باسمه تعالی

پروژه انتلر درس نظریه زبانها و ماشینها دکتر سوادی

نیمسال اول سال تحصیلی 1401-1402

هدف کلی پروژه : آشنایی و کار با گرامر ها و نحوه گرامر نویسی و طراحی قوانین parser و lexer به کمک ابزار ANTLR ، آشنایی بیشتر با فرآیند parsing و رسم parse tree ، آشنایی با نحوه عملکرد یک language recognizer ساده و ابتدایی(که بخشی از یک کامپایلر را شامل میشود).

شرح پروژه : در این پروژه شما باید با کمک یک گرامر مناسب و پیاده سازی قوانین lexer و parser مربوطه، به طراحی ساختار یک زبان جدید و ساده برنامه نویسی بپردازید و گرامر شما (در واقع language recognizer حاصل از گرامر) بتواند رشته های ورودی مطابق با ساختار زبان مدنظر را پذیرفته و عمل parsing و رسم درخت پارس را به درستی انجام دهد و نیز رشته های ورودی که مربوط و منطبق بر ساختار مدنظر نیست را پذیرش نکند. توضیحات مربوط به ساختار زبان برنامه نویسی مدنظر در ادامه مطرح میگردد.

قوانین و قرارداد های زبان:

- دقت فرمایید در ادامه توضیحات، تمام کلمات کلیدی زبان (keyword ها) با رنگ قرمز مشخص خواهند شد. دقت کنید که keyword ها به صورت case-sensitive یا حساس به کوچک و بزرگی حروف نیستند (مثلا class و CLASS و Class همگی به عنوان کلمه کلیدی مربوط به کلاسها باید شناسایی و پذیرش شوند). مواردی که وجود آنها اختیاری است در syntax دستورات در [ ] قرار گرفته اند.
- ذکر کاراکتر semicolon (;) در انتهای دستورات زبان اختیاری است(به جز دستوراتی که در syntax آنها به طور جداگانه و مشخص، ذکر میگردد). یعنی هر دستور در این زبان برنامه نویسی می تواند به ; ختم شود و یا نشود. اما باید امکان وجود هر دو حالت پیاده سازی شود.

- دستورات اضافه کردن کتابخانه ها و اجزای آنها باید قبل از تمامی دستورات دیگر برنامه و به شکل زیر پیاده سازی شوند:
  - **import::from(<library_name>, <class_name> [, <class_name] …) .;;**

or

  - **import : : from(<library_name>, *) . ; ;**

for example:

- **import : : from(math, random , floor) . ;;**
- **import::from(io,*).;;**

- تعریف کلاس ها به صورت زیر خواهد بود:
  - **[DirectAccess | inDirectAccess | restricted] <class_name> CLASS**
    [Inherited from <parent_class_name> [,<parent_class_name>] …]
    [implements <interface_name> [, <interface_name>] …]
    BEGIN
    <class_body>
    END
    [;]
- نام کلاس های تعریف شده توسط کاربر ،حتما باید با حروف بزرگ آغاز شود(یعنی مثلا نباید نام کلاسی به صورت "student" بلکه حتما باید به شکل "Student" یا "STUDENT" یا "StuDent" یا … باشد و حرف اول نام آن کلاس باید capital باشد). دقت کنید که ممکن است در بخشی هایی از توضیحات ذکر شده این امر **سهواً** رعایت نشده باشد اما شما باید به این موضوع دقت فرمایید و فرض را بر همین قرارداد فوق بگذارید.
- کلمات کلیدی DirectAccess , inDirectAccess , restricted نشان دهنده ی سطوح دسترسی هستند و معادل public, protected, private در زبان های برنامه نویسی معمول میباشند.

For example:

- **inDirectAccess Student CLASS Inherited from Human implements listening , reading,teaching BEGIN**
  var sID : integer ; DirectAccess FUNC getSID() returns integer

END;

- تعریف توابع به صورت زیر خواهد بود :

- **[DirectAccess | inDirectAccess | restricted] [static] FUNC <function_name> ([<parameter_1_name> <parameter_1_type> ;] [<parameter_2_name> <parameter_2_type> ;] [var <parameter_3_name> : <parameter_3_type> ;] [var <parameter_4_name> : <parameter_4_type> ;]…) returns <data_type> { <function_body> [return (<desired_value_in_defined_data_type>);] }[;]**

For example:

- **FUNC myFunc (a integer ;b integer ; var c : float)returns void { c <- a + b }**
- **inDirectAccess FUNC myFunc2(a integer;b integer)returns integer**
- **{return(a + b);}**

- تعریف رویه ها به صورت زیر خواهد بود:

- **[DirectAccess | inDirectAccess | restricted] [static] Proc <procedure_name> (<procedure_body>) ;**

For example:

- **inDirectAccess static Proc printHelloWorld (**
- **print(‘*’) counter++ print(‘helloworld’) counter++; printLine(‘*’)) ;**

- تعریف متغیرها (و خصیصه ها) به شکل زیر است:

- **[DirectAccess | inDirectAccess | restricted] [constant] var <variable_name> : <variable_data_type> [= <initial_value>] [;]**

For example:

- **restricted constant var myVar : integer = 0x0809A1F**
- **DirectAccess var i , j : float = 3.1e23 ,0.873 ;**
- **var s:string**

  - قوانین نامگذاری متغیرها به شرح زیر است:

    - نام تمامی متغیرها و خصیصه ها باید با یک حرف کوچک لاتین شروع شود.

- طول نام متغیرها حداکثر 30 کاراکتر است
- نام متغیر ها میتواند تنها شامل حروف کوچک و بزرگ لاتین، ارقام و کاراکتر underline باشد.
- نام متغیر ها نمی تواند همانند کلمات کلیدی زبان باشد.(نمی توانیم متغیری با نام if , while و ... داشته باشیم)

- تعریف آرایه ها به شکل زیر مطلوب است:

- **[DirectAccess | inDirectAccess | restricted] [constant] var[[0 : <desired_length>]] <array_name> : <elements_data_type> [= {[<first_element_value>, <second_element_value> , ...]}] [;]**

For example:

- **var[0:9] myArray : Boolean ;**
- **restricted**
  var[] Array3 : integer = {[10245, 0x34b21D9 , 101001110101b]}

- تعریف ارجاعات(reference ها) به صورت زیر مد نظر است:

- **[DirectAccess | inDirectAccess | restricted] <class_name> <reference_name> -> (new <class_name>([<parameters>]) | null)**

for example:

- **indirectaccess Circle c1 -> new Circle();**
- **Rectangle r2 -> new Square(x1,y1,45)**
- **music[] album -> new pop[0:12]();**
- **Point p -> NULL**

- دستورات حلقه for به اشکال زیر مد نظر است:

- **for ([[DirectAccess | inDirectAccess | restricted] [constant] var <variable_name> : <variable_data_type> [= <initial_value>]][; <condition>][; increment | decrement]) { <more_than_one_statements> } [;]**
- **for ([[DirectAccess | inDirectAccess | restricted] [constant] var <variable_name> : <variable_data_type> [= <initial_value>]] [; <condition>] [; increment | decrement]) [{] <one_statement_at_last> [}] ;**

- **for every [DirectAccess | inDirectAccess | restricted] var <variable_name> in <iterative_list_or_array_name> do <statements> until end**

for example:

- **for ()\{\};**
- **for() { a++ ; };**
- **for() a++ ;**
- **for(var i : float = 9e-3);**
- **for(var i:integer= -100;i++)\{a++ b++ c++ \}**
- **for(directaccess constant var k : float ; k < 10);**
- **for every var k in myArray do y <-k.getSmth() until end**
- **for every var k in myArray do j++ a-- y <-k.getSmth() print(y) until end**
- **for every var k in myArray do p++ k.set(p); a-- b <- a ; i-- until end**

- دیگر دستورات حلقه به اشکال زیر مد نظر است:

- **while ([<conditions>]) begin [<statements>] END**
- **do [<statements>] as_Long_As <conditions> ;**

for example:

- **while () begin end**
- **while (true) begin i++ end;**
- **do k-- as_long_as k > 0 ;**

- دستورات شرطی به شکل زیر مطلوب است:

- **if { <conditions> } then [<statements>] ; [elif (<conditions>) do [<statements>] END] [else do as follow [<statements>] stop]**
- **<condition> ? <statements> : <statements> [;]**

for example:

- **if {true } then;**
- **if{ a==b}then print(“equal”) ; elif (a > b)**

do end

- **if**
  { a==b}
  then;elif (a > b) do print(“a is greater”) end else do as follow print(“b is greater”) stop
- **true ? print(1) : print(0)**

- دستورات switch-case به شکل زیر مطلوب است:

- **switch <expression> match**
  case <first_case_relevant_value> : [<statements>] [break;]
  [case <second_case_relevant_value> : [<statements>] [break;]]
  …
  [default <default_case_relevant_value> : [<statements>]] end

for example:

- **switch a match case 0: case 1 : break; case 2 : print(2) break; end**

- دستورات exception به شکل زیر خواهد بود:

- **try {{ [<statements>] }} catch ([<error_class_name> <error_name>] [, <error_class_name> <error_name>]...) begin [<statements>] end**

for example:

- **try {{ a/b}} catch (DivideExcep dividedByZero) begin print(“error happened”) end**

- دستورات انتساب به شکل زیر مدنظرند(به جز حالت انتساب اولیه متغیرها در هنگام تعریف آنها):

- **<variable_name> <- <value> [;]**

for example:

- **a <- 123**
- **b <- null;**
- **restricted var f : string = “hello”**

- کامنت گذاری به صورتهای زیر انجام میشود و دقت کنید که کامنت ها نباید در درخت پارس ترسیم شوند:

- **-- <single_line_comment>**
- **!# <multi_line_comment> !#**

for example:

- **-- this a comment**
- **!# this**
  is
  a comment !#

- انواع داده ها به صورت زیر هستند:

- integer :

- نشان دهنده ی نوع داده ی عدد صحیح می باشد که می تواند در سه مبنای دهدهی، دودویی و مبنای شانزده به صورت زیر قرار گیرد(فاصله ی بین ارقام و حروف در هیچ مبنایی مجاز نیست):
  - decimal : 1, 2, 3, -100 ,0, 45 ,....
  - hexadecimal : 0(x | X)(0 | 1 | 2 … | 9 | A | a | B | b … | F |f)(0 | 1 | 2 … | 9 | A | a | B | b … | F |f)....
  - binary : (0 | 1)(0 | 1) … (0 | 1)(b | B)

- float :

- نشاندهنده ی اعداد اعشاری است و فقط می توانند به صورت نماد عملی مورد استفاده قرار گیرد یعنی به شکل زیر:
  - X.YYYYY....[e][-][ZZZ...]
  - مثلا:
  - 3.6789 یا 1.45678e4 (= 14567.8) یا 0.89e-3 (= 0.00089)

- char :

- نشان دهنده ی کاراکتر است و همانطور که در دیگر زبان های برنامه نویسی معمول تعریف شده است ، در اینجا نیز استفاده میشود.

- Boolean :

- نشان دهنده ی مقدار بولی true یا false

- string:

- نشان دهنده ی رشته ای از کاراکتر هاست و همانطور که در دیگر زبان های برنامه نویسی معمول تعریف شده است ، در اینجا نیز استفاده میشود.

- null:

- نشان دهنده ی هیچی است و همانطور که در دیگر زبان های برنامه نویسی معمول تعریف شده است ، در اینجا نیز استفاده میشود.

- انواع عملگر های مورد استفاده در زبان که باید تعریف شده و به ترتیب ذکر شده زیر شناسایی و مورد استفاده قرار گیرند، عبارتند از:
  - ( )
  - ^ (توان)
  - عملگر های یگانه یا unary:

- این عملگرها فقط روی یک عملوند، عمل میکنند و دقت داشته باشید که **نباید** میان عملوند و این عملگرها ، فاصله ای وجود داشته باشد.
  - **#++ #-- ++# --#**
  - **-**
  - **>> << (شیفت)**
  - *** / // %**
  - \- +
  - **&&**
  - **& and**
  - **||**
  - **| or**
  - **== !=**
  - **<= >=**
  - **< >**
  - **-= +=**
  - **/= *=**

توضیحات تکمیلی:

- همانطور که واضح است زبان تعریف در این پروژه زبانی با سینتکس جدید اما ترکیبی از دیگر زبانهای معروف و معمول برنامه نویسی است و بر پایه همان مفاهیم مشترک زبان های برنامه نویسی شناخته شده عمل میکند.
- همچنین بدیهی است این زبان از شیئ گرایی پشتیبانی میکند و از مفاهیم آن استفاده میکند.
- پیاده سازی موارد ذکر شده در این فایل برای تحویل پروژه کافی است. اما در صورتی که موارد دیگر و بیشتری به طور منطقی و منطبق و مرتبط با موارد ذکر شده به ساختار این زبان اضافه کنید، تا حد امکان نمره اضافه برای شما لحاظ خواهد شد.
- پیشنهاد میشود برای راحتی کار بیشتر در گرامر نویسی، با تحقیق و مراجعه به اینترنت و منابع مختلف، در مورد نحوه عملکرد عملگر "?" و نیز کاربرد عبارت "fragment" در گرامر نویسی با ابزار انتلر، اطلاعات کافی کسب کرده و از این دو مورد جهت پیاده سازی راحت تر گرامر خود استفاده بفرمایید.
- اگر در موارد و بخش هایی از زبان و ساختار آن ابهام وجود دارد، می توانید به دلخواه خود اما به صورت منطقی و منطبق بر موارد ذکر شده ، آن موارد را تعریف و پیاده کنید.

- لازم به ذکر است رعایت موارد و اصولی که در ویدیوهای آموزش انتلر در مورد نحوه گرامر نویسی به طور مفصل مورد بحث قرار گرفت، حائز اهمیت است و ممکن است در صورت رعایت نکردن آنها بخشی از نمره مربوط به پروژه را از دست بدهید.
- جدا کردن فایل قوانین lexer و parser مانعی ندارد و می توانید تمام گرامر و قوانین را در یک فایل تعریف و پیاده سازی کنید (یک فایل g4.) و یا آنها در چندین فایل مجزا و مرتبط قرار دهید.
- حتما چندین رشته ورودی مختلف برای تست کردن گرامر خود و بررسی تمامی بخش های ذکر شده در ساختار زبان در نظر بگیرید. به عبارتی دیگر این رشته های ورودی حکم تست کیس های گرامر نهایی شما را خواهند داشت و باید به خوبی و به اندازه کافی تمامی بخش های مرتبط را گرامر رو مورد بررسی و آزمایش قرار دهند و صحت کامل تمام بخش های گرامر و قوانین شما و قابلیت های مختلف آن را نشان دهند. همچنین باید چندین تست کیس نشان دهنده ی عدم پذیرش ورودی های غیر مرتبط با زبان مطرح شده و ساختار آن نیز در نظر گرفته شود. می توانید این تست کیس ها(رشته های ورودی) را در فایل های با پسوند txt. و یا پسوند doc. قرار دهید و باید همه ی آنها را در کنار فایل(ها) گرامر و قوانین خود به صورت یک فایل zip. درآورده و آنرا با نام studentID_antlr_project.zip در سامانه آموزش مجازی vu بارگذاری و تحویل دهید. در صورت عدم وجود تست کیس های کافی و مناسب بخشی از نمره کسر خواهد شد.
- دقت فرمایید که باید تسلط کافی بر گرامر خود داشته باشید و در صورت نیاز بتوانید در هنگام تحویل پروژه ، بخشهایی از گرامر را تغییر دهید و به شکل مطلوب دیگری در آورید. لازم به ذکر است تسلط بر پروژه بخش قابل توجهی از نمره نهایی آن را تشکیل میدهد(شاید حتی تا 50٪ نمره) و در صورت عدم تسلط کافی ممکن نمره صفر برای شما ثبت گردد.
- همچنین لازم به ذکر است پروژه باید به صورت انفرادی انجام شود و در صورت محرز شدن هر گونه تخلف و تقلب، برای فرد یا افراد مربوطه نمره 100- ثبت خواهد شد.

با آرزوی موفقیت و سلامتی