به نام خدا

پروژه درس اصول طراحی کامپایلر

کامپایلر زبان japy

مقدمه:

هدف از این پروژه طراحی کامپایلر زبان japy می باشد. طراحی این کامپایلر به صورت فاز به فاز پیش خواهد رفت. گرامر زبان japy در فایل ضمیمه در اختیار شما قرار گرفته است. در این فاز از شما انتظار می رود پس از مطالعه سند این زبان و آشنایی با قواعد آن، برای یک ورودی که قطعه کدی به زبان japy است خروجی مورد نظر که توضیحات آن در ادامه است را تولید نمایید. فاز یکم پروژه صرفا جهت آشنایی شما با ابزار ANTLR و فراگیری چگونگی خروجی گرفتن از توابع طراحی شده است.

توضیحات گرامر:

این گرامر توصیف کننده یک زبان شی‌گرا و ترکیبی از پایتون و جاوا است که تغییراتی را شامل می شود. در ادامه به بررسی جزئیات می‌پردازیم.

یک نمونه از کد نوشته شده به این زبان:

```
MAIN class Sample begin
    func main() returns double begin
        double a = 2.0
        var b = 1
        var s2 = new string[2]
        var ss = "a"
        s1[0] = ss
        if(a == b)
            sout("True")
        else
            sout("False")
        return 1.0
    end
end
```

قواعد کلی:

- این زبان به بزرگ و کوچک بودن حروف حساس است. همچنین کاراکتر های space, new lline, tab تاثیری در خروجی نخواهند داشت.
- ابتدا محدوده ی بلاک ها (مجموعه ای از statementها) با کلیدواژه begin و انتهای محدوده نیز با end مشخص می‌شود.
- نام گذاری توابع، کلاس ها و متغیر ها به این صورت است:
    - تنها از کاراکتر های A-Z، a-z، - و ارقام تشکیل شده باشند.

- نمی‌تواند با رقم شروع شود.
- معادل کلیدواژه ها نباشد.

- کلید واژه this به کلاسی که در آن هستیم اشاره می‌کند.

- تعریف کلاس: نام هر کلاس در برنامه باید یکتا باشد.

```
MAIN class Sample begin
    func main() returns int begin
    ...
    return 10
    end
end
```

کلید واژه MAIN نشان دهنده ی این است که کلاس مورد نظر باید حتما شامل متد main که public است باشد.
هر کلاس می‌تواند از حداکثر یک کلاس دیگر ارث‌بری کند(با استفاده از کلیدواژه inherits) .

- تعریف تابع: نام هر متد در یک کلاس یکتاست.

```
func main() returns int begin
    ...
    return 10
end
```

بعد از کلید واژه func نام تابع(در صورت داشتن پارامتر ورودی درون (...) ذکر می‌شوند) و بعد از کلید واژه returns، نوع بازگشتی تابع مشخص می‌شود. دقت داشته باشید که توابع باید حتما مقدار بازگشتی داشته و نوع بازگشتی آن نیز مشخص شده باشد.
متد ها می‌توانند public، private و protected باشند.(پیش‌فرض: public)

- انواع داده ای:
  شامل ۳ نوع پایه string، double ، bool می‌باشد. علاوه بر این هر متغیر می‌تواند از جنس یکی از کلاس های برنامه باشد.
  همچنین در این زبان یک نوع آرایه نیز تعریف شده است که یک بعدی است و می‌تواند از هر نوعی باشد: type[]
- عملگر ها: مشابه هر آنچه تا کنون آموخته اید 😊 .
  نکته: مفهوم lvalue و rvalue در این زبان مشابه زبان جاوا است.

- ساختار تصمیم گیری:
- تنها ساختار تصمیم گیری که شامل یک if، چندین elif و یک else می‌باشد. البته ساختار if می تواند بدون elif وelse نیز استفاده شود.

```
MAIN class Sample begin
    func main() returns double begin
        double a = 2.0
        var b = 1
        if(a == b)
            sout("True")
        else
            sout("False")
        return 1.0
    end
end
```

- ساختار حلقه:

تنها ساختار حلقه while می باشد که یک عبارت expression با نوع bool را می‌گیرد و تا زمانی که برقرار باشد بدنه حلقه را تکرار می‌کند.

```
MAIN class Sample begin
    func main() returns double begin
        ...
        while (a > 0) begin
            if(a == 2) begin
                a = b * 10
                break
            end
            elseif(a == 3)
                continue
            else
                a--
        end
        return 1.0
    end
end
```

دستورات break و continue فقط در حلقه قابل استفاده هستند و مشابه دیگر زبان ها عمل می‌کنند.

- توابع و فیلدهای پیش‌فرض:

تابع sout همان print است . می‌تواند یک آرایه از double و یا متغیر double و string دریافت و آن را در کنسول چاپ کند. (new line هم چاپ می‌کند).

فیلد length تنها برای آرایه ها تعریف می‌شود و طول آرایه را برمی‌گرداند.

توضیحات فاز اول:

با توجه به ویدیویی که در اختیارتان قرار داده شده است به راه اندازی اولیه پروژه بپردازید. در این ویدئو چگونگی عملکرد گرامر ها و طرز کار با listener ها نیز توضیح داده شده است. با توجه به ویدئو شما باید پس import کردن یک قطعه کد japy با استفاده از Listener ها یک خروجی تولید نمایید. این خروجی نمایانگر اجزای مختلف قطعه کد ورودی و جزئیات آن است.
هدف در این فاز دریافت کد ورودی زبان japy و تبدیل به خروجی با فرمت تعریف شده ذیل است:

Class definition:
Input:

```
class A begin
   ….
end
public class B inherits A begin
   ...
end
```

Output:

```
<class ‘A’ >
        …
</class>
<class ‘B’, public, inherits ’A’ >
        …
</class>
```

Function definition:
Input:

```
private func a (x: bool [], y: string, z: double) returns string begin
   …
   return b + c
end
```

Output:

```
<function ‘a’, private, parameters: [(x, bool), (y, string), (z, double)]>
   …
</function return (b + c, string)>
```

*ارسال function به عنوان parameter ممکن نیست

Variable Initialization:
Input:

```
field double [] x
protected field string y, z
var a = new_function (۱۰)
b = ۱۰ * c
d++
```

Output:

```
x: (field, double [])
y, z: (field, protected, string)
new_function (۱۰) -> (a, var)
۱۰ * c -> b
```

\ + d –> d

For:
Input:

```
for (i = 0; i < n; i++) begin
    …
end
```

Output:

```
<for init: <i = 0>, condition: <i < n>, step: <i++>>
    …
</for>
```

If:
Input:

```
if (a) begin
    …
end
if (b > c) begin
    …
end
elif (c > b) begin
    …
end
else begin
    …
end
```

Output:

```
<if condition: <i < n>>
    …
</if>
<if condition: <b >  c>>
    …
<elif condition: <c > b>>
    …
<else>
    …
</else>
```

While:
Input:

```
while (i < n) begin
    …
end
```

Output:

```
<while condition: <i < n>>
    …
</while>
```

* درصورت استفاده از break باید به جای آن از goto N که در آن N به شماره خط اولین دستور پس از حلقه اشاره دارد استفاده کنید و هنگامی که با continue مواجه شدید نیز به جای آن از goto N استفاده کنید که N نمایانگره شماره خط تگ باز حلقه لوپ مورد نظر است.

توجه داشته باشید از شما خواسته شده است همانند مثال بالا دندانه گذاری (Indentation) بلاک های کد را در خروجی برآورده سازید. به این معنی که خطوط خروجی می بایستد با توجه جایگاهشان در ساختار کد با فاصله مناسب از ابتدای خط چاپ شوند.
هر indent level چهار عدد space می باشد.

توضیحات فاز دوم :

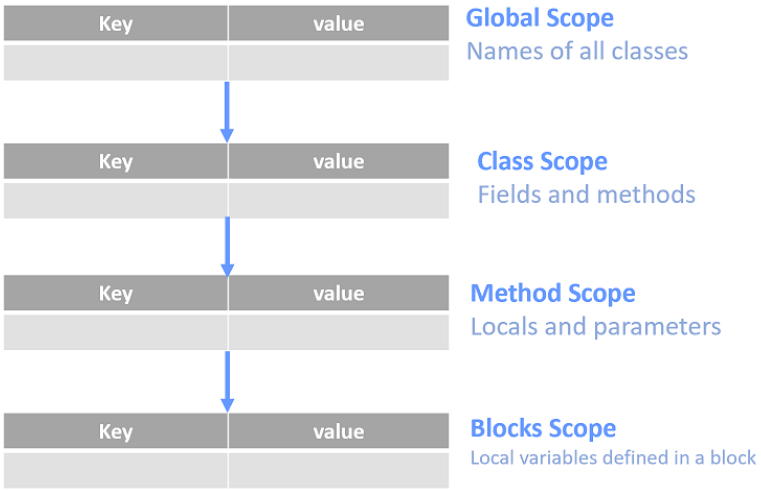
در این فاز قصد داریم اطلاعاتی را جمع آوری و در جدول علائم ذخیره کرده و سپس جدول را نمایش دهیم.
جدول علائم: ساختار داده‌ای است که برای نگهداری شناسه های علائم تعریف شده در کد ورودی استفاده میشود.
طراحی جدول علائم:
استفاده کرد که با (List, Linked List, Hash Table, ...) برای طراحی این جدول میتوان از روشهای مختلفی توجه به نیاز، نوع زبان، پیچیدگی و نظر طراح انتخاب میشود.
value آن نام شناسه و key میباشد. به اینصورت که Hash Table ساده ترین نوع پیاده سازی این جدول استفاده از آن مقدار(مجموعه مقادیر) ویژگیهای مربوط به شناسه است.
هر جدول علائم دو متد اصلی دارد که اطلاعات مربوط به شناسه از طریق این دو متد در جدول ذخیره یا از جدول بازیابی میشوند.

insert (idefName, attributes)
lookup (idefName)

* در زبان japy هر scope یک جدول علائم مخصوص به خود دارد.
Scopes:

- تعریف کلاس
- تعریف توابع
- ساختار های تصمیم گیری
- ساختار های تکرار

اسکوپ ها و جداول علائم (صرفا جهت اطلاع)
همانطور که پیشتر گفته شد، هر اسکوپ شامل یک جدول علائم میباشد. بنابرین علائمی(شناسه هایی) که در هر اسکوپ تعریف میشوند در جدول علائم این اسکوپ ذخیره میشوند. از آنجایی که اسکوپ ها میتوانند تو در تو باشند، جداول علائم اسکوپ ها با یکدیگر رابطه درختی دارند.



نکته: در صورت تعریف دوباره یک کلاس، متد و یا فیلد اسم آن را عوض میکنیم و به symbol table نام name_line_column اضافه میکنیم و اسم آن را به این صورت ذخیره میکنیم

Input Sample:

```
private class A begin
   public field string new_var
end
MAIN class B inherits A begin
   public func main (args: bool[], world: bool) returns string begin
      new_var = "Hello"
      if (world) begin
                var i = ١٠
         sout(new_war)
      end
      else begin
         var i = ٠
         while( i < ١٠ ) begin
            i++
            sout(new_war + " World")
         end
      end
   end
end
```

Output Sample:

```
----------------- porogram:"C:\\compiler\\project\\main.txt" -----------------
key = class_A, value = (name: A) (accessModifier: private)
key = class_B, value = (name: B) (accessModifier: public) (inherits: class_A) (main)
-------------------------------------------------------------------------------
---------------------------------- A: (١, ٣) ----------------------------------
key = field_new_var, value = (name: new_var) (accessModifier: public) (type: string)
-------------------------------------------------------------------------------
---------------------------------- B: (٤, ١٩) ---------------------------------
key = function_main, value = (name: main) (accessModifier: public) (return: string)
```

```
parameters: [[(index: ۰), (name: args), (type: (bool, is_array))],
      [(index: ۱), (name: world), (type: bool)]]
--------------------------------------------------------------------------------
-------------------------------- main: (۵, ۱۸) --------------------------------
                            !NO KEY FOUND!
--------------------------------------------------------------------------------
-------------------------------- if: (۷, ۱۰) ---------------------------------
key = var_i, value = (name: i) (first_appearance: ۸)
--------------------------------------------------------------------------------
------------------------------- else: (۱۱, ۱۷) -------------------------------
key = var_i, value = (name: i) (first_appearance: ۱۲)
--------------------------------------------------------------------------------
------------------------------- while: (۱۳, ۱۶) -------------------------------
                            !NO KEY FOUND!
--------------------------------------------------------------------------------
```

توضیحات فاز سوم:

در این فاز میخواهیم با استفاده از جدول علائم به بررسی خطاهای معنایی موجود در برنامه بپردازیم. (دقت شود که از فاز قبل در این فاز باید استفاده شود)

فرمت گزارش خطا:

خطاهای موجود در برنامه را بر اساس فرمت زیر گزارش دهید:

line شماره خط ارور و column پوزیشن آن را در یک خط نشان میدهد.

خطاهایی که لازم است بررسی کنید به صورت زیر است:

- خطای تعریف دوباره متد/خصیصه:

  تعریف دوباره متد:

  Error۱۰۲ : in line [line:column] , method [name] has been defined already

  تعریف دوباره خصیصه:

  Error۱۰٤ : in line [line:column], field [name] has been defined already

  نکته: دو نوع متفاوت میتوانند هم نام باشند به عنوان مثال اگر یک فیلد و متد هم اسم باشند مشکلی نیست.

  نکته: در صورت تعریف دوباره یک کلاس، متد و یا فیلد اسم آن را عوض میکنیم و به symbol table با نام name_line_column اضافه میکنیم و اسم آن را به این صورت ذخیره میکنیم.

  نکته: هر کدام از موارد ذکر شده اگر دوبار تعریف شوند مورد دوم مطرح نیست و فرض میکنیم اصلا وجود ندارد و تنها ازمورد اول استفاده میشود. به عنوان مثال اگر یک کلاس دوبار تعریف شده باشد تنها میتوان از کلاس اول استفاده کرد.

- عدم تطابق نوع بازگشتی

  عدم تطابق نوع بازگشتی متد با نوع بازگشتی تعریف شده توسط متد:

  Error۲۱۰ : in line [line:column], ReturnType of this method must be [MethodReturnType]

- تعداد و نوع پارامترهای متد در هنگام فراخوانی با تعداد پارامترهای رسمی در هنگام تعریف برابر نباشد

- بررسی اندیس آرایه از جهت نوع int و همچنین خارج از بازه نبودن

نمره اضافه:

- در دستورات انتساب نوع عملوند چپ و راست یکی نباشد:
- وجود دور در ارث بری