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. در ابتدای برنامه و یا در ابتدای توابع، کتابخانه‌های مورد نیاز import می‌شوند.
۲. هر برنامه شامل یک یا چند کلاس است.
۳. در انتهای هر دستور باید semicolon (;) قرار داده‌شود.
۴. بلاک‌ها با { و } تعریف می‌شوند.
۵. هر کلاس می‌تواند شامل توابع و تعریف متغیر باشد.
۶. قوانین نام‌گذاری:
   a. حداقل شامل دو کاراکتر هستند.
   b. با عدد شروع نمی‌شوند.
   c. متشکل از حروف کوچک و بزرگ لاتین، ارقام و کاراکترهای '$' و '_' هستند.
   d. کلمات مشخص شده **bold** در این فایل، کلمات کلیدی هستند و نمی‌توانند نام متغیر باشند.
۷. قبل از تعریف متغیرها و توابع در کلاس، می‌توان سطح دسترسی را با **public**، **private** و یا **protected** مشخص نمود.
۸. 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 وجود داشته باشند یا خیر.

- صدا زدن کتابخانه‌ها:

```
import math;                          // imports all from math
from math import *;                   // imports all from math
from root.myMath import random;       // imports random from root.myMath
from math import random, floor;       // imports random and floor from math
from math import random => rand;      // imports random as rand
```

- تعریف متغیر:

```
var/const <name> : <data_type> [ = <initial_value>];
```

```
const myConst = “Lorem Ipsum”;           // defining constants
var myVar : Int = 25, myStr : String;    // defining two variables,
                                         // one is initiated
var myArray : new Array [Double] (4);    // array definition
const myInitiatedArray = Array (11, 42, 90.25, 43);
```

اعداد هنگام انتساب اولیه می‌توانند به صورت نماد علمی نیز وارد شوند (برای مثال: 1.1209e-19 یا 047.) 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 اساسی مانند **Int**، **Double**، **String** و ... را داشته باشد.

- تعریف کلاس و instantiation:

```
class MyClass [extends <parent_name>] [implements <trait_1> with <trait_2> with …] {
	<class_body>
}
```

```
// example
class Point implements Movable with Plottable {
	private var px: Int, py: Int;
	Point (Int px = 0, Int py = 0) {
		this.px = px;
		this.py = py;
	}
	Int moveHorizontal (Int step) {
		px += step;
		return px;
	}
}
```

```
var/const <object_name> : new <class_name> (<parameters>)
```

```
// object instantiation examples
var point : new Point(1 , 2);
const origin : new Point();
```

- دستورات حلقه:

```
for (<initialization>; <conditions>; <inc/dec>){
      <code>
}
```

```
// for loop example
for (myVar = 0; myVar < count && count > 5; myVar++) {
      sum += myVar;
}
```

```
for (<variable_name> in <iterator_name>){
      <code>
}
```

```
// iterative for example
for (var obj in myList ) {
      newList.add (obj.name);
}
```

```
while (<conditions>) {
      <code>
}
```

```
do {
      <code>
} while (<conditions>)
```

- دستورات شرط:

```
if (<conditions>) {
      <code>
}
elif (<conditions>) {
      <code>
}
else {
      <code>
}
```

- Switch/Case:

```
switch (<expression>) {
      case <value> :
            <code>
            [break]
      [ default:
            <code>
            [break]
      ]
}
```

```
// switch/case example
switch (name) {
      case “Jan”:
            print(“it’s January”);
```

```
            break;
      case “Feb”:
      case “Dec”:
            print(“close enough”);
            break;
      default:
            print(“try again”);
   }
```

- تعریف تابع:

```
<return_type> <function_name> ([<parameter_list>]) {
      <code>
}
```

```
Double divide (Int num1, Int num2) {
      var result : Double;
      result = num1 / num2;
      return result;
}
```

- Exceptions:

```
try {
      <code>
}
on <exception_name> [catch (<variable_name)] {
```

```
      <code>
}
// or:
catch (<variable_name>) {
      <code>
}
```

```
// exception handling example
try {
      res = num1 / num2;
}
on DivisionByZeroException catch (err) {
      print (“num 2 = 0”);
      print (err);
}
catch (err) {
      print (err, “oops.”);
}
```

- String Interpolation: استفاده از متغیرها در رشته به کمک **“${<variable>}”** امکان‌پذیر است.

```
var id: Int = 9812762000;
var grade: Double = 18.25, bonus: Double = 0.5;
var stuInfo : String = “student ${id} has grade ${grade+bonus}”;
print(stuInfo);
// output: student 9812762000 has grade 18.75
```

- 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:

- می‌توانید قسمت‌های مبهم زبان را 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 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:** https://www.dropbox.com/request/saJS4uDwAfeYy56p4Qbu
- **مهلت تحویل:** جمعه ۱۰ اردیبهشت، ساعت ۲۳:۵۵

موفق باشید

تیم حل‌تمرین