پروژه شماره ۱ درس نظریه زبان‌ها و ماشین‌ها

گرامر نویسی با ANTLR

شرح پروژه:

در این پروژه هدف پیاده‌سازی ساختار کلی یک زبان ساده برنامه‌نویسی و رسم درخت کدهای نوشته‌شده به این زبان است.

ساختار زبان:

۱. هر برنامه شامل یک یا چند کلاس و دستور **require** است. دستورات **require** پیش از تعریف کلاس‌ها می‌آیند.
۲. هر کلاس می‌تواند شامل توابع و تعریف متغیر باشد.
۳. در انتهای هر دستور باید semicolon (;) قرار داده شود.
۴. بلاک‌ها با **begin** و **end** تعریف می‌شوند.
۵. قوانین نام‌گذاری:
   - حداقل شامل دو کاراکتر هستند.
   - با رقم یا ‘_’ شروع نمی‌شوند.
   - متشکل از حروف کوچک و بزرگ لاتین، ارقام و کاراکترهای ‘$’ و ‘_’ هستند.
   - کلمات مشخص شده به صورت **bold** در این فایل، کلمات کلیدی هستند و نمی‌توانند نام متغیر باشند.
۶. قبل از تعریف متغیرها و توابع در کلاس، می‌توان سطح دسترسی را با **public** یا **private** مشخص نمود.
۷. کامنت‌گذاری به دو صورت single-line و multi-line تعریف می‌شود و کاراکتر مشخص کننده این کامنت‌ها را می‌توانید به دلخواه مشخص کنید. خطوط کامنت نباید در درخت ترسیم شوند.

در ادامه، ساختار بعضی از دستورات و بلوک‌ها به همراه مثال آورده شده است. در موارد زیر، قسمت‌های درون [ ] دلخواه هستند و ممکن است در کد مربوط به آن آمده باشند یا نه.

- استفاده کردن از کتابخانه‌ها و اجزای آن ها:

```
//  for using math library
lib1 = require <math>;


// for using floor from math library
func1 = from <math> require <floor>;


// for using floor from math library (alternative way)
func2 = from <math> => <floor>;


// for using floor and rand from math
func1,func2 =  from <math> require <floor> , from <math> require <rand>;
```

- تعریف متغیر:

```
[public/private][const] <datatype> <name>  [= <initial_value>];
---------------------------------------------------------------

const string myConst = “Lorem Ipsum”;      // defining constants
int myVar = 25, myVar2 = 56;               // defining two variables
int myArray[ ] = new int[4];                    // array definition
```

```
float myInitiatedArray[ ] = [0.05, 42, 42.25, 43];

private bool flag;
```

اعداد هنگام انتساب اولیه می‌توانند به صورت نماد علمی نیز وارد شوند (بـرای مثـال: 1.1209e-19 یـا 0.47) دقت کنید که نماد علمی حداکثر یک رقم قبل از ممیز دارد.

انواع data type را می‌توانید به دلخواه خود تعریف کنید، اما کد شما باید قابلیت تشخیص انواع اساسـی ماننـد **char**، **int**، **double**، **string**، **bool** و ... را داشته باشد.

- تعریف کلاس و instantiation:

```
class className [(<class_name1>)] [implements <variable_name> , <variable_name> ] begin

        <class_body>

end
```

```
// example

class Dot() implements Movable , Plottable, Euclidean begin

        private int px, py;

        // constructor function with no return type

        Dot (int px, int py) begin

                this.px = px;

                this.py = py;

        end
```

```
        // function with an optional parameter 'positive'
        int moveHorizontal (int step, bool positive = true) begin
                px += step;
                return px;
        end
end
```

در تعریف کلاس استفاده از () اختیاری است. تنها زمانی از () استفاده میشود که MyClass از کلاس دیگری (class_name1) ارث بری کند. در این حالت نام کلاس در داخل پرانتز قرار میگیرد.

```
[private/public] [const] <class_name> <object_name> [= <class_name> (<parameters>)]
```

```
// object instantiation examples
public Dot dot =  Dot(1 , 2);
const Dot origin = Dot();
public Random rand;              // uninstatiated object
Point point = Null          // assigning Null to an object
```

- دستورات حلقه:

```
// inc / dec          ++ /--

for ([<datatype>] <initialization>; <conditions> [; <inc/dec>]) begin

        <code>

end
```

```
// for loop examples

for (int myVar = 0; myVar < count or count > 5; myVar++) begin

        sum += myVar;

end

for (int myVar = 0; myVar < count and count > 5; myVar--) begin

        sum += myVar;

end
```

```
for <variable_name> in <iterator_name>begin

        <code>

end
```

```
// iterative for example

for p in myList begin

        newList.add(p.name);

        print(myList[0]);
```

```
end
```

```
while (<conditions>) begin
        <code>
end
```

```
do begin
        <code>
end while (<conditions>)
```

- دستورات شرط:

```
if (<conditions>) begin
        <code>
end
else if (<conditions>) begin
        <code>
end
else begin
        <code>
end
```

```
// if example
if (forecast == Null) begin
        <code>
end
```

```
//ternary expression
<conditions> ? <expression> : <expression>
```

```
myFaveWeather = isSummer ? “Sunny” : “Snowy”;  // used in assignments
...
return isSunny ? 25 : (temperature / 2);              // or in any other expression
```

:Switch/Case •

```
switch <expression> begin
        case <value> :
                <code>
                [break]
        [ default:
```

```
        <code>
        [break]
    ]
end
```

```
// switch/case example
switch month.name begin
    case "Jan":
        print("it's January");
        break;
    case "Feb":
    case "Dec":
        print("close enough");
        break;
    default:
        print("try again");
end
```

- تعریف تابع:

```
```

```
<return_type> <function_name> ([<parameter_list>]) begin
        <code>
        return<result_name>;
end
```

```
double divide (int num1, int num2) begin
        double result;
        if !check_zero(num2) begin
                return Null;
        end
        result = num1 / num2;
        return result;
end
```

- Exceptions:

```
try begin
        <code>
end
catch (<variable_name> [, <variable_name>] )begin
        <code>
```

```
end
---------------------------------------------------------------

// exception handling example
try begin
        res = num1 / num2;
end
catch (DivideByZero, ValueError )begin
        print(err, “oops.”);
end
```

عملگرها و اولویت: برنامه شما باید بتواند عملگرهای زیر را با اولویت‌بندی داده شده تشخیص دهد.

1. ( )
2. **
3. ~
4. – + (unary, e.g. –**a**)

```
5.  ###                 (unary operator, e.g. a++ or ++a)
6.  *     /     //      %
7.  -     +             (binary, e.g. a - b)
8.  <<    >>
9.  &     ^     |       (bitwise operators)
10. ==    !=    <>
11. <     >     <=      >=
12. not   and   or      ||      &&
13. =     ##=           (e.g. //= or +=)
```

توضیحات تکمیلی:

- قسمت‌های مبهم زبان را تا حد منطقی، می‌توانید خود تعریف و پیاده‌سازی کنید.
- در کنار گرامر خود، حتماً یک یا چند نمونه برنامه در زبانی که برای آن گرامر نوشته‌اید قرار دهید تا قابلیت‌های مختلف گرامرتان را نمایش دهد. در صورت عدم وجود تست‌کیس، نمره کسر خواهد شد.
- فقط فایل گرامر (g4.) و نمونه برنامه‌های خود (فایل‌های .txt یا مشابه آن) را در قالب یک فایل zip با نام **StudentID_Antlr** ارسال کنید.
- در هنگام تحویل پروژه، لازم است به گرامر خود تسلط کافی داشته‌باشید و در صورت نیاز، بتوانید تغییراتی در آن ایجاد کنید.
- انجام پروژه به صورت انفرادی است. در صورت مشاهده هرگونه تخلف، نمره پروژه فرد یا افراد، معادل ۱۰۰- خواهد بود.
- **آدرس تحویل پروژه:** سامانه آموزش مجازی ویو
- **مهلت تحویل:** یکشنبه ۱۱ اردیبهشت, ۲۳:۵۹

موفق باشید

تیم حل‌تمرین