

المترجمات

دليل عملي مخصص للطلاب



باستخدام ال ANTRL

تعلم كيف تكتب لغتك الخاصة

سامي قزح

الفهرس :

(٣)..... المقدمة

الفصل الأول : أساسيات

(٥)..... مفاهيم نظرية أساسية

(٦)..... مراجعة للتعبير المنتظمة

(٧)..... شرح ملفات ال Anlir

(١١)..... بعض الأخطاء الشائعة (١)

الفصل الثاني : بناء المترجم (١)

(١٦)..... الإضافة على لغة أم كتابة لغة جديدة؟

(١٧)..... كتابة رموز اللغة

(١٩)..... إضافة قواعد اللغة

(٢٠)..... توليد شجرة ال Parse Tree

الفصل الثالث : بناء المترجم (٢)

(٢٢)..... ما هي ال شجرة ال AST؟

(٢٣)..... مقارنة بين ال AST و ال Parse Tree

(٢٤)..... بناء شجرة ال AST

(٣٦)..... توليد شجرة ال AST

(٣٧)..... بعض الأخطاء الشائعة (٢)

الفصل الرابع : بناء المترجم (٣)

(٣٩)..... ما هو ال Symbol Table؟

(٤٠)..... بناء Symbol Table

(٤٦)..... ما هي ال Semantic Check

(٤٧)..... إنجاز ال Semantic Check

(٥٠)..... توليد الكود Code Generation

(٥٤)..... المراجع

المقدمة

خلال مسيرتي الجامعية واجهت الكثير من العقبات ، و المطبات الصعبة لكنها ما لبثت أن تذلت 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 بسيطةً 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 (Intellij (IDE) (يفضل استخدامه)
- Java Jdk (يمكنك استخدام أي لغة أخرى تدعم ال OOP)
- أداة Antlr (يمكن تثبيتها كإضافة Plugin موجودة ضمن ال IDE)
- قد تحتاج Vpn لتثبيت الأداة السابقة (Proton vpn كإقتراح)
- 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 .

Sami kazah

الفصل الأول:
أساسيات

Sami kazah

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 - 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 .

٢- 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 :

١- ترميز مفردات اللغة

٢- بناء قواعد اللغة

٣- التحقق من الصحة الدلالية للقواعد

٤- توليد الكود الجديد

مراجعة للتعبير 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 لانهاية 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" ;

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

في حال أردنا ترميز (كلمة "وحش" بالإضافة إلى سلسلة من الأرقام والأحرف أو فقط كلمة وحش) ب
Monster فترمز ببساطة بكتابة :

Monster : (('0' .. '9') | ('a'..'z'|'A'..'Z') *) "وحش" ;

* (('0' .. '9') | ('a'..'z'|'A'..'Z') *) 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 ;
```

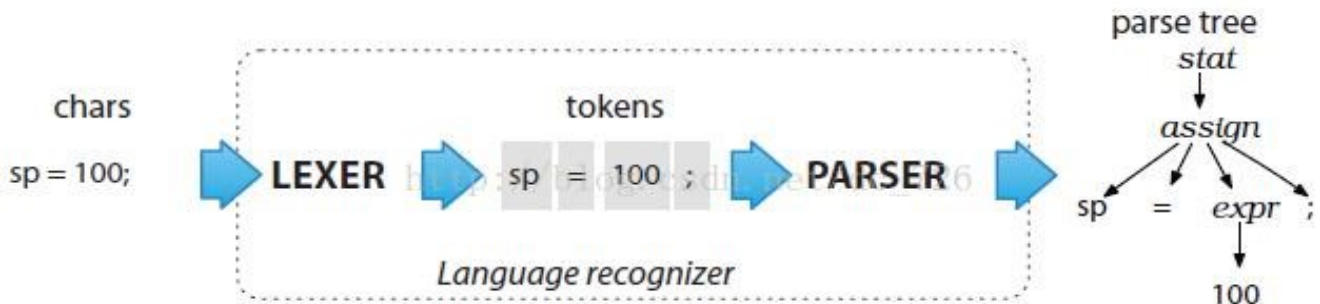
```
expr :NUMBER;
```

```
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 :

- أولاً 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 : ';' ;
```

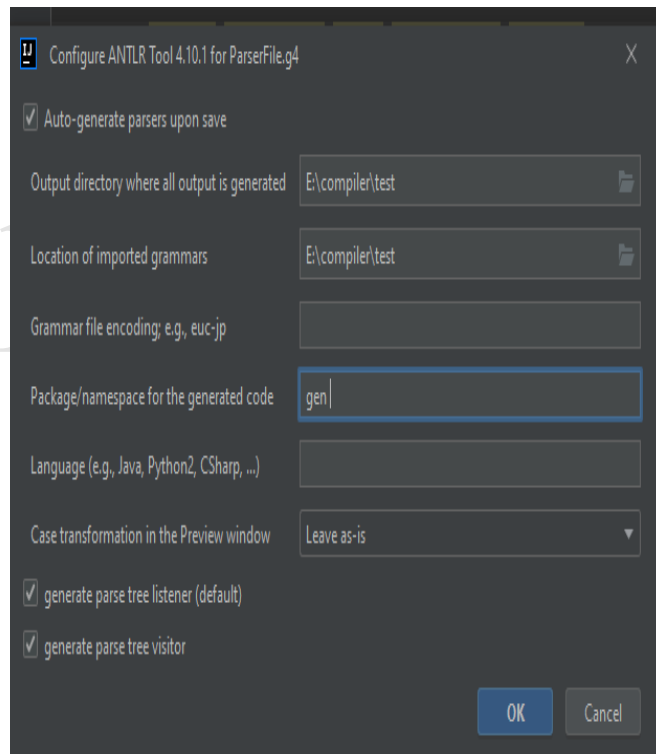
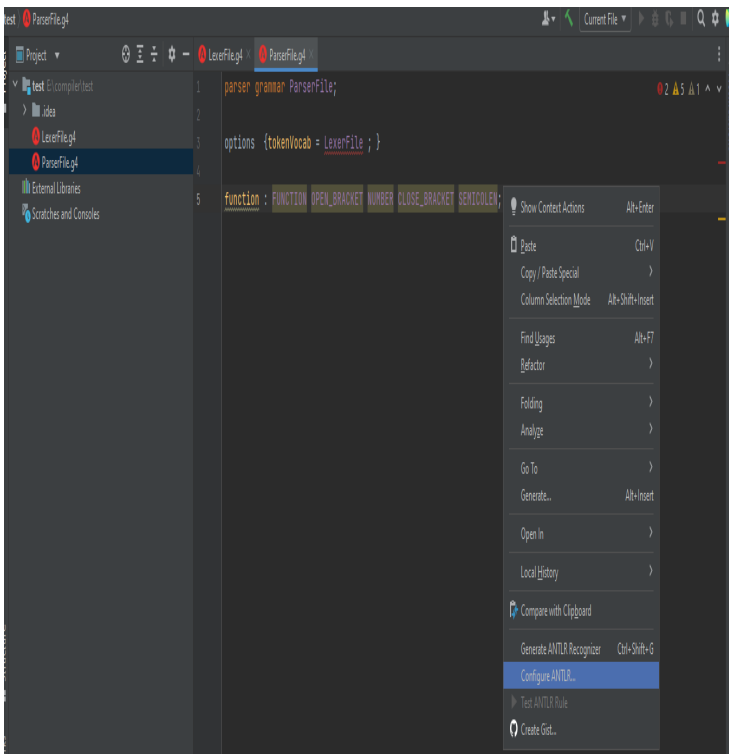
-ثالثاً نقوم بإنشاء ParserFile.g4 لكتابة القاعدة :

```
parser grammar ParserFile;  
options {tokenVocab = LexerFile ; }
```

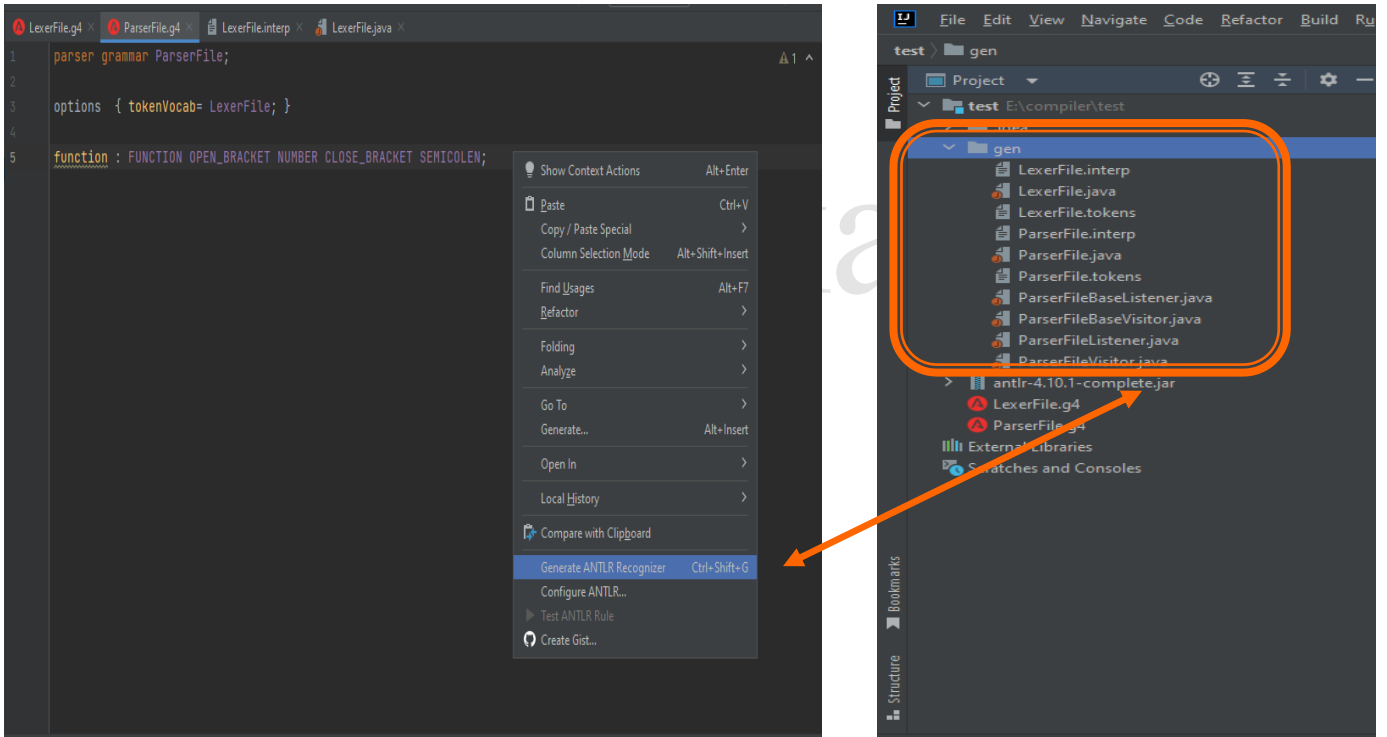
function : FUNCTION OPEN_BRACKET NUMBER CLOSE_BRACKET SEMICOLEN;

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

- رابعاً نقوم بتهيئة البرنامج لتوليد ال tokens

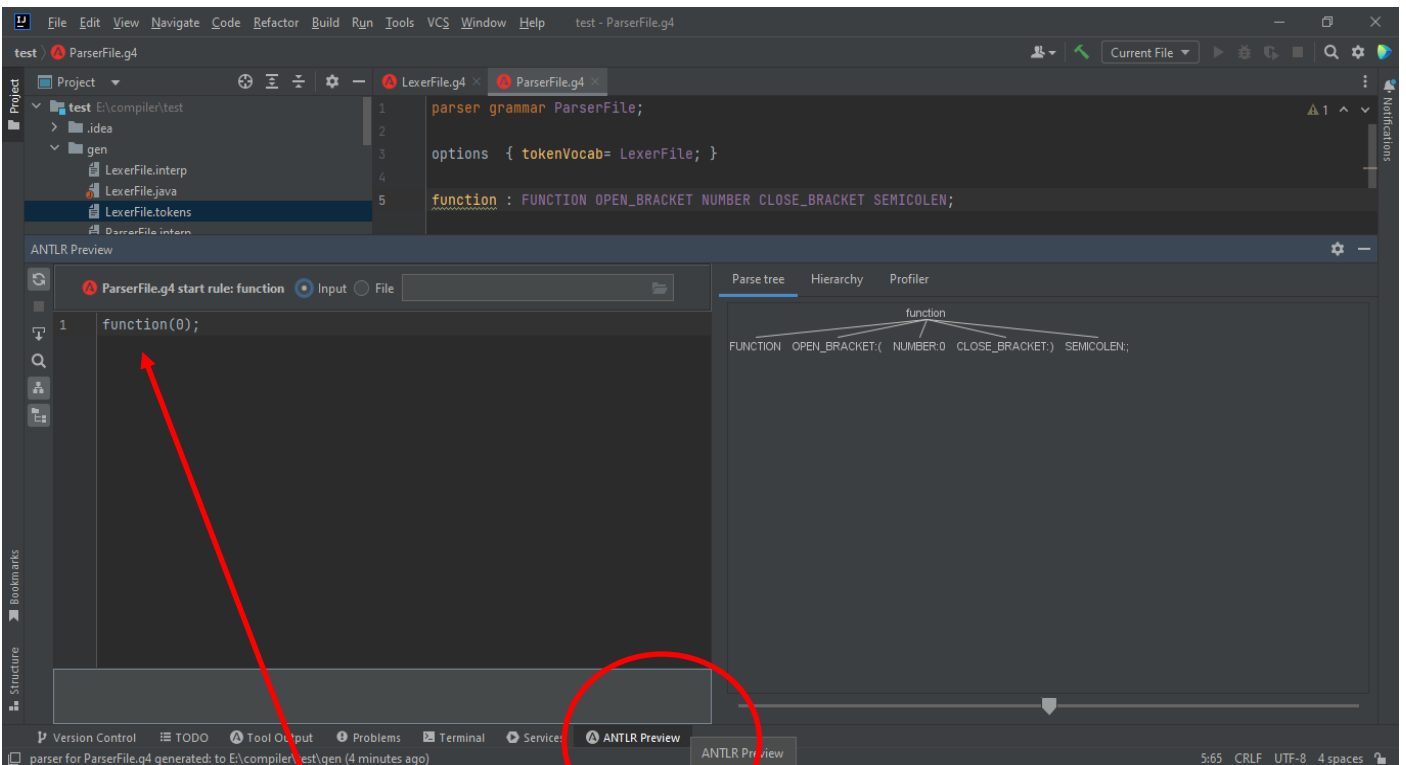


- خامساً نقوم بتوليد الملفات



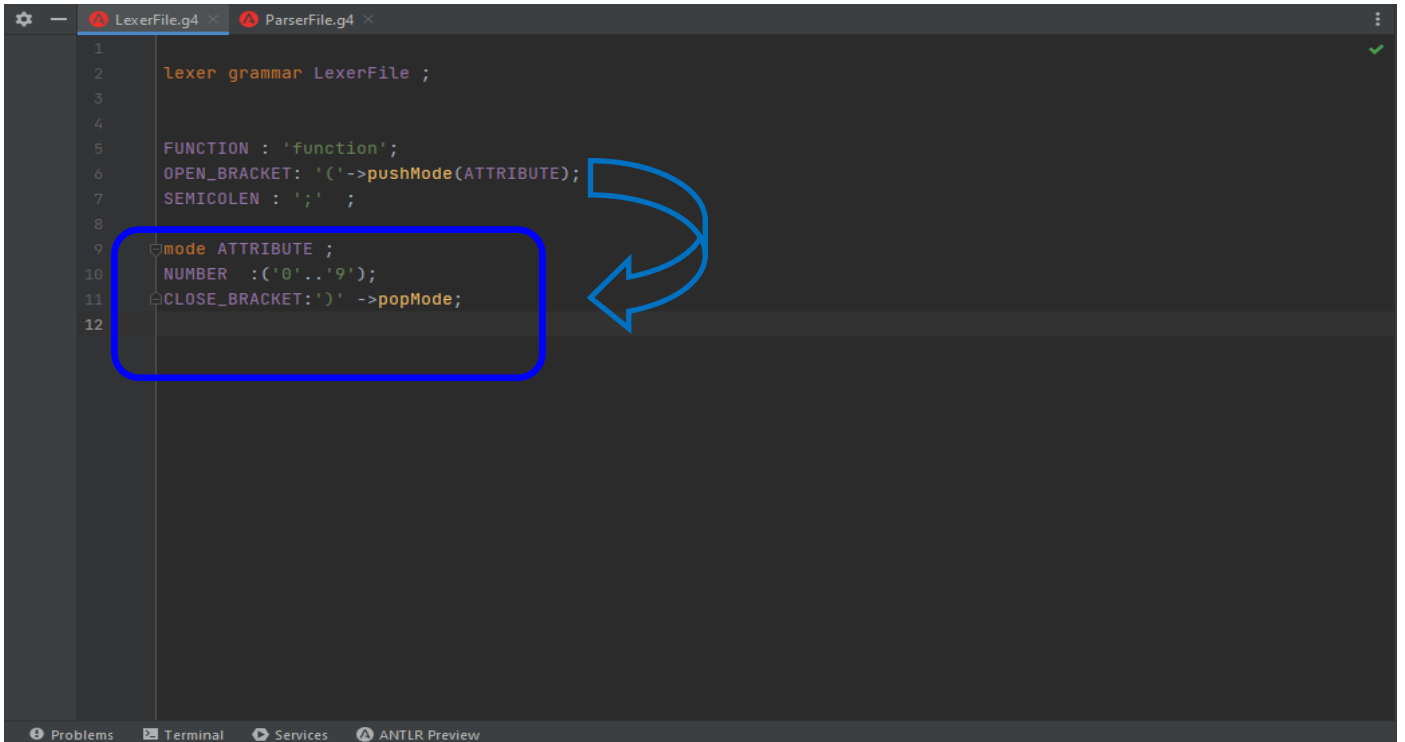
Sami kazah

- سادساً نقوم بتوليد الشجرة



نضغط هنا ثم نكتب النص المراد تجربته

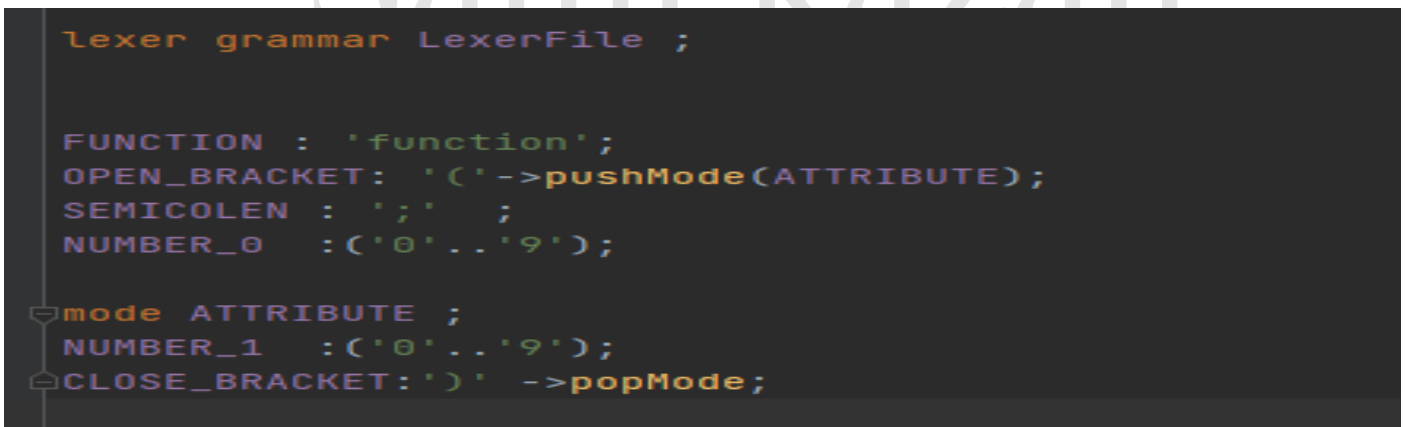
- يمكن تعريف ال Tokens ضمن ملف ال lexer بطريقة اخرى باستخدام ال mode الذي يقوم بصنع
مكدس تكون عناصره منفصلة عن بقية العناصر الأخرى الموجودة بالملف و في مثالنا هنا يكون أول
عنصر فيه هو "(" و آخر عنصر فيه هو ")".



```
1
2  lexer grammar LexerFile ;
3
4
5  FUNCTION : 'function';
6  OPEN_BRACKET : '(' ->pushMode(ATTRIBUTE);
7  SEMICOLEN : ';' ;
8
9  mode ATTRIBUTE ;
10 NUMBER : ('0'..'9');
11 CLOSE_BRACKET : ')' ->popMode;
12
```

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 نستطيع 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)

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

```
function (  ) ;
```

```
function
├── FUNCTION
├── OPEN_BRACKET(
├── NUMBER_1:0
├── CLOSE_BRACKET.)
└── SEMICOLEN;
```

```
line 1:23 token recognition error at: ''
line 1:24 token recognition error at: ''
```

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

```
lexer grammar LexerFile ;

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

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

```

1  lexer grammar LexerFile ;
2
3  FUNCTION : 'function';
4  OPEN_BRACKET : '(' ->pushMode(ATTRIBUTE);
5  SEMICOLEN : ';' ;
6  NUMBER_0  : ('0'..'9');
7
8  SEA_WS1: (' '|'\t'|\n'? '\n')+;
9
10 mode ATTRIBUTE ;
11
12 SEA_WS2: (' '|'\t'|\n'? '\n')+;
13
14 NUMBER_1 : ('0'..'9');
15 CLOSE_BRACKET: ')' ->popMode;
16

```

Samı kazah

```

1  parser grammar ParserFile;
2
3  options { tokenVocab= LexerFile; }
4
5  function :SEA_WS1 FUNCTION SEA_WS1 OPEN_BRACKET SEA_WS2 NUMBER_1 SEA_WS2 CLOSE_BRACKET SEA_WS1 SEMICOLEN;

```


- السلسلة الفارغة غير مسموحة في ال Parser

```
LexerFile.g4 x ParserFile.g4 x
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 parser grammar ParserFile;
2
3 options { tokenVocab= LexerFile; }
4
5 number : NUMBER_0 ;
6 function : FUNCTION ;
7
8 series : (number)* function ;
9
```

في حال أردت أن تضع شرط ليتأكد من إمكانية ورود سلسلة لمرة واحدة فقط أو عدم ورودها إطلاقاً يمكنك استخدام " ? " بالشكل التالي :

```
LexerFile.g4 x ParserFile.g4 x
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 parser grammar ParserFile;
2
3 options { tokenVocab= LexerFile; }
4
5 number : NUMBER_0 ;
6 function : FUNCTION ;
7
8 series : (number+)? function ;
9
10
```

- عدم مراعاة أهمية ترتيب القواعد

```
LexerFile.g4 x ParserFile.g4 x
1 lexer grammar LexerFile ;
2
3 FUNCTION : 'function';
4 OPEN_BRACKET: '('->pushMode(ATTRIBUTE);
5 SEMICOLEN : ';';
6
7
8 STRING : (('a'..'z') | ('0'..'9'))+;
9 VAR : ('a'..'z')+;
10
11
12
13 SEA_WS1: (' '|'\t'|\n'? '\n')+;
14
15 mode ATTRIBUTE ;
16
17 SEA_WS2: (' '|'\t'|\n'? '\n')+;
18 NUMBER_1 : ('0'..'9');
19 CLOSE_BRACKET: ')' ->popMode;
20
```

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

```
LexerFile.g4 x ParserFile.g4 x
1 lexer grammar LexerFile ;
2
3 OPEN_BRACKET: '('->pushMode(ATTRIBUTE);
4
5 FUNCTION : 'function';
6 SEMICOLEN : ';' ;
7 VAR : ('a'..'z')+;
8 QOUT: '*\'' '*';
9 SEA_WS1: (' |\t|\n'? '\n')+;
10
11
12
13 mode ATTRIBUTE ;
14 SEA_WS2: (' |\t|\n'? '\n')+;
15 NUMBER_1 :('0'..'9');
16 CLOSE_BRACKET: ')';
17
```

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.

```
LexerFile.g4 x ParserFile.g4 x
1 lexer grammar LexerFile ;
2
3 OPEN_BRACKET: '('->pushMode(ATTRIBUTE);
4
5 FUNCTION : 'function';
6 SEMICOLEN : ';' ;
7 VAR : ('a'..'z')+;
8 QOUT: '*\'' '*';
9 SEA_WS1: (' |\t|\n'? '\n')+;
10
11
12
13 mode ATTRIBUTE ;
14 SEA_WS2: (' |\t|\n'? '\n')+;
15 NUMBER_1 :('0'..'9');
16 CLOSE_BRACKET: ')'->popMode;
17
```

بحيث في حال أردت استخدام الرموز الموجودة في default mode بعد ولوجك إلى ATTRIBUTE يمكنك الخروج من mode ATTRIBUTE باستخدام CLOSE_BRACKET مما يعيدك إلى default mode

Sami kazah

الفصل الثاني:
بناء المترجم (١)

Sami kazah

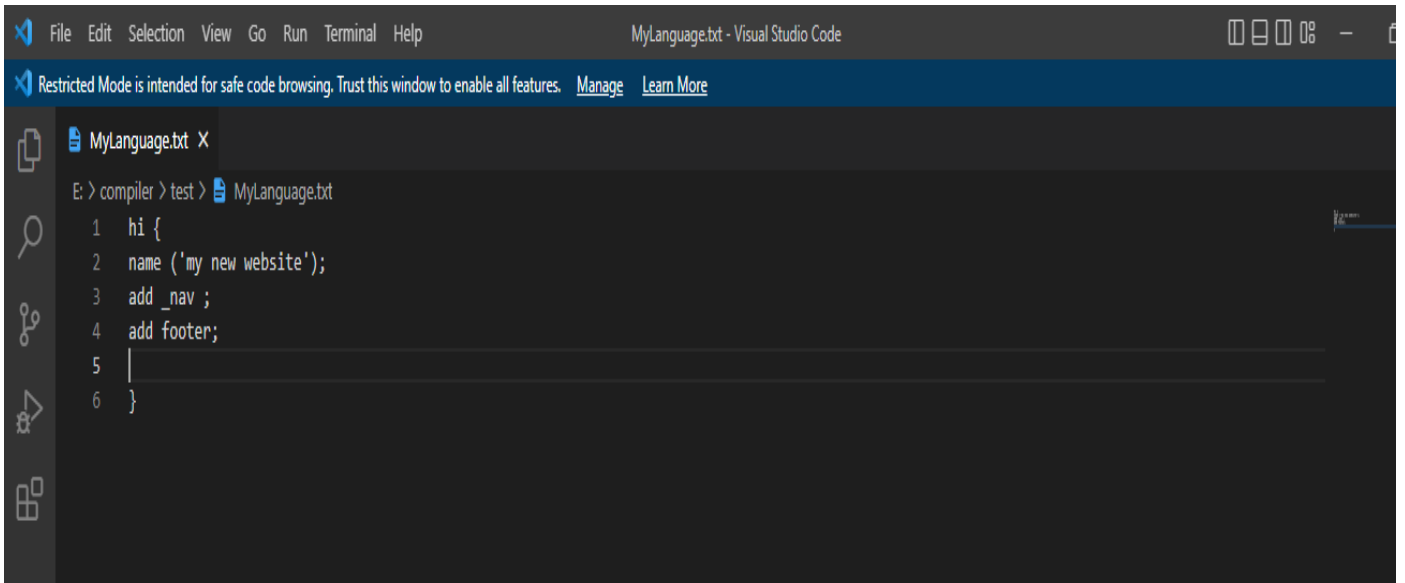
الإضافة على لغة أم كتابة لغة جديدة ؟

- 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 :



```
File Edit Selection View Go Run Terminal Help
MyLanguage.txt - Visual Studio Code
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
MyLanguage.txt X
E: > compiler > test > MyLanguage.txt
1 hi {
2 name ('my new website');
3 add_nav ;
4 add footer;
5 |
6 }
```

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:

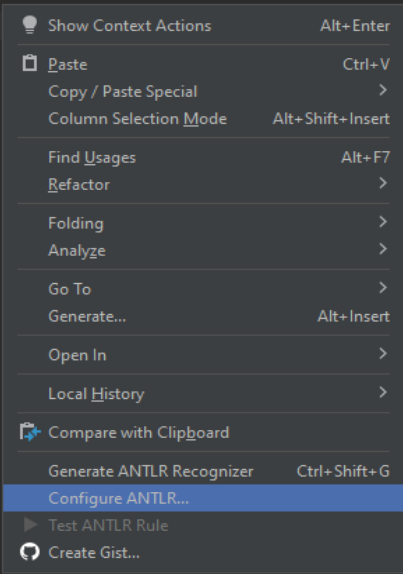
```
1  Lexer grammar MyLanguageLexer ;
2
3
4  NEW_PAGE : ' '* 'hi' ' '* '{' ' '* ->pushMode(PAGE);
5
6  mode PAGE;
7  CLOSE_PAGE : ' '* '}' ' '* ->popMode;
8  NAME : ' '* 'name' ' '* ;
9  NAV : ' '* 'add nav' ' '* ;
10 FOOTER : ' '* 'add footer' ' '* ;
11 OPEN_BRACKET : ' '* '(' ' '* ->pushMode(ATTRIBUTES);
12 SEIMIE_QO : ' '* ';' ' '* ;
13
14 mode ATTRIBUTES;
15 CLOSE_BRACKET : ' '* ')' ' '* ->popMode;
16 QOUT : ' '* '\"' ' '* ;
17 STRING : ('a'..'z' ' '*)+ ;
18
```

وضعت NEW_PAGE في default mode لأنها أول ما سيتم ترميزه .

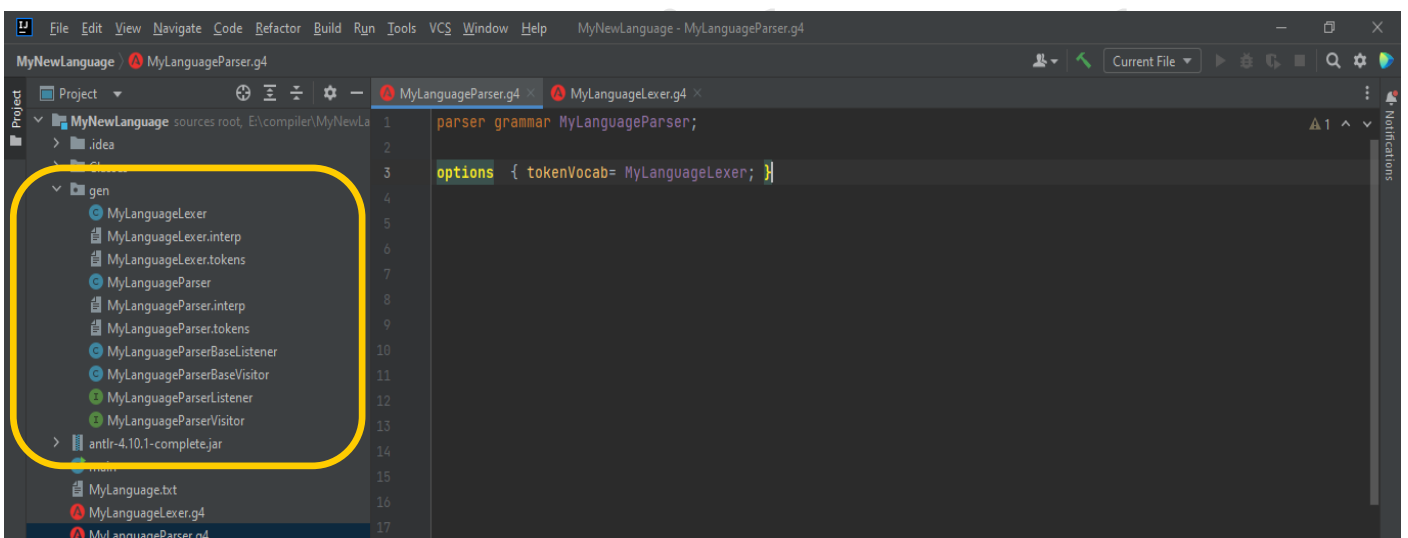
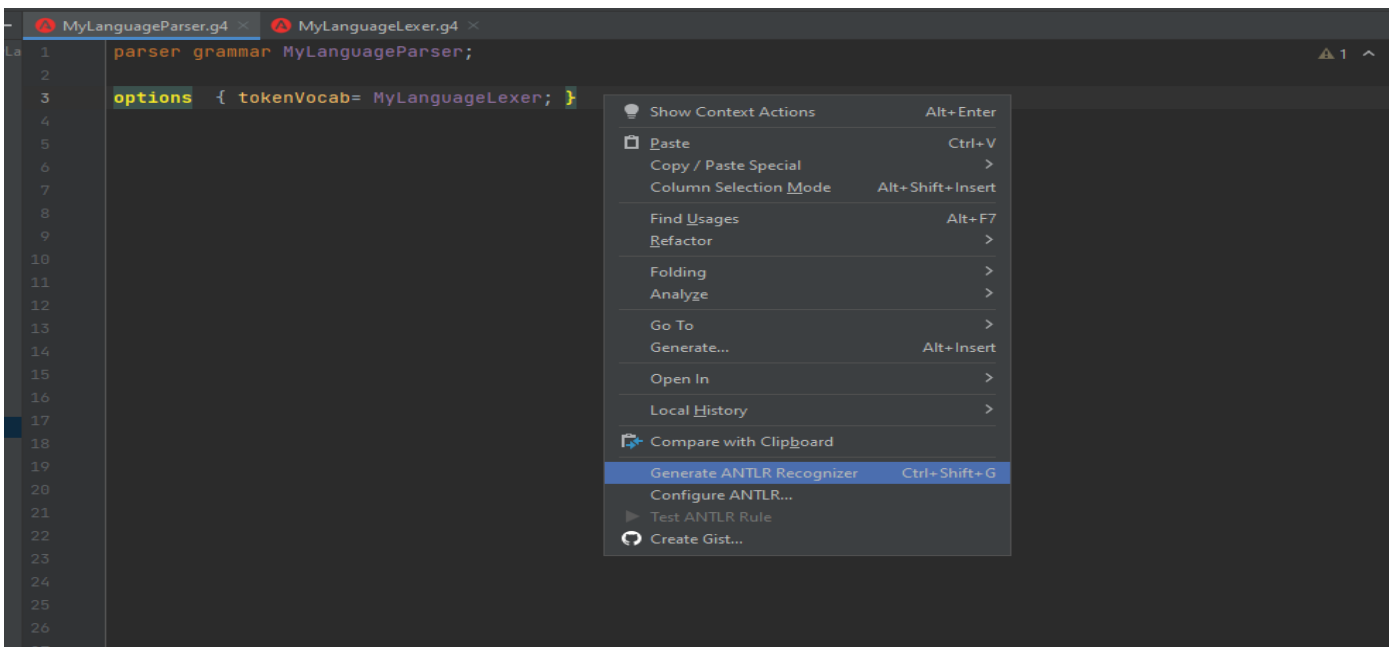
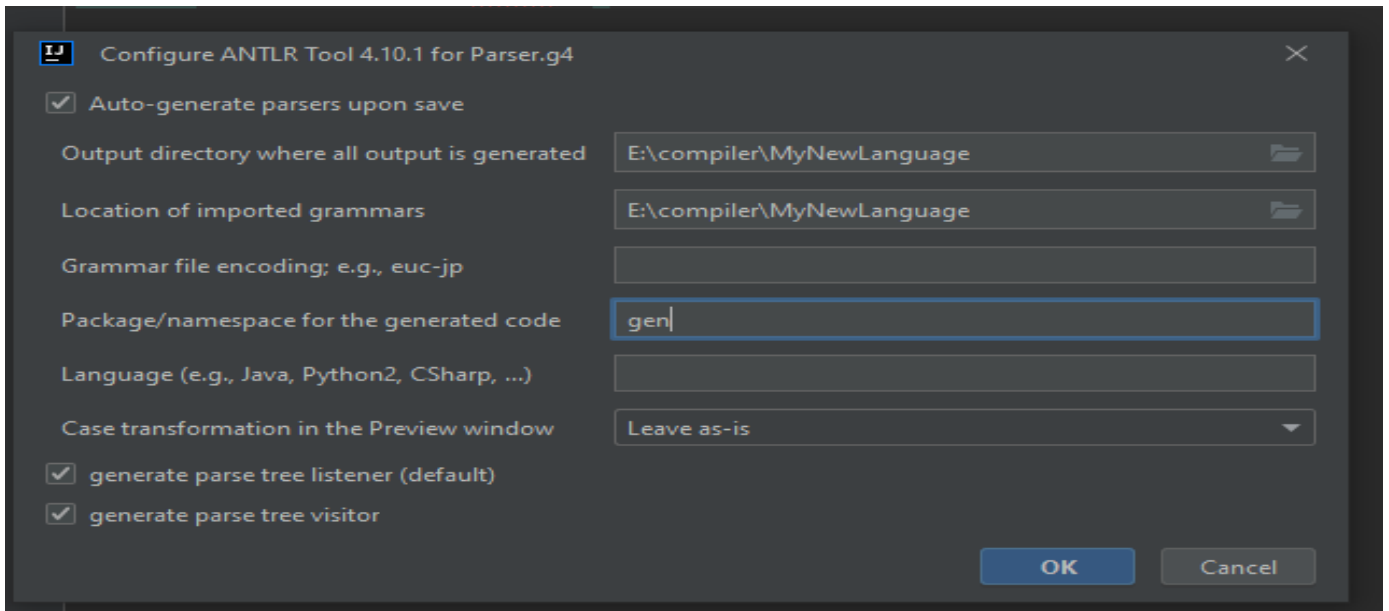
- بعدها قمت بعمل pushMode(PAGE) لفصل الرموز الخاصة بالصفحة عن باقي الرموز .
- قمت بترميز المفردات الخاصة بالصفحة ضمن ال mode PAGE
- قمت بعمل mode ATTRIBUTES لفصل الرموز عن ال mode PAGE

-الآن نقوم بتهيئة البرنامج لتوليد الرموز :

```
1  parser grammar MyLanguageParser;
2
3  options { tokenVocab= MyLanguageLexer; }
```



• الآن نقوم بتوليد الرموز



إضافة قواعد اللغة

```
MyLanguageParser.g4 x MyLanguageLexer.g4 x
1 parser grammar MyLanguageParser;
2
3 options { tokenVocab= MyLanguageLexer; }
4
5 page : NEW_PAGE page_style+ CLOSE_PAGE;
6
7 page_style : (name|nav|footer) SEIMIE_QO ;
8
9 name :NAME OPEN_BRACKET QOUT page_name? QOUT CLOSE_BRACKET;
10
11 page_name:STRING;
12
13 nav : NAV;
14 footer:FOOTER;
15
```

القاعدة الأولى page

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

Sami kazah

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 .

Sami kazah

القاعدة الرابعة page_name

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

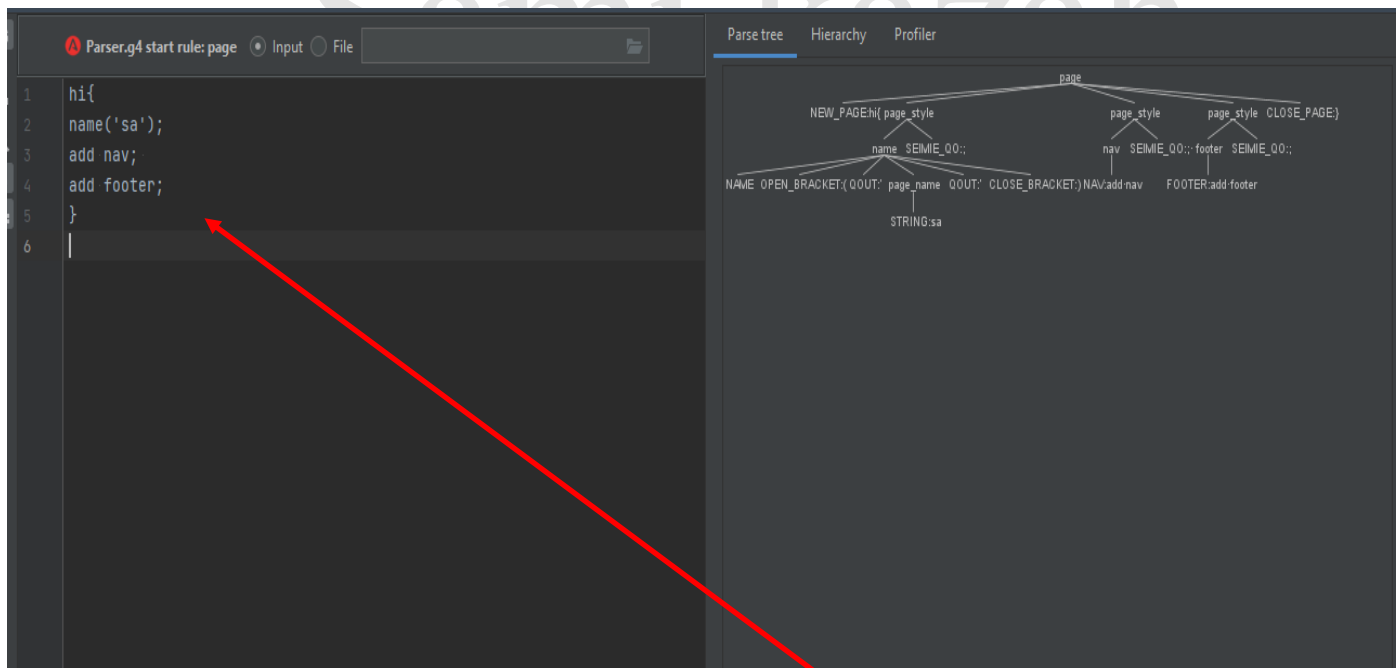
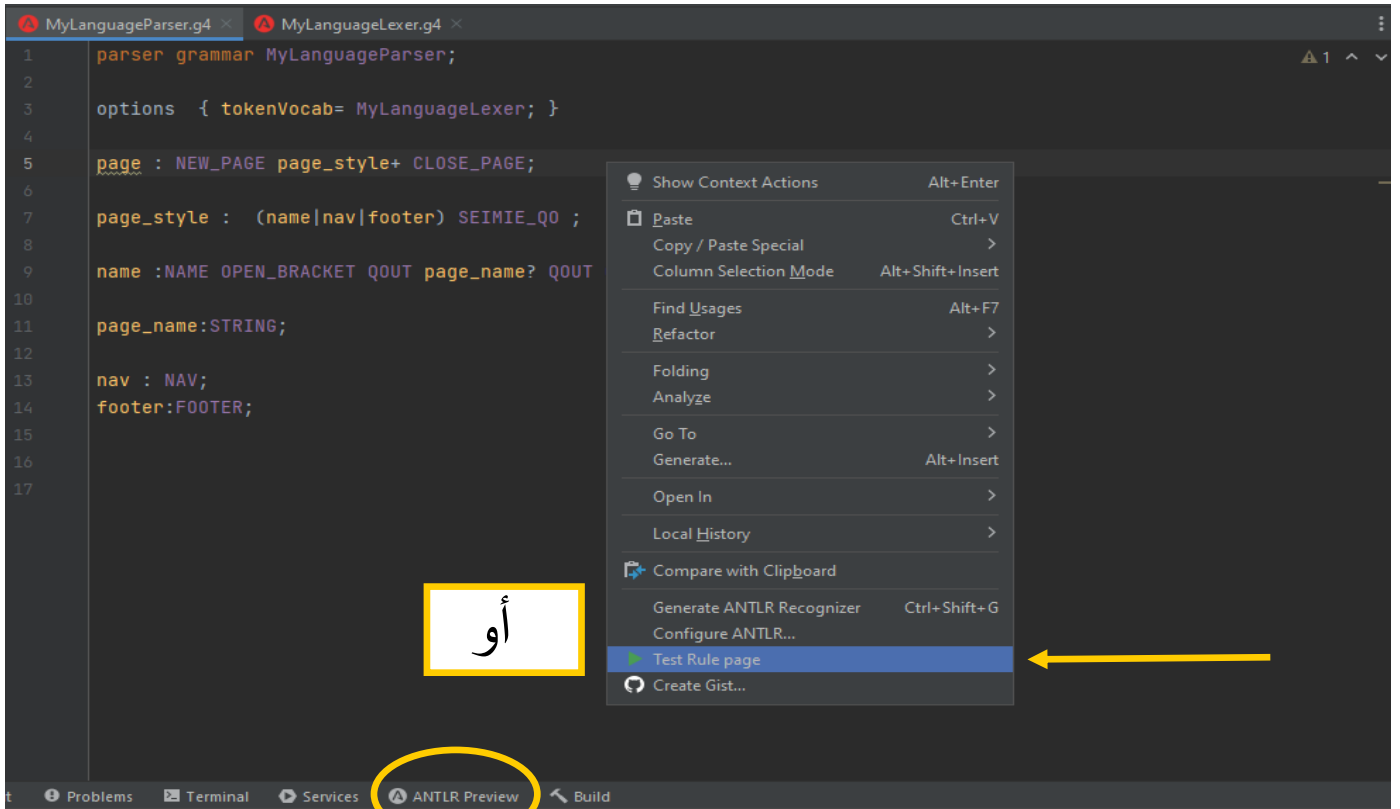
القاعدة الخامسة nav

- تحتوي على رمز NAV الذي يعبر عن nav

القاعدة السادسة footer

- تحتوي على رمز FOOTER الذي يعبر عن footer

توليد شجرة ال Parse Tree



اكتب الكود المراد رسم شجرته

Sami kazah

الفصل الثالث:
بناء المترجم (٢)

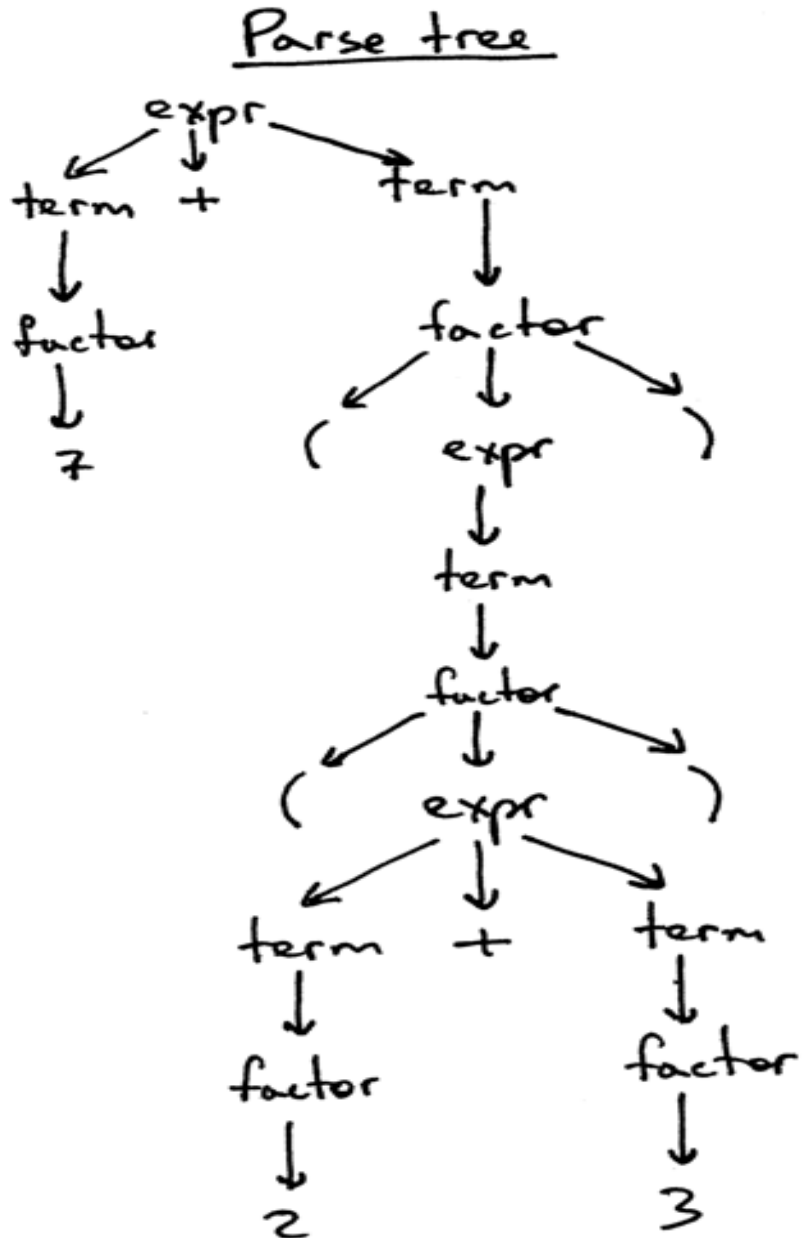
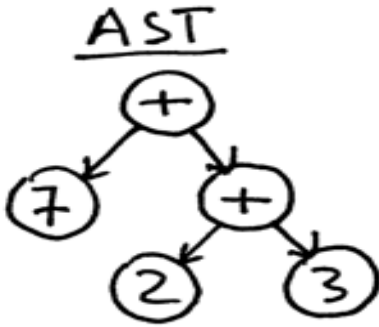
Sami kazah

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.

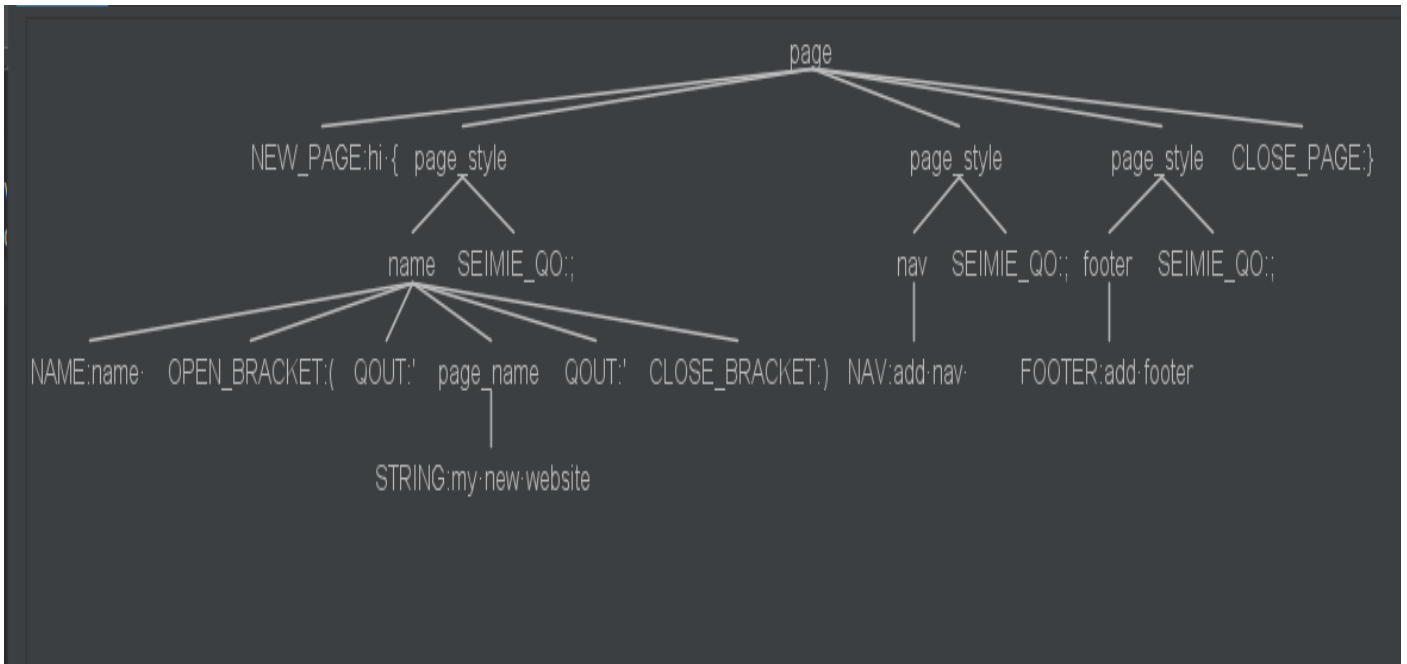
- تمثل ال AST جميع العناصر النحوية للغة البرمجة على غرار ال Parse Tree إلا أنها تركز على القواعد بدلاً عن العناصر مثل الأقواس و الفواصل و الفواصل المنقوطة .

$7 + ((2 + 3))$

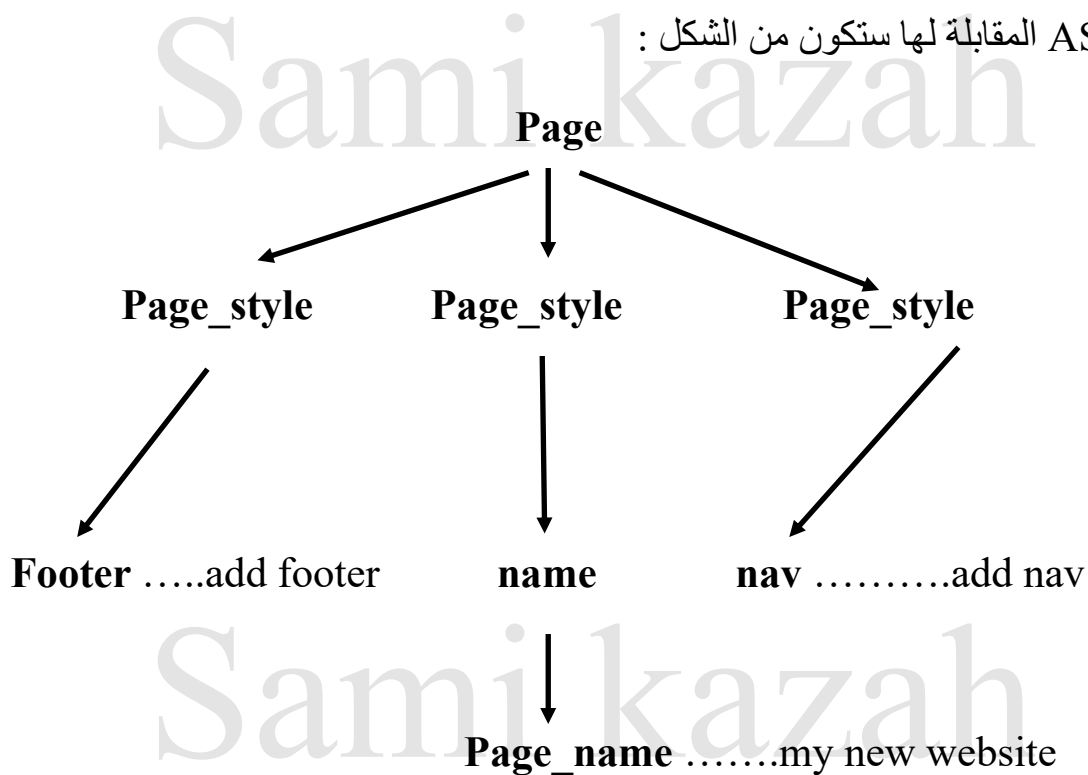


مقارنة بين ال AST و ال Parse Tree

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 :



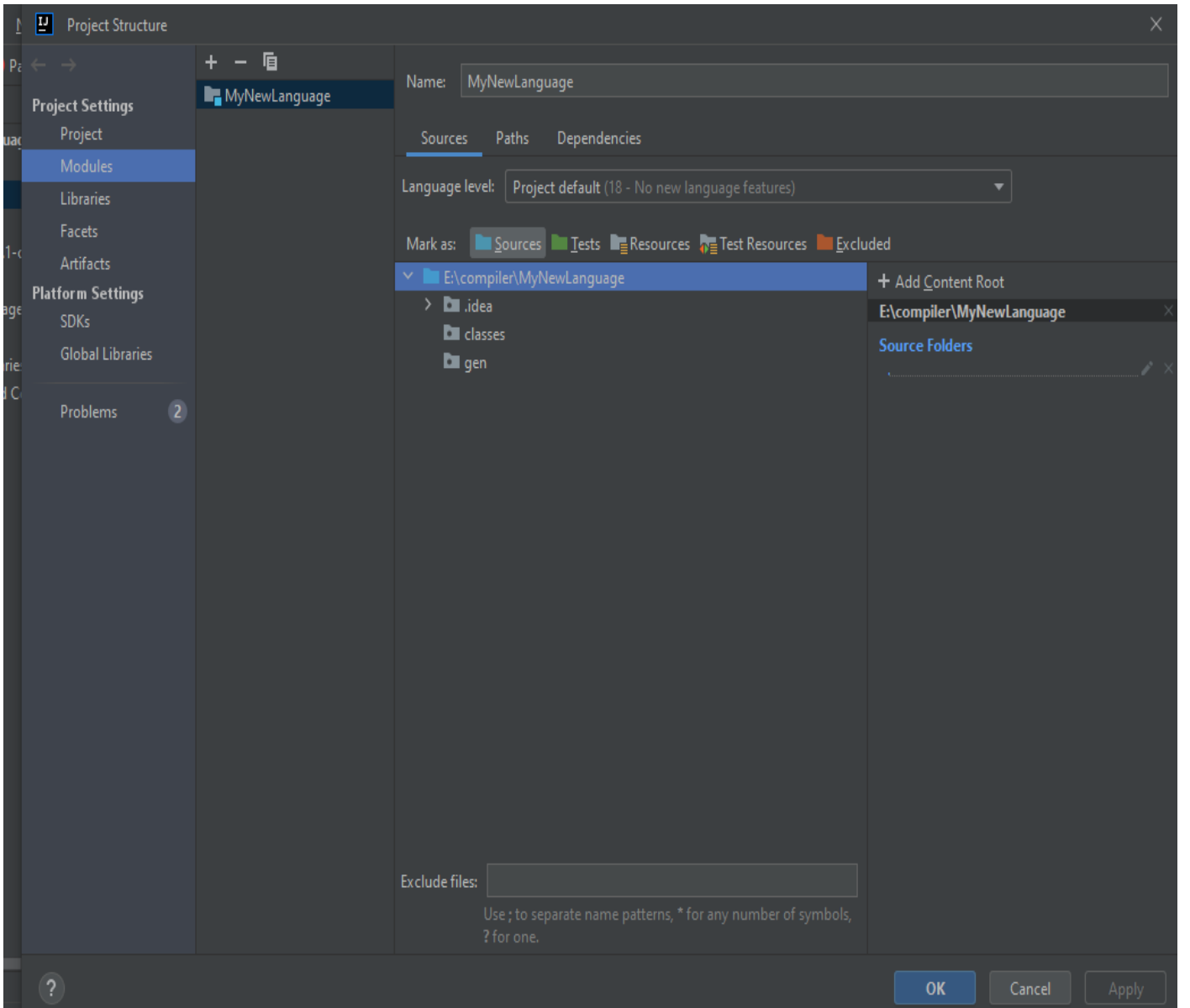
حيث تحتوي شجرة ال AST على قواعد اللغة بالإضافة للرموز الرئيسية فقط (التي تمثل قيمة عددية أو تعطي معنى منطقي) .

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 نسمة CLASSES سنضع داخله صفوف القواعد التي سنقوم بإنشائها .

نقوم بإنشاء الصف Page:

```
un Tools VCS Window Help MyNewLanguage - Page.java
ParserFile.g4 Page.java LexerFile.g4 MyLanguage.txt
1 package Classes;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 public class Page {
7     List<Page_Style> page_styleList = new ArrayList<Page_Style>();
8 }
9
```

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

```
ParserFile.g4 Page.java LexerFile.g4 MyLanguage.txt
1 package Classes;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 public class Page {
7     List<Page_Style> page_styleList = new ArrayList<Page_Style>();
8
9     public List<Page_Style> getPage_styleList() {
10         return page_styleList;
11     }
12
13     public void setPage_styleList(List<Page_Style> page_styleList) {
14         this.page_styleList = page_styleList;
15     }
16
17     @Override
18     public String toString() {
19         return "Page{" +
20             "page_styleList=" + page_styleList +
21             "}";
22     }
23 }
24
```

نعدّل قليلا على تابع ال toString لتحسين شكل الطباعة

```
@Override
public String toString() {
    return "\n Page{" +
        "\n " + page_styleList +
        "\n }";
}
```

نقوم بإنشاء الصف Page_Style

```
ParserFile.g4 x Page_Style.java x LexerFile.g4 x MyLanguage.txt x
La 1 package Classes;
2
3 public class Page_Style {
4     Name name;
5     Nav nav;
6     Footer footer;
7     |
8
9 }
10
```

نضغط alt + insert ونضيف getter and setters ثم نضغط مرة أخرى ونضيف toString

```
ParserFile.g4 x Page_Style.java x LexerFile.g4 x MyLanguage.txt x
3 public class Page_Style {
4     3 usages
5     Name name; Nav nav; Footer footer;
6     public Name getName() {
7         return name;}
8     public void setName(Name name) {
9         this.name = name;}
10    public Nav getNav() {
11        return nav;}
12    public void setNav(Nav nav) {
13        this.nav = nav;
14    }
15    public Footer getFooter() {
16        return footer;
17    }
18    public void setFooter(Footer footer) {
19        this.footer = footer;
20    }
21    @Override
22    public String toString() {
23        return "Page_Style{" +
24            "name=" + name +
25            ", nav=" + nav +
26            ", footer=" + footer +
27            '}';
28    }
}
```

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

```
7 page_style : (name|nav|footer) SEIMIE_QO ;|
```

لذلك ستكون قيمة القاعدتين الباقيتين null فلتجنب 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

```

package Classes;

public class Nav {

    3 usages
    String nav;

    public String getNav() {
        return nav;
    }

    public void setNav(String nav) {
        this.nav = nav;
    }

    @Override
    public String toString() {
        return "\nNav{" +
            "\n" + nav + '\'' +
            "\n}";
    }
}

```

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

```
nav = NAV;
```

- الآن نقوم بإنشاء الصف Footer

```
package Classes;

public class Footer {
    String footer;

    public String getFooter() {
        return footer;
    }

    public void setFooter(String footer) {
        this.footer = footer;
    }

    @Override
    public String toString() {
        return "\nFooter{" +
            "\n" + footer +
            "\n}";
    }
}
```

- الآن نقوم بإنشاء الصف Name

```
package Classes;

public class Name {
    Page_Name page_name;

    public Page_Name getPage_name() {
        return page_name;
    }

    public void setPage_name(Page_Name page_name) {
        this.page_name = page_name;
    }

    @Override
    public String toString() {
        if(page_name!=null)
            return "\nName{" +
                "\n" + page_name +
                "\n}";
        return "\nName{" +
            "\n" + page_name +
            "\n}";
    }
}
```

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

```
name :NAME OPEN_BRACKET QOUT page_name? QOUT CLOSE_BRACKET;
```


- الآن نقوم بإنشاء صف Page_Name

```
Run Tools VCS Window Help MyNewLanguage - Page_Name.java
ParserFile.g4 x Page_Name.java x Footer.java x Name.java x Page_Style.java x Nav.java x LexerFile.g4 x MyLanguage.txt x
1 package Classes;
2
3 public class Page_Name {
4
5     3 usages
6     String string;
7
8     public String getString() {
9         return string;
10    }
11
12    public void setString(String string) {
13        this.string = string;
14    }
15
16    @Override
17    public String toString() {
18        return "Page_Name{" +
19            "string='" + string + '\'' +
20            '}';
21    }
22 }
```

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 Mylanguage Lexer , 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

```

1 import Classes.Page;
2
3 import gen.MyLanguageParser;
4 import gen.MyLanguageLexer;
5
6 import org.antlr.v4.runtime.CharStream;
7 import org.antlr.v4.runtime.CommonTokenStream;
8 import org.antlr.v4.runtime.tree.ParseTree;
9 import java.io.IOException;
10 import static org.antlr.v4.runtime.CharStreams.fromFileName;
11
12 public class main {
13
14     public static void main(String[] args) throws IOException {

```

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

```

6 import org.antlr.v4.runtime.CharStream;
7 import org.antlr.v4.runtime.CommonTokenStream;
8 import org.antlr.v4.runtime.tree.ParseTree;
9 import java.io.IOException;
10 import static org.antlr.v4.runtime.CharStreams.fromFileName;
11
12
13
14 public class main {
15
16     public static void main(String[] args) throws IOException {
17
18         String source = "myLanguage.txt";
19         CharStream cs = fromFileName(source);
20         MyLanguageLexer lexer = new MyLanguageLexer(cs);
21         CommonTokenStream token = new CommonTokenStream(lexer);
22         MyLanguageParser parser = new MyLanguageParser(token);
23         ParseTree tree = parser.page();
24         Page doc = (Page) new BaseVisitor().visit(tree);
25         System.out.println(doc);
26     }
27 }

```

مسار الملف

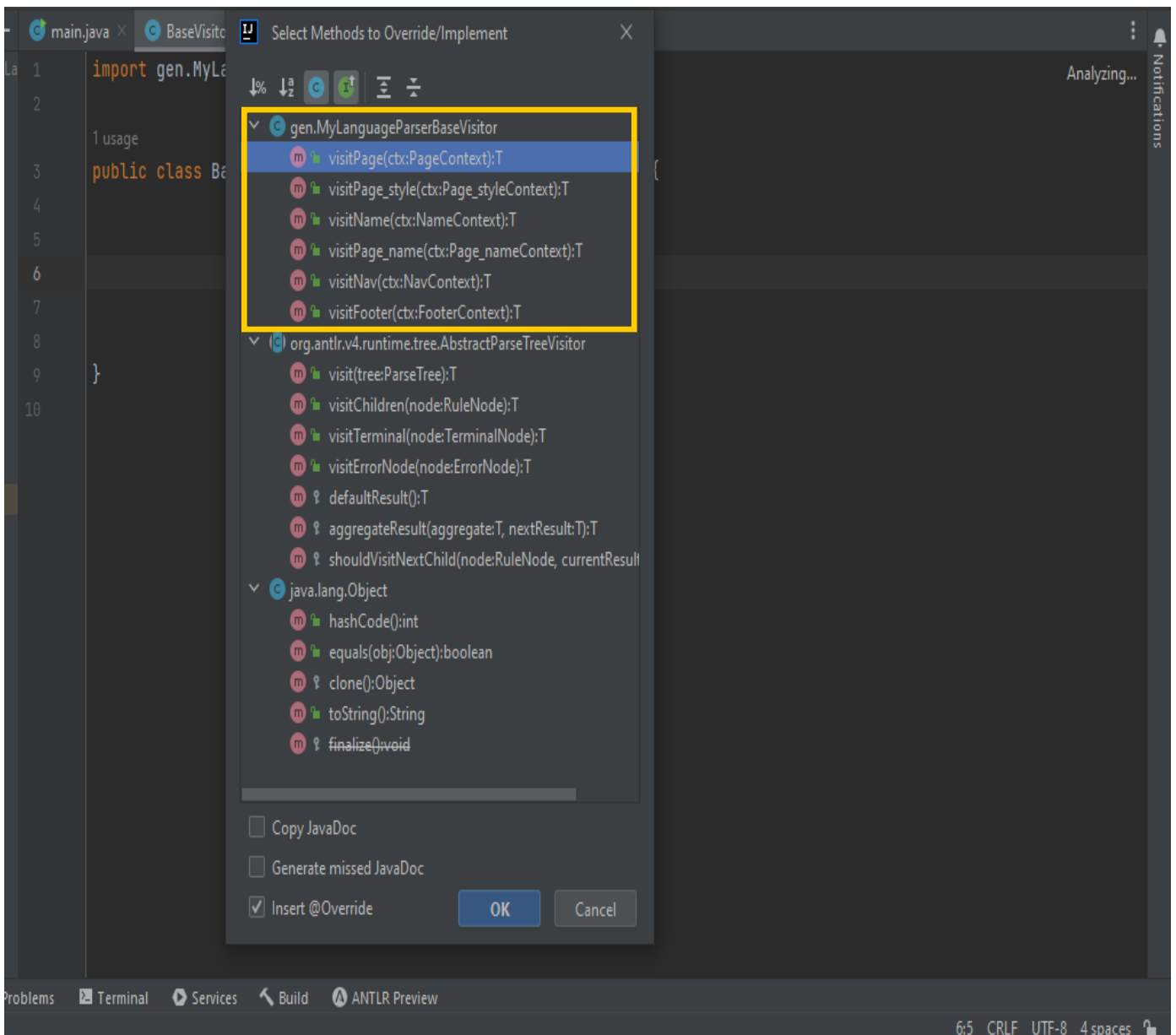
Parsing للملف

ابتداءً من القاعدة page

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

```
main.java x BaseVisitor.java x Page.java x
1 import gen.MyLanguageParserBaseVisitor;
2
3 1 usage
4 public class BaseVisitor extends MyLanguageParserBaseVisitor {
5
6 }
```

حيث أن هذا الصف سيرث visit لجميع قواعد اللغة من MyLanguageParserBaseVisitor بمجرد أن تضغط على CTRL + O ستظهر لك ال methods



- 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 .

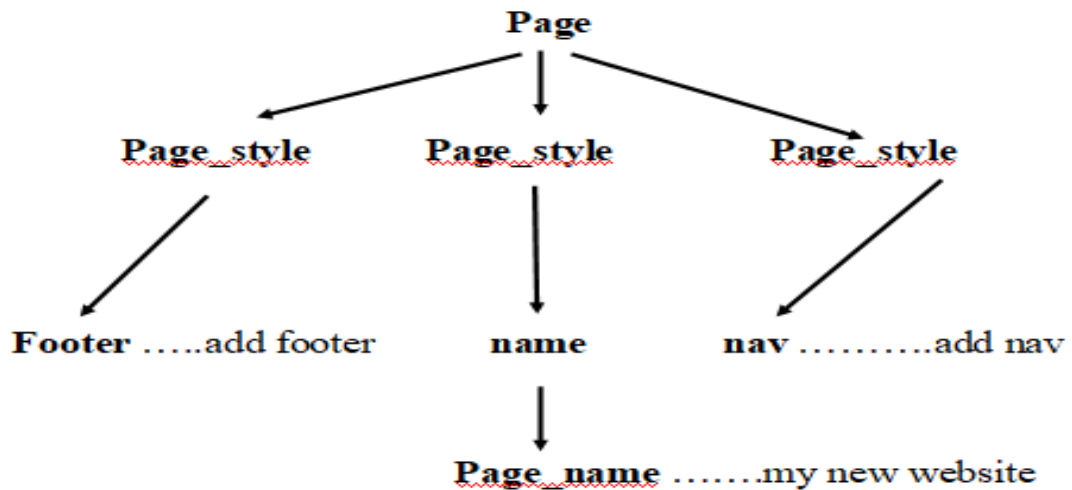
مثال توضيحي : عند كتابة hi{} في ملف MyLanguage.txt نجد إن hi{} واقعة ضمن القاعدة page في ملف ال Parser فإذا احتجنا لتخزين كلمة hi للتأكد من ورودها ، أو لاستخدامها لاحقاً في عملية توليد الكود ، نقوم بعمل override للتابع visitpage الذي نستطيع من خلاله الحصول على قيم ال Token الموجودة ضمن القاعدة page ومنها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

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_style و نحتاج لتخزين جميع ال

page_style الموجودة ضمن القاعدة page نقوم بعمل حلقة تقوم عند كل عملية مرور بال

pageStyle بعمل visitPage_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 ونقوم بعمل OverRide للتابع visitPage_style

- نقوم بتغيير نوع القيمة التي يرددها التابع إلى كائن من نوع Page_Style ثم نقوم بإنشاء كائن

page_style وعمل return له .

- ننتبه إلى أن القاعدة page_style تترد 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 ونقوم بعمل OverRide للتابع visitName

- نقوم بتغيير نوع القيمة التي يرددها التابع إلى كائن من نوع Name ثم نقوم بإنشاء كائن name وعمل

return له .

- ننتبه إلى أن القاعدة name تترد 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 يرددها 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 يرددها 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

```
Page{
  [
    Page_Style{
      Name{
        Page_Name{string='my new website'}
      }
    },
    Page_Style{
      Nav{
        add nav ' '
      }
    },
    Page_Style{
      Footer{
        add footer ' '
      }
    }
  ]
}
```

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



Sami kazah

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 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))); 
```

Sami kazah

الفصل الرابع:
بناء المترجم (٣)

Sami kazah

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 بالمرحل 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 الإجراءات
 - 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 Value + Type .

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

```
BaseVisitor.java x Row.java x MyLanguageParser.g4 x main.java x
1
2
3 public class Row {
4     String type;
5     String value;
6
7     public String getType() {
8         return type;
9     }
10
11    public void setType(String type) {
12        this.type = type;
13    }
14
15    public String getValue() {
16        return value;
17    }
18
19    public void setValue(String value) {
20        this.value = value;
21    }
22 }
23
```

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

```
visitor.java x Row.java x SymbolTable.java x MyLanguageParser.g4 x main.java x
import java.util.List;

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

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

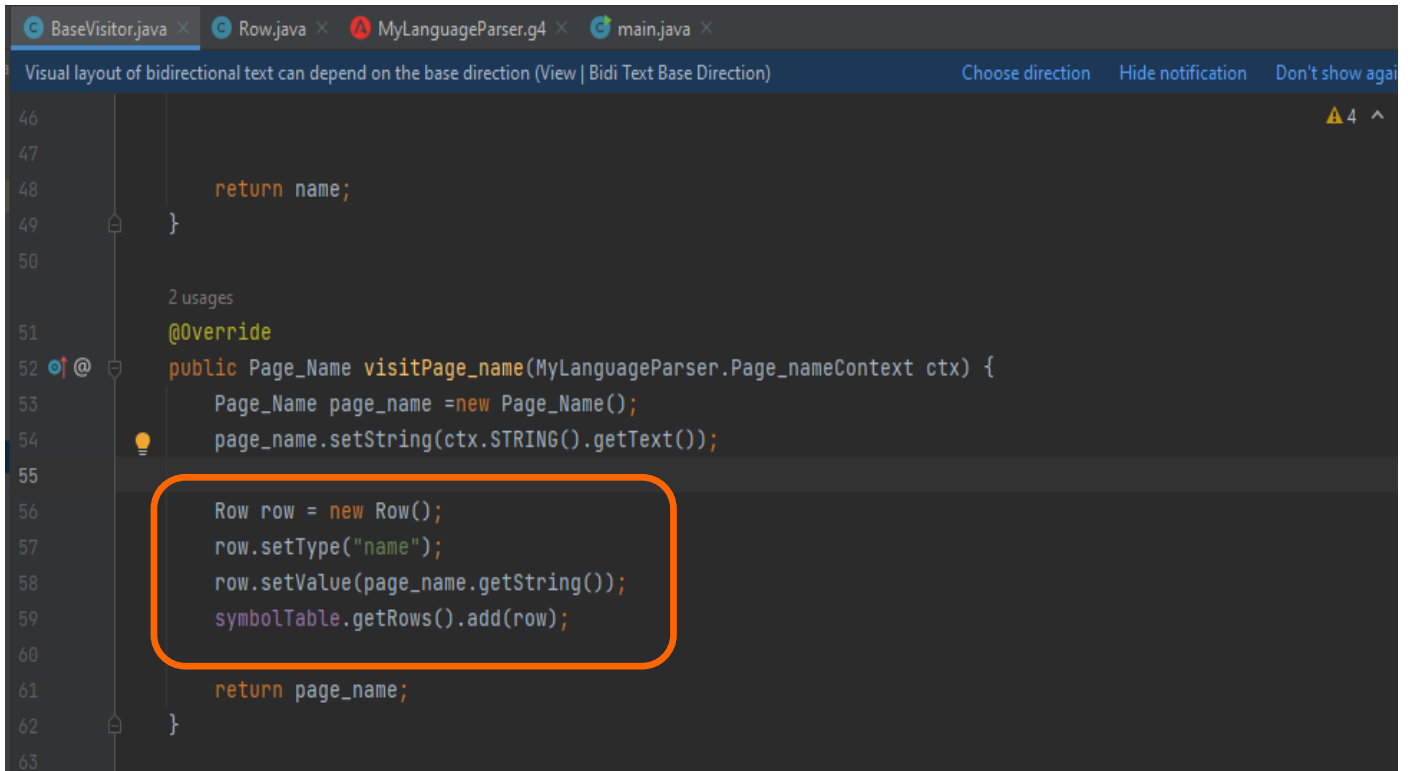
    public void setRows(List<Row> rows) {
        this.rows = rows;
    }

    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" + rows.get(i).getValue());
            }
        }
    }
}
```

• نذهب إلى ال Base Visitor ونقوم بتعريف SymbolTable

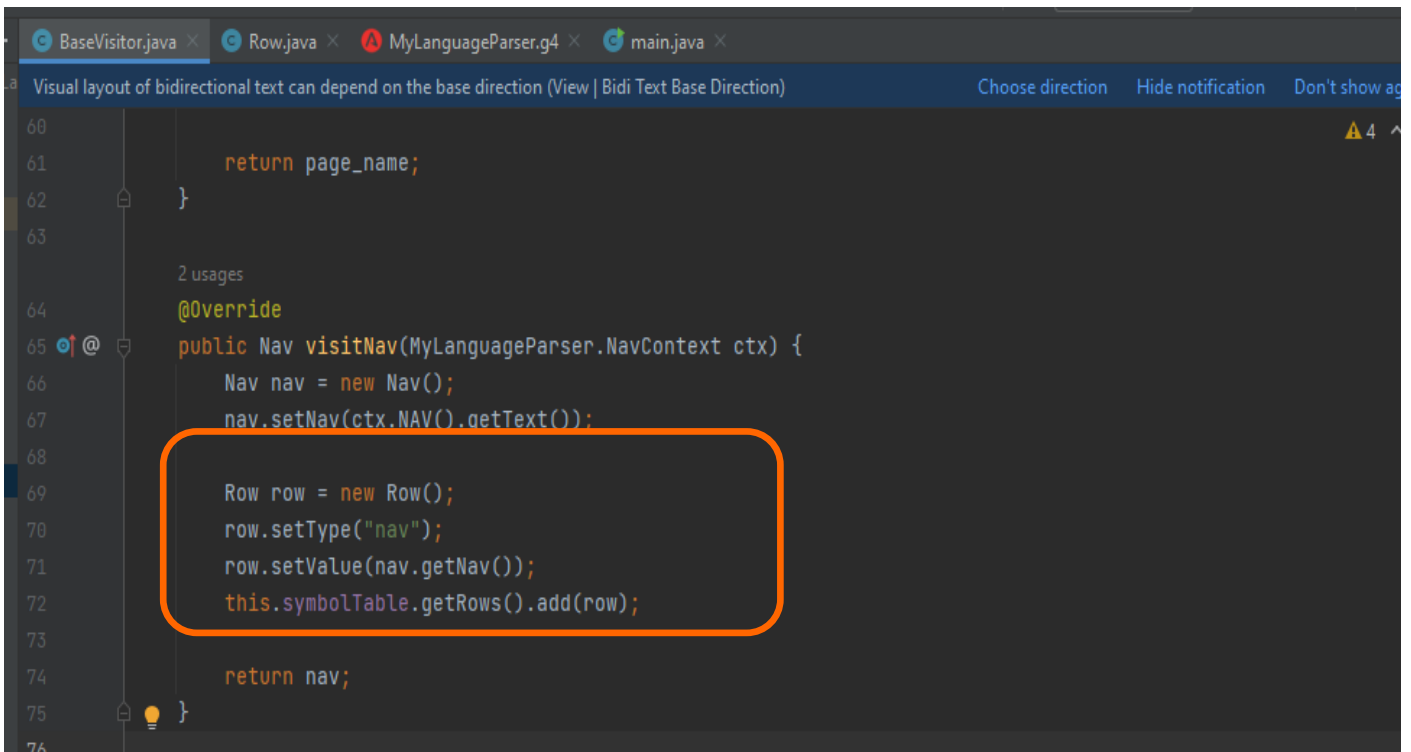
```
BaseVisitor.java x Row.java x SymbolTable.java x MyLanguageParser.g4 x main.java x
2 import gen.MyLanguageParser;
3 import gen.MyLanguageParserBaseVisitor;
4
1 usage
5 public class BaseVisitor extends MyLanguageParserBaseVisitor {
6
7     SymbolTable symbolTable = new SymbolTable();
8     Row row = new Row();
9
10
11
12 1 usage
13 @Override
14 public Page visitPage(MyLanguageParser.PageContext ctx) {
15     Page page = new Page();
16     for (int i = 0; i < ctx.page_style().size(); i++) {
17         if(ctx.page_style(i)!=null)
18         {
19             page.getPage_styleList().add(visitPage_style(ctx.page_style(i)));
20         }
21     }
22     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 :



```
46
47
48     return name;
49 }
50
51 2 usages
52 @Override
53 public Page_Name visitPage_name(MyLanguageParser.Page_nameContext ctx) {
54     Page_Name page_name = new Page_Name();
55     page_name.setString(ctx.STRING().getText());
56     Row row = new Row();
57     row.setType("name");
58     row.setValue(page_name.getString());
59     symbolTable.getRows().add(row);
60
61     return page_name;
62 }
63
```

Samir Kazan



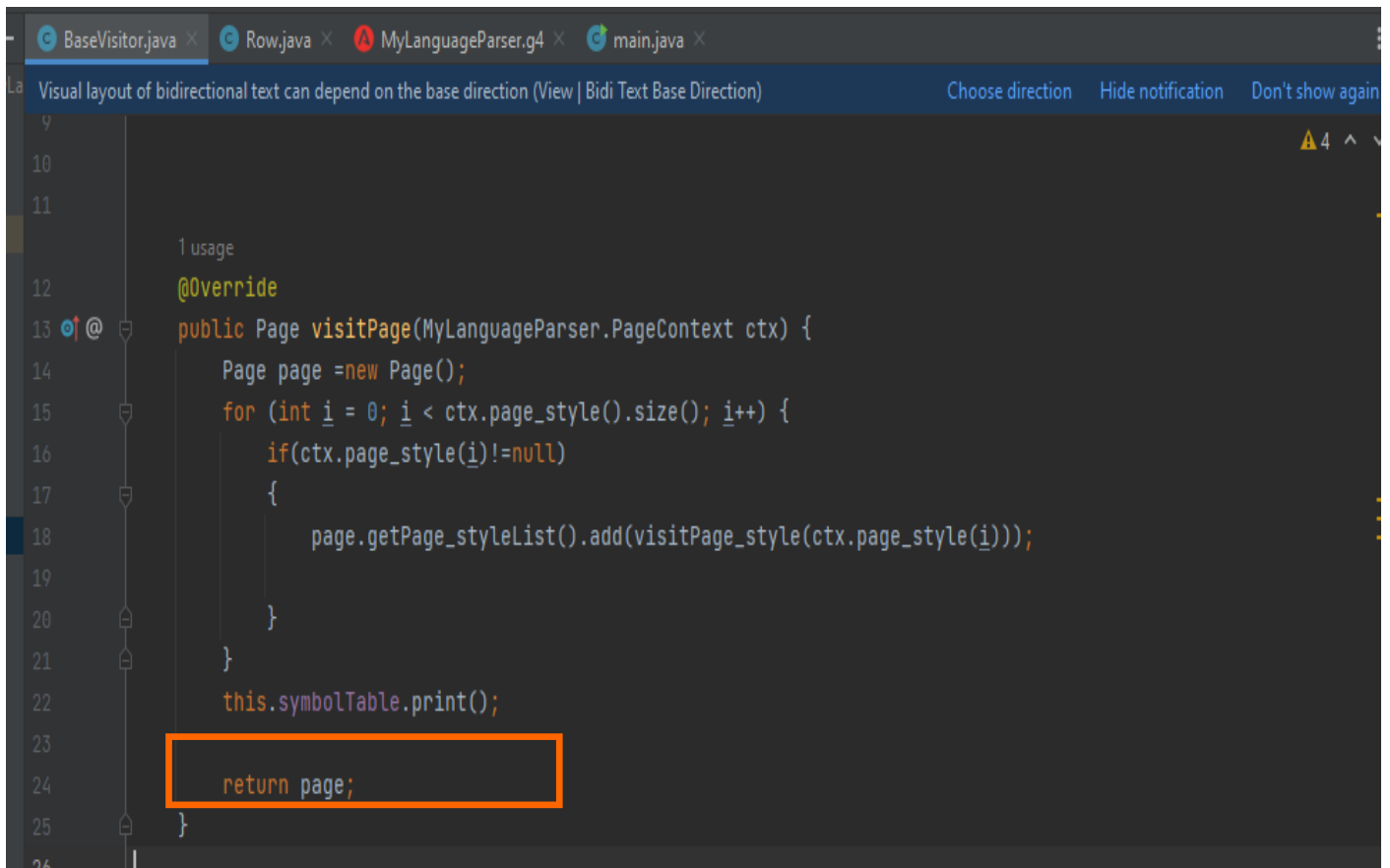
```
60
61     return page_name;
62 }
63
64 2 usages
65 @Override
66 public Nav visitNav(MyLanguageParser.NavContext ctx) {
67     Nav nav = new Nav();
68     nav.setNav(ctx.NAV().getText());
69     Row row = new Row();
70     row.setType("nav");
71     row.setValue(nav.getNav());
72     this.symbolTable.getRows().add(row);
73
74     return nav;
75 }
76
```

```
BaseVisitor.java x Row.java x MyLanguageParser.g4 x main.java x
Visual layout of bidirectional text can depend on the base direction (View | Bidi Text Base Direction) Choose direction Hide notification Don't show again
72     this.symbolTable.getRows().add(row);
73
74     return nav;
75 }
76
77 2 usages
78 @Override
79 public Footer visitFooter(MyLanguageParser.FooterContext ctx) {
80     Footer footer =new Footer();
81     footer.setFooter(ctx.FOOTER().getText());
82
83     Row row = new Row();
84     row.setType("footer");
85     row.setValue(footer.getFooter());
86     this.symbolTable.getRows().add(row);
87
88     return footer;
89 }
90
91
```

```
BaseVisitor.java x MyLanguageParser.g4 x main.java x MyLanguageParserListener.java x
Visual layout of bidirectional text can depend on the base direction (View | Bidi Text Base Direction) 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);
    }

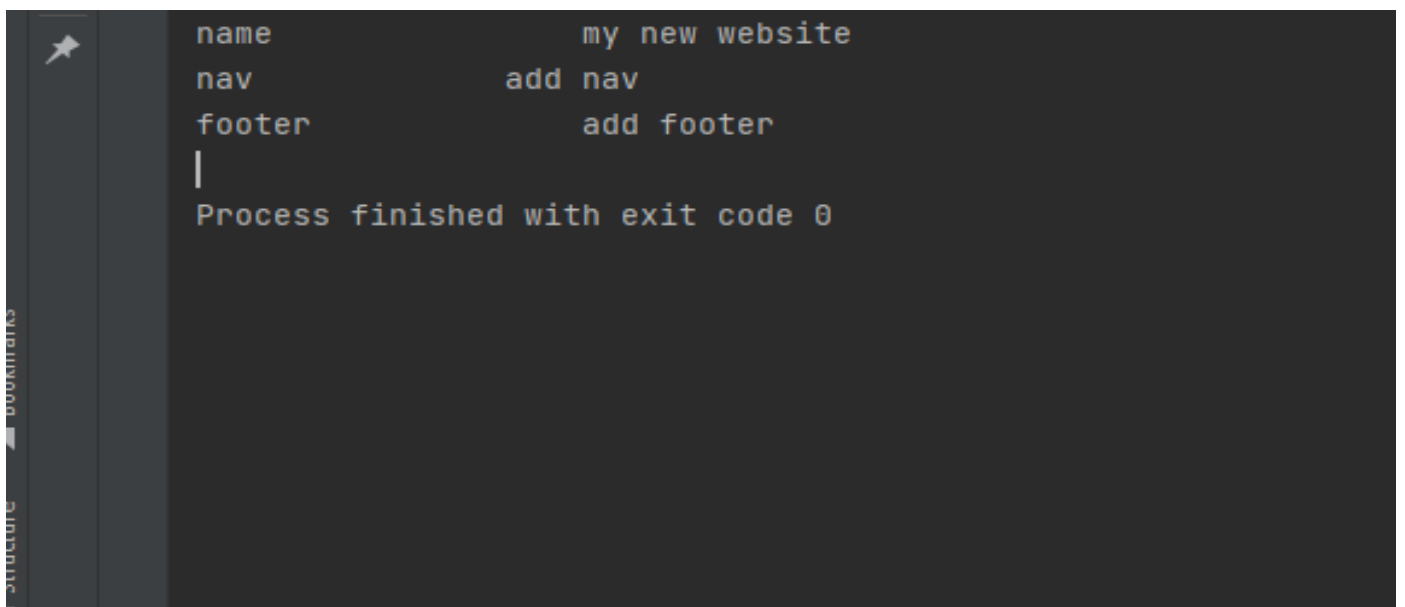
    return name;
}
```

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



```
1 usage
12 @Override
13 public Page visitPage(MyLanguageParser.PageContext ctx) {
14     Page page = new Page();
15     for (int i = 0; i < ctx.page_style().size(); i++) {
16         if (ctx.page_style(i) != null)
17         {
18             page.getPage_styleList().add(visitPage_style(ctx.page_style(i)));
19         }
20     }
21     this.symbolTable.print();
22     return page;
23 }
```

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();
```

Sami kazah

Sami kazah

Sami kazah

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 معرفاً .

```
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 كبراميتر .

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 .

```
1 usage
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;
}
1 usage
```

- 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();
    SemanticCheck 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 pages
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" +
        "                \n" +
        "            </nav>\n" +
        "        </div> \n" +
        "    \n" +
        "    <!-- Image menu in Header to contain an Image\n" +
        "        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/\n" +
        "\">About Us</a>|\n" +
        "        <a href=\"\n" +
        "\"https://www.geeksforgeeks.org/privacy-policy/\n" +
        "\">Privacy Policy</a>|\n" +
        "        <a href=\"\n" +
        "\"https://www.geeksforgeeks.org/careers/\n" +
        "\">Careers</a>\n" +
        "    \n" +
        "    \n" +
        "    \n" +
        "<p>@geeksforgeeks, Some rights reserved</p>\n" +
        "    \n" +
        "    \n" +
        "    \n" +
        "</footer>";
}

```

```

public void generate (SymbolTable symbolTable) {
    PrintWriter writer;
    boolean isFooter = false;
    this.Nav = navCode();
    this.Footer = footerCode();
    this.FileName = symbolTable.rows.get(0).getValue().replaceAll( regex: " ", replacement: "");
    try {
        writer = new PrintWriter( fileName: this.FileName+".html");
    } catch (FileNotFoundException e) {
        throw new RuntimeException(e);
    }
    for (int i = 0; i < symbolTable.rows.size(); i++) {
        if(symbolTable.rows.get(i)!=null)
        {
            if(symbolTable.rows.get(i).getType().contains("footer"))
            {
                isFooter = true;
                break;
            }
        }
    }
    if(!isFooter){ this.Footer = "";}
    writer.println("<html> " +this.Nav+ "\n\n" + this.Footer + "</html>");
    writer.close();
}

```

- 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 .


```

mynewwebsite.html ×
E > compiler > MyNewLanguage > mynewwebsite.html > ...
1 <html> <header>
2 <!-- Top header menu containing
3 | logo and Navigation bar -->
4 <div id="top-header">
5
6 <!-- Logo -->
7 <div id="logo">
8
9 </div>
10
11 <!-- Navigation Menu -->
12 <nav>
13
14 </nav>
15 </div>
16
17 <!-- Image menu in Header to contain an Image and
18 | a sample text over that image -->
19 <div id="header-image-menu">
20
21 </div>
22 </header>
23
24 <body> <footer>
25
26 | <a href=
27 | "https://www.geeksforgeeks.org/about/">About Us</a>|
28 | <a href=
29 | "https://www.geeksforgeeks.org/privacy-policy/">Privacy Policy</a>|
30 | <a href=
31 | "https://www.geeksforgeeks.org/careers/">Careers</a>
32
33
34
35 <p>@geeksforgeeks, Some rights reserved</p>
36
37
38
39 </footer> </body> </html>
40

```

<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://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

