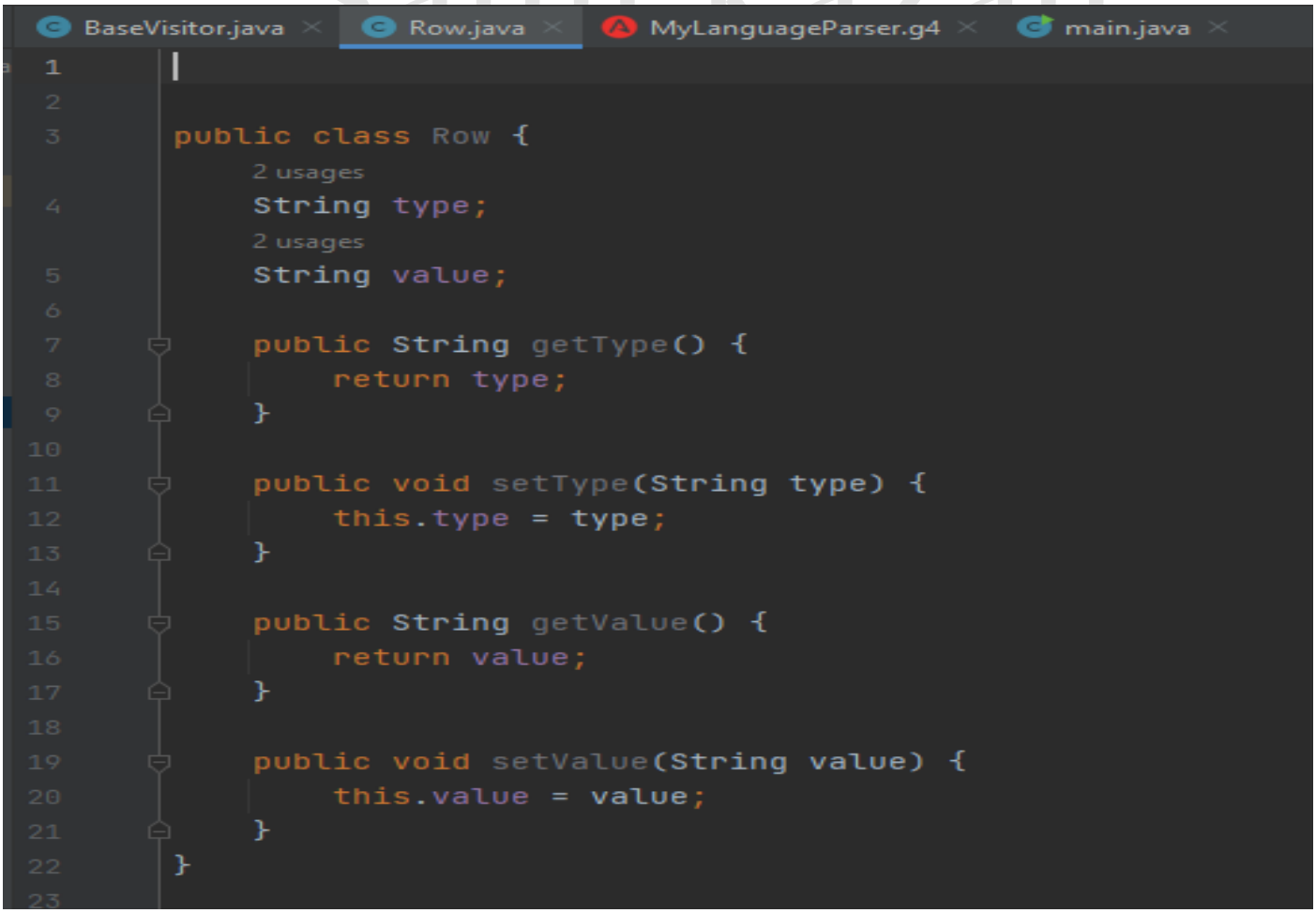
بناء ال Symbol Table

بما أن جدول الرموز المتصور من الشكل :

Name	MyWebPageName / Home
Nav	Add nav
Footer	Add footer

فيمكننا أن نعتبر أن الجدول هو Object يحتوي على List of rows حيث أن row هو أيضاً Object مؤلف من Type + Value .

- نقوم بإنشاء صف ال Row و عمل توابع ال get و ال set الخاصة به



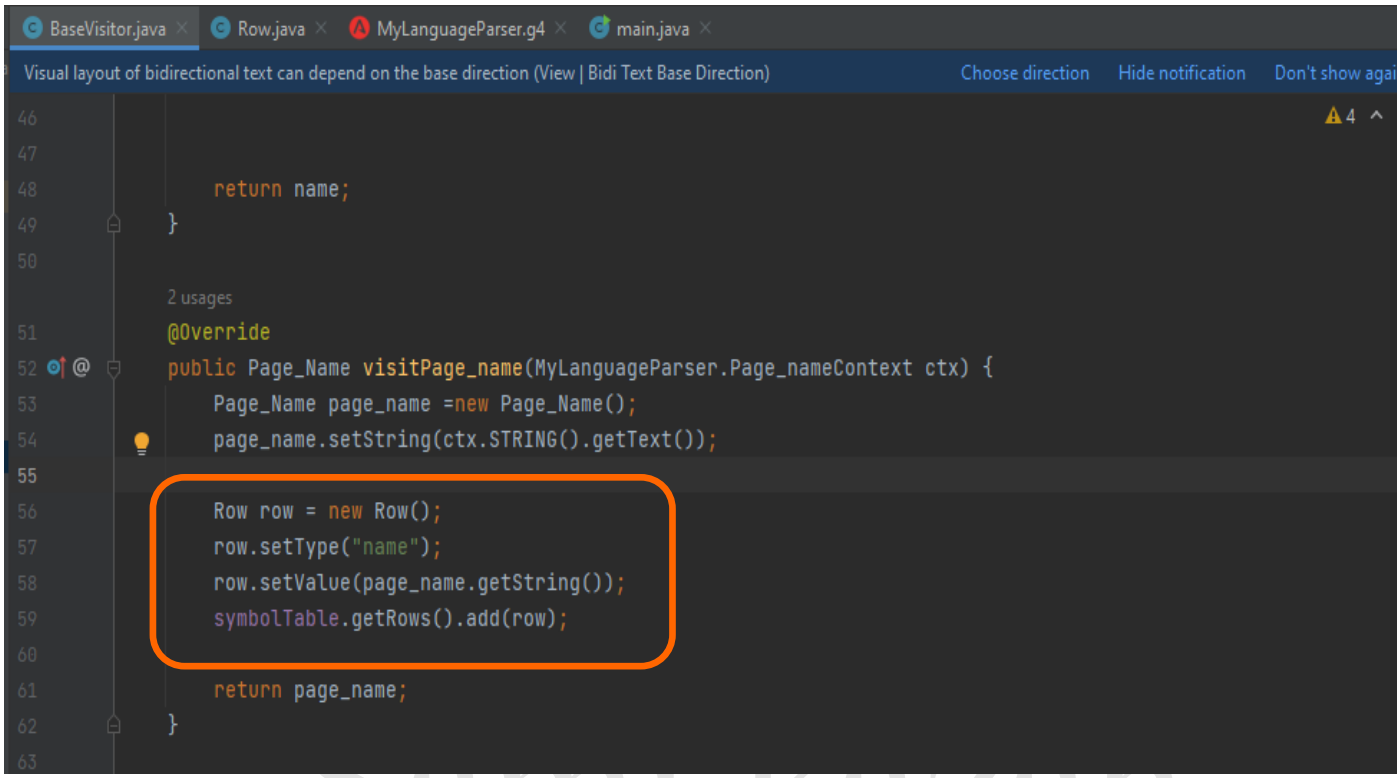
- نقوم بإنشاء الصف SymbolTable يحتوي على List of rows بالإضافة لتوابع ال get و ال set بالإضافة لتابع للطباعة .

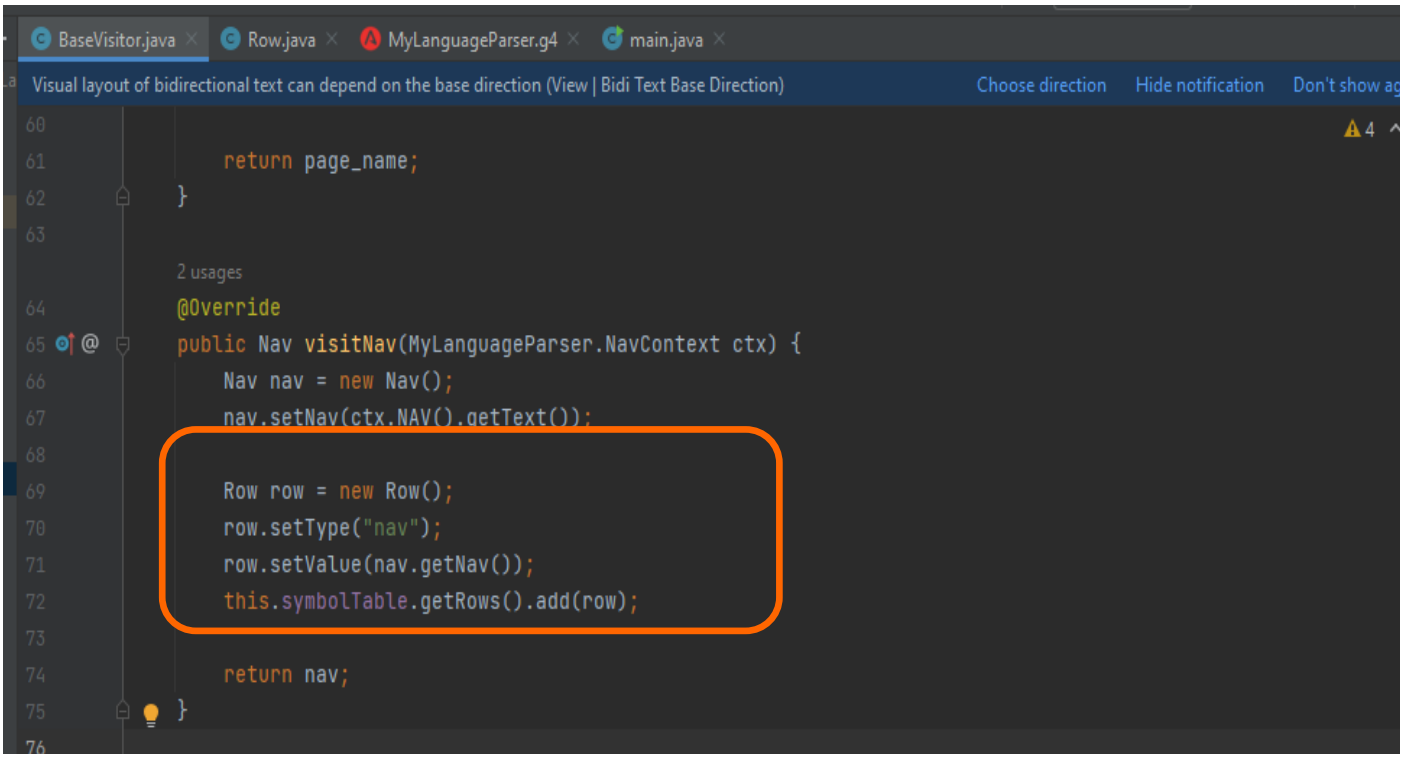
```java
// Visitor.java | Row.java | SymbolTable.java | MyLanguageParser.g4 | main.java
import java.util.List;

2 usages
public class SymbolTable {
    6 usages
    List <Row> rows = new ArrayList<>();

    3 usages
    public List<Row> getRows() {
        return rows;
    }

    public void setRows(List<Row> rows) {
        this.rows = rows;
    }
    1 usage
    public void print() {
        for (int i = 0; i < rows.size(); i++) {
            if(rows.get(i)!=null)
            {
                System.out.println(rows.get(i).getType() + "\t\t\t\t" + rows.get(i).getValue());

            }
        }
    }
}
```

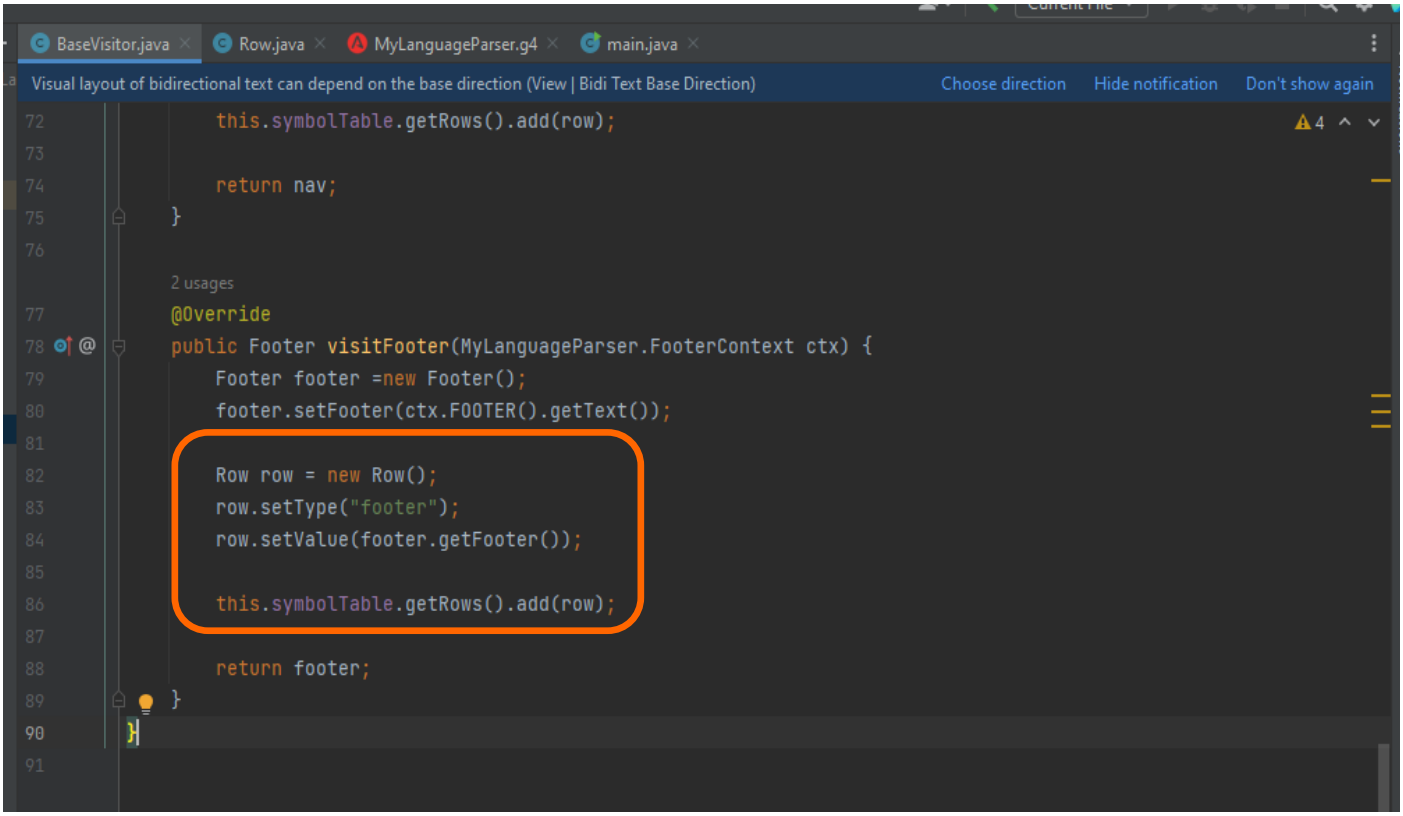
- نذهب إلى ال Base Visitor ونقوم بتعريف SymbolTable

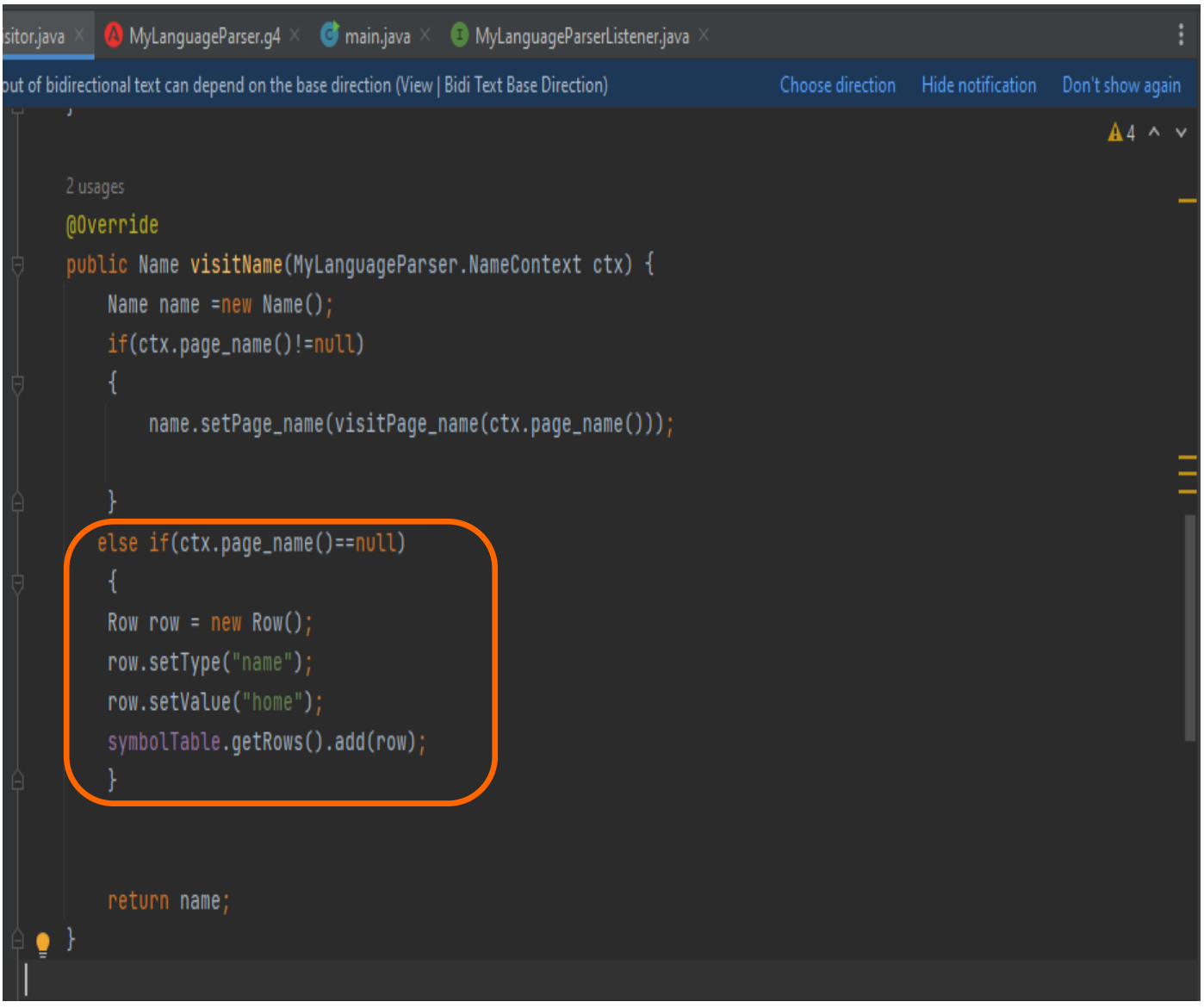
```java
// BaseVisitor.java | Row.java | SymbolTable.java | MyLanguageParser.g4 | main.java
import gen.MyLanguageParser;
import gen.MyLanguageParserBaseVisitor;

1 usage
public class BaseVisitor extends MyLanguageParserBaseVisitor {


    SymbolTable symbolTable = new SymbolTable();
    Row row = new Row();



    1 usage
    @Override
    public Page visitPage(MyLanguageParser.PageContext ctx) {
        Page page =new Page();
        for (int i = 0; i < ctx.page_style().size(); i++) {
            if(ctx.page_style(i)!=null)
            {
                page.getPage_styleList().add(visitPage_style(ctx.page_style(i)));

            }
        }
        return page;
```

بما أن ال Base visitor يمر على القواعد بالترتيب فعند الوصول لتابع زيارة القاعدة المطلوب إضافتها لجدول الرموز نقوم بتعريف سط جديد ثم نقوم بإضافته للجدول كما في الشكل :






BaseVisitor.java
Row.java
MyLanguageParser.g4
main.java
Visual layout of bidirectional text can depend on the base direction (View | Bidi Text Base Direction)
Choose direction
Hide notification
Don't show again
this.symbolTable.getRows().add(row);
return nav;
}
2 usages
@Override
public Footer visitFooter(MyLanguageParser.FooterContext ctx) {
Footer footer =new Footer();
footer.setFooter(ctx.FOOTER().getText());
Row row = new Row();
row.setType("footer");
row.setValue(footer.getFooter());
this.symbolTable.getRows().add(row);
return footer;
}
}


MyLanguageParser.g4
main.java
MyLanguageParserListener.java
Choose direction
Hide notification
Don't show again
2 usages
@Override
public Name visitName(MyLanguageParser.NameContext ctx) {
Name name =new Name();
if(ctx.page_name()!=null)
{
name.setPage_name(visitPage_name(ctx.page_name()));
}
else if(ctx.page_name()==null)
{
Row row = new Row();
row.setType("name");
row.setValue("home");
symbolTable.getRows().add(row);
}
return name;
}

بعد ذلك نضع method print في القاعدة الرأس page لطباعة ال symbol table

```
BaseVisitor.java   Row.java   MyLanguageParser.g4   main.java
Visual layout of bidirectional text can depend on the base direction (View | Bidi Text Base Direction)   Choose direction   Hide notification   Don't show again

    1 usage
    @Override
    public Page visitPage(MyLanguageParser.PageContext ctx) {
        Page page =new Page();
        for (int i = 0; i < ctx.page_style().size(); i++) {
            if(ctx.page_style(i)!=null)
            {
                page.getPage_styleList().add(visitPage_style(ctx.page_style(i)));

            }
        }
        this.symbolTable.print();

        return page;
    }
```

نقوم بعمل run لملف ال main فيكون جدول الرموز بالشكل :

```
name                    my new website
nav                 add nav
footer                  add footer

Process finished with exit code 0
```

و هو مماثل للجدول المتصور :

Name	MyWebPageName / Home
Nav	Add nav
Footer	Add footer

ملاحظة : يمكنك الحصول على رقم السطر الذي يحوي token باستخدام getSymbol().getLine()

ctx.PAGENAME.getSymbol().getLine();

Semantic Check

و هي مرحلة التحقق الدلالي و هي متعلقة بشكل أساسي من التأكد من خلو الأكواد البرمجية من الأخطاء المتعلقة بالنوع مثل اسناد قيم إلى متغيرات معرفة على أنها من نوع مختلف أو استخدام متغير غير معرف مسبقاً .

```
-------1--------
int x =5 ;
x = "monster" ;
--------2--------
x = x + y ;
-------3---------
int x = 3;
String x = "monster";
```

اما بلغة البرمجة الجديدة التي قمنا بإنشائها سنقوم بمعالجة ثلاث حالات :

١- - يجب أن يكون ال name أول ما يتم تعريفه .

٢- لا يسمح بتكرار أي تعليمة .

٣ .يجب أن يكون ال nav معرفا .

```
1  ----------1----------
2  hi {
3  add nav ;
4  add footer;
5  name ('my new website');
6  }
7  --------2-----------
8  hi {
9  name ('my new website');
10 name ('my new website2');
11 add nav ;
12 add nav ;
13 add footer;
14 add footer;
15 }
16 --------3----------
17 hi {
18 add nav ;
19 add footer;
20 }
21
```

إنجاز ال Semantic Check

- نقوم بإنشاء صف جديد SemanticCheck.

- يحتوي على كائن SymbolTable / بالإضافة لل get / set الخاصة به .

- يحتوي على تابع منطقي check.

- يحتوي على ثلاث توابع كل تابع مخصص لحالة معينة .

- تأخذ هذه التوابع ال SymbolTable كبرامتر .


```
1 usage
boolean check() {
    if(!checkIfNameIsFirst(this.symbolTable))
    {
        System.out.println("Exception Name Must Come First !");
        return false ;}

         else if(!checkForRedublication(this.symbolTable))
         { System.out.println("Exception Reduplicated functions !");
             return false ;
         }
            else if(!checkIfNavExist(this.symbolTable))
             { System.out.println("Exception Nav Does Not Exist !");
                 return false;
             }
                 else
                     return true ;
}
```

بما أن SymbolTable جاهز و تم تعبئته بشكل مرتب حسب الكود البرمجي المدخل فلمعالجة أول حالة يكفي التأكد من أن أول عنصر هو name .

```
boolean checkIfNameIsFirst (SymbolTable symbolTable) {
    for (int i = 0; i < symbolTable.rows.size(); i++) {
        if(symbolTable.rows.get(i)!=null)
        {
            if(!symbolTable.rows.get(0).getType().contains("name"))
            {return false;}
        }
    }
   return true;
}
1 usage
```

كما يتم معالجة الحالة الثالثة و هي التأكد من وجود nav بنفس الطريقة لكن بتعديل الشرط الثاني و قلب القيمة التي يعيدها التابع :

```
    1 usage
@   boolean checkIfNavExist (SymbolTable symbolTable)
    {
        for (int i = 0; i < symbolTable.rows.size(); i++) {
            if(symbolTable.rows.get(i)!=null)
            {
                if(symbolTable.rows.get(i).getType().contains("nav"))
                {return true;}
            }
        }
        return false;
    }
```

- لمعالجة الحالة الثانية (عدم التكرار) يجب علينا المرور بحلقتين أول حلقة تمر بالعناصر و الحلقة الثانية تتأكد من عدم تكراره .

```
1 usage
boolean checkForRedublication (SymbolTable symbolTable)
{
    for (int i = 0; i < symbolTable.rows.size(); i++) {
        if(symbolTable.rows.get(i)!=null)
        {
            for (int j = symbolTable.rows.size()-1; j > i; j--) {
                if(symbolTable.rows.get(j).getType().equals(symbolTable.rows.get(i).getType()))
                {
                    return false;
                }
            }
        }
    }
    return true;
}
```

- الآن نقوم بإنشاء كائن Semantic Check و نستخدمه في ال BaseVisitor


```
@Override
public Page visitPage(MyLanguageParser.PageContext ctx) {
    Page page =new Page();
    for (int i = 0; i < ctx.page_style().size(); i++) {
        if(ctx.page_style(i)!=null)
        {
            page.getPage_styleList().add(visitPage_style(ctx.page_style(i)));

        }
    }

    this.symbolTable.print();
    SemnticCheck semnticCheck = new SemnticCheck();
    semnticCheck.setSymbolTable(this.symbolTable);
    semnticCheck.check();

    return page;
}
```

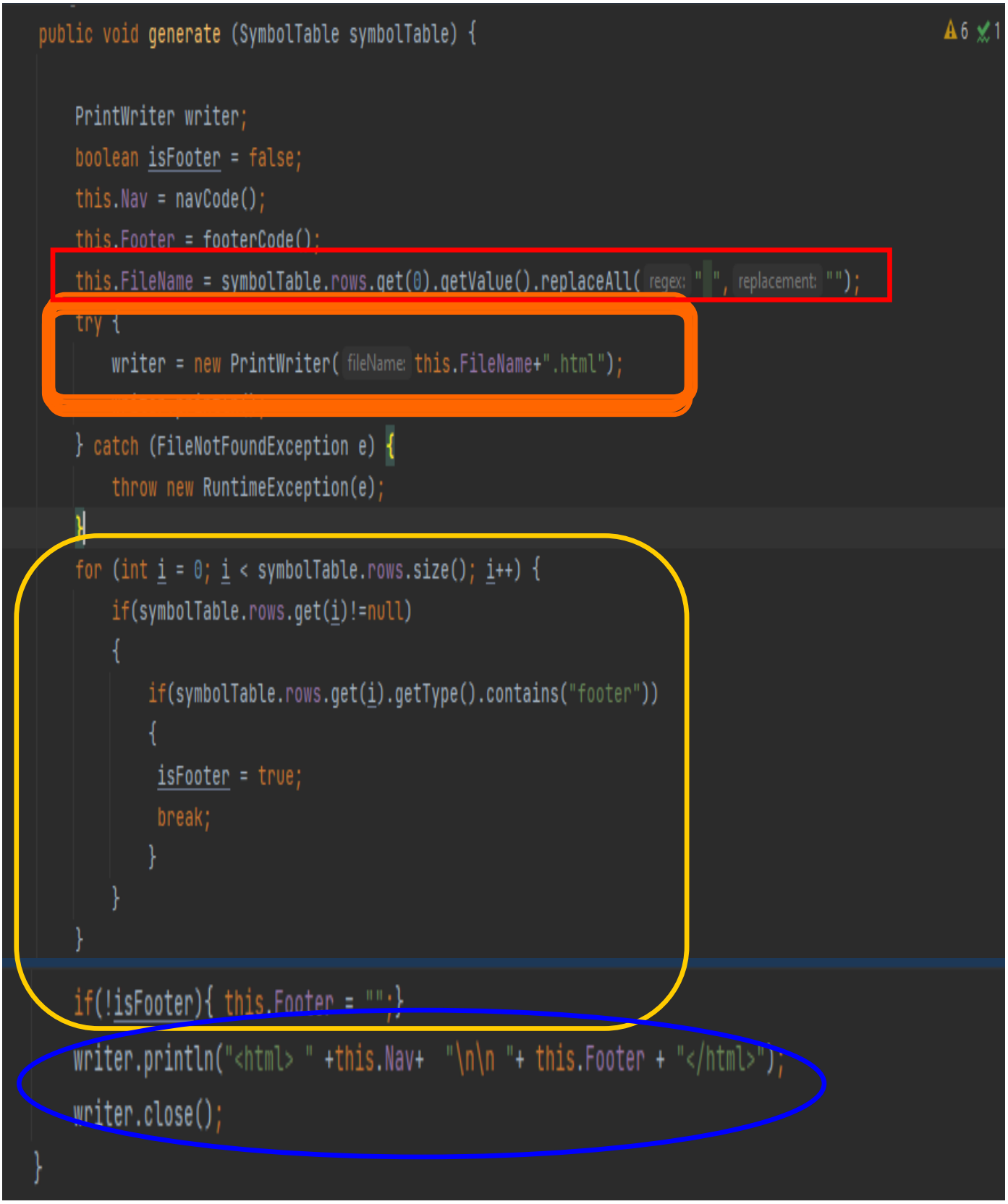
Code Generation

- و هي المرحلة الأخيرة التي نقوم بها باستبدال الكود المكتوب بلغتنا الخاصة بكود مقابل من ال Html و ال CSS.
- بدايةً نقوم بتعريف صف CodeGeneration
- نعرف داخله ال Symbol table بالإضافة لتوابع ال get / set
- نعرف داخله ثلاث string و هم filename , navCode , footerCode
- بالإضافة للتابع generate الذي سننشئ منه الملف و نكتب داخله الكود المولد .
- التابعين navCode footerCode يعيد كل منهما كود على شكل string


```
2 usages
public class CodeGeneration {

    2 usages
    SymbolTable symbolTable;

    String FileName = "";
    1 usage
    String Nav = "";
    1 usage
    String Footer ="";
    1 usage
    public void generate (SymbolTable symbolTable) {
    }
    public String navCode () {...}
    public String footerCode() {...}
    public SymbolTable getSymbolTable() {
        return symbolTable;
    }
    public void setSymbolTable(SymbolTable symbolTable) {
        this.symbolTable = symbolTable;
    }
```

```
public String navCode () {

    return this.Nav ="<header>\n" +
                    "     <!-- Top header menu containing\n" +
                    "         logo and Navigation bar -->\n" +
                    "     <div id=\"top-header\">\n" +
                    "                         \n" +
                    "          <!-- Logo -->\n" +
                    "          <div id=\"logo\">\n" +
                    "   \n" +
                    "          </div>    \n" +
                    "           \n" +
                    "          <!-- Navigation Menu -->\n" +
                    "          <nav>\n" +
                    "                              \n" +
                    "          </nav>\n" +
                    "     </div>    \n" +
                    "   \n" +
                    "     <!-- Image menu in Header to contain an Image
                    "           a sample text over that image -->\n" +
                    "     <div id=\"header-image-menu\">\n" +
                    "   \n" +
                    "     </div>\n" +
                    "</header>";
```

```
public String navCode () {...}
public String footerCode() {
    return this.Footer = " <footer>\n" +
            "         \n" +
            "                    <a href=\n" +
            "\"https://www.geeksforgeeks.org/about/\">About Us</a>|\n" +
            "                    <a href=\n" +
            "\"https://www.geeksforgeeks.org/privacy-policy/\">Privacy Policy</a>|\n" +
            "                    <a href=\n" +
            "\"https://www.geeksforgeeks.org/careers/\">Careers</a>\n" +
            "                \n" +
            "             \n" +
            " \n" +
            "<p>@geeksforgeeks, Some rights reserved</p>\n" +
            " \n" +
            " \n" +
            "             \n" +
            "         </footer>";
}
```



- بعد الحصول على اسم الصفحة قمنا بإنشاء ملف يحمل الاسم الذي تم إدخاله باللغة الجديدة .
- تأكدنا من وجود ال footer و في حال عدم وجودها حذفنا الكود .
- قمنا بكتابة الكود الجديد ضمن الملف .

المنتج النهائي :

mynewwebsite.html

E: > compiler > MyNewLanguage > mynewwebsite.html > ...

```
<html> <header>
    <!-- Top header menu containing
       logo and Navigation bar -->
    <div id="top-header">

        <!-- Logo -->
        <div id="logo">

        </div>

        <!-- Navigation Menu -->
        <nav>

        </nav>
    </div>

    <!-- Image menu in Header to contain an Image and
        a sample text over that image -->
    <div id="header-image-menu">

    </div>
</header>

 <body>  <footer>

                    <a href=
"https://www.geeksforgeeks.org/about/">About Us</a>
                    <a href=
"https://www.geeksforgeeks.org/privacy-policy/">Privacy Policy</a>
                    <a href=
"https://www.geeksforgeeks.org/careers/">Careers</a>


<p>@geeksforgeeks, Some rights reserved</p>



        </footer> </body>   </html>

```

المراجع :

https://ar.wikipedia.org/wiki/%D9%82%D9%88%D8%A7%D8%B9%D8%AF_%D8%AE%D8%A7%D9%84%D9%8A%D8%A9_%D9%85%D9%86_%D8%A7%D9%84%D8%B3%D9%8A%D8%A7%D9%82

https://ar.wikipedia.org/wiki/%D9%84%D8%BA%D8%A9_%D8%B7%D8%A8%D9%8A%D8%B9%D9%8A%D8%A9#:~:text=%D8%A7%D9%84%D9%84%D8%BA%D8%A7%D8%AA%20%D8%A7%D9%84%D8%B7%D8%A8%D9%8A%D8%B9%D9%8A%D8%A9%20%D8%AA%D8%AE%D8%AA%D9%84%D9%81%20%D8%B9%D9%86%20%D8%A7%D9%84%D9%84%D8%BA%D8%A7%D8%AA,%D8%A8%D8%B4%D9%83%D9%84%20%D8%A7%D8%B5%D8%B7%D9%86%D8%A7%D8%B9%D9%8A%20%D8%AA%D9%82%D9%84%D9%8A%D8%AF%D8%A7%D9%8B%20%D9%84%D9%84%D9%84%D8%BA%D8%A7%D8%AA%20%D8%A7%D9%84%D8%B7%D8%A8%D9%8A%D8%B9%D9%8A%D8%A9.

https://tomassetti.me/antlr-mega-tutorial/

https://www.techtarget.com/whatis/definition/compiler

https://www.chakray.com/programming-languages-types-and-features/#:~:text=A%20programming%20language%20is%20a,form%20and%20organise%20computer%20instructions.

https://3alam.pro/ihanan95/articles/semantic-analysis-part-4

https://ar.theastrologypage.com/abstract-syntax-tree#:~:text=%D8%B4%D8%AC%D8%B1%D8%A9%20%D8%A8%D9%86%D8%A7%D8%A1%20%D8%A7%D9%84%D8%AC%D9%85%D9%84%D8%A9%20%D8%A7%D9%84%D9%85%D8%AC%D8%B1%D8%AF%D8%A9%20(AST,%D8%A7%D9%84%D8%A5%D9%86%D8%B4%D8%A7%D8%A1%D8%A7%D8%AA%20%D9%81%D9%8A%20%D8%A7%D9%84%D9%84%D8%BA%D8%A9%20%D9%88%D8%A7%D9%84%D9%82%D9%88%D8%A7%D8%B9%D8%AF%20%D8%A7%D9%84%D9%84%D8%A7%D8%AD%D9%82%D8%A9.

https://www.geeksforgeeks.org/symbol-table-compiler/

https://www.geeksforgeeks.org/html5-footer-tag/

https://www.geeksforgeeks.org/html-course-creating-navigation-menu/

The End