پروژه درس اصول طراحی کامپیوتر(فاز دوم و سوم)

دکتر امین طوسی

تدریس یاران : الهه متّقین، عادل جلال احمدی

مهلت تحویل : ۱۴۰۱/۴/۳

مقدمه :

هدف این فاز : ۱) "پیاده‌سازی تحلیل‌گر معنایی" ۲)"تشخیص خطا" می‌باشد.

در مرحله اول اطلاعاتی را جمع آوری و در جدول علائم ذخیره می‌کنیم و در آخر جدول را نمایش می‌دهیم.

در مرحله دوم خطاها را توسط تحلیل‌گر معنایی بررسی و سپس چاپ می‌کنیم.

توضیحات:

- جدول علائم (Symbol Table)

ساختار داده‌ای است که برای نگه‌داری شناسه‌های(علائم) تعریف شده در کد ورودی استفاده می‌شود.

- طراحی جدول علائم:

برای طراحی این جدول می‌توان از روش‌های مختلفی(List, Linked List, Hash Table, …) استفاده کرد که با توجه به نیاز، نوع زبان، پیچیدگی و نظر طراح انتخاب می‌شود.

ساده‌ترین نوع پیاده‌سازی این جدول استفاده از Hash Table می‌باشد. به این‌صورت که key آن نام شناسه و value 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 Jython هر Scope یک جدول علائم مخصوص به خود دارد.

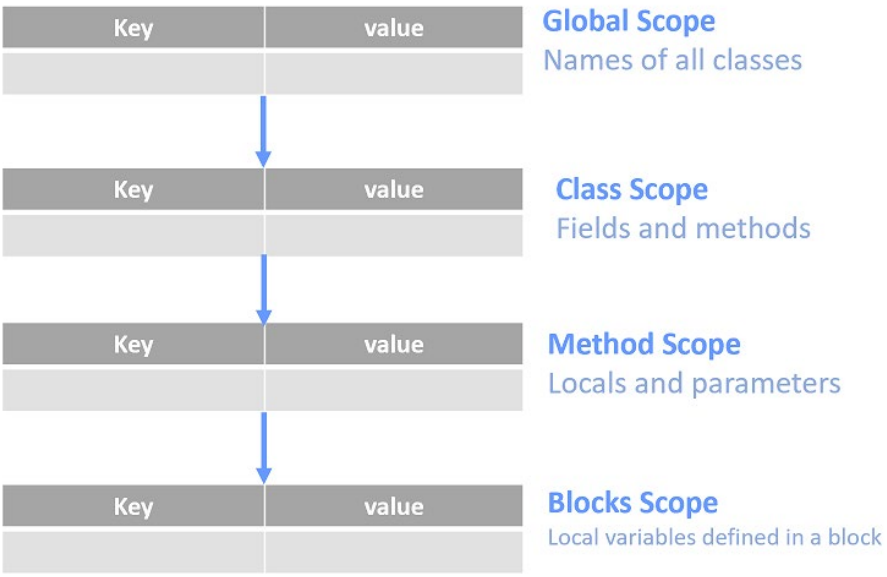
- Scopes:

هر یک از موارد زیر در زبان جایتون یک اسکوپ به حساب می‌آیند:

- تعریف برنامه
- تعریف کلاس
- تعریف متد
- ساختار تصمیم گیری(شروع if و elif و else)
- ساختار تکرار(while و for)

اسکوپ ها و جداول علائم (صرفا جهت اطلاع)

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



در این دو فاز چه باید انجام دهیم؟

فاز دوم:

در این فاز ابتدا چند برنامه به زبان Jython بنویسید؛ سپس هر قطعه کد را به عنوان ورودی دریافت و اسکوپ‌های آن را پردازش کنید و جدول علائم مربوط به آن را بسازید و همه جداول را در یک خروجی و به ترتیب شماره خط شروع اسکوپ چاپ کنید. در ادامه مثالی از ورودی و خروجی به زبان Jython آمده است.

Input:

```
1. import Nothing
2. import Nothing2
3. class Human(Nothing, Nothing2){
4.     Nose nose
5.     Hand[2] hands
6.     Leg[2] legs
7.     int calories
8.     bool isHungry
9.
10.      def Human(Nose n){
11.          self.nose = n
12.      }
13.      def Voice speak(){
14.          Voic voice
15.          return voice
16.      }
17.      def void eat(Food food, int c){
18.          calories += c
19.          newFood = food
20.          Drink water
21.          while(self.isHungry){
22.              Food newFood = Food()
23.              if(1){
24.                  int a
25.              }
26.              eat(newFood)
27.              self.isHungry = self.checkIsHungry()
28.          }
29.      }
30.  }
```

Output:

```
---------program: 1 ---------
Key : import_Nothing | Value : import (name: Nothing)
Key : import_Nothing2 | Value : import (name: Nothing2)
Key: Class_Human | Value : Class (name: human) (parent: Nothing, Nothing2)


================================================================================


---------Human: 3---------
Key : Field_nose | Value : ClassField (name: nose) (type: [ classType= Nose,
isDefiend: False)
Key : Field_hands | Value : ClassArrayField (name : hands) (type : [ class
type= Hand, isDefiend: False)
Key : Field_legs | Value : ClassArrayField (name : legs) (type : [ class
type= Leg, isDefiend: False)
Key : Field_calories | Value : Field (name : calories) (type : [int,
isDefiend: True)
Key : Field_ isHungry | Value : Field (name : isHungry) (type : [bool,
isDefiend: True)
Key : Constructor_Human | Value : Constructor (name : Human) ][parameter
list: [type: Nose, index: 1]
Key : Method_speak | Value : Method (name : speak) (return type: [class type
= Voice])
Key : Method_eat | Value : Method (name : eat) (return type: [void]
[parameter list: [type: Food, index: 1], [type: int, index: 2])

================================================================================

---------Human: 10---------

================================================================================

---------speak: 13---------
Key : Field_voice | Value : MethodField (name: voice) (type: [ classType=
Voice, isDefiend: False)

================================================================================

---------eat: 17---------
Key : Field_water | Value : MethodField (name: water) (type: [ classType=
Drink, isDefiend: False)
Key : Field_food | Value : Parametr (name: food) (type : [ classType=Food ,
isDefined=False)(index: 1)
Key : Field_c | Value : Parametr (name:c) (type : int) (index: 2)
================================================================================

---------while: 21---------

Key : Field_newFood | Value : MethodField (name: newFood) (type: [ classType=
Food, isDefiend: False)


---------nested: 23---------

Key : Field_a | Value : MethodField (name: newFood) (type: [int, isDefiend:
True)
```

مراحل گرفتن خروجی :

۱. برای هر SymbolTable باید دو تابع زیر فراخوانی شوند. تابع toString برای چاپ کردن مقادیر symbolTable و تابع getValue برای دریافت مقادیر از Hashmap استفاده می‌شوند.

```
  public String toString() {

   return "-------------  " + name + " : " + scopeNumber + "  -------------\n" +
              printItems() +
             "----------------------------------------\n";
}
```

```
public String printItems(){
   String itemsStr = "";
   for (Map.Entry<String,SymbolTableItem> entry : items.entrySet()) {
       itemsStr += "Key = " + entry.getKey() + "  | Value = " + entry.getValue()
+ "\n";
   }
   return itemsStr;
}
```

۲. برای چاپ هر item نیز باید متد toString بنویسیم.

فرمت مثال زده شده صرفا یک نمونه فرمت قابل قبول برای خروجی زبان Jython می باشد و دیگر فرمت‌های خوانا، مرتب و نمایش‌دهنده تمام اجزای هر بخش قابل قبول می‌باشند. (اگر فرمت شما خلاقانه،مرتب و بسیار کامل باشد و به طور کاملا واضح و زیبا نمایانگر تمام اجزا جدول علائم اسکوپ باشد می‌تواند شامل نمره اضافه شود.)

نکات:

- شماره خط شروع هر اسکوپ را در ابتدا به همراه نام آن نمایش دهید:

```
------------  Base : 18  ------------
```

- در صورت خالی بودن یک جدول باز هم نیاز به نمایش دادن آن می‌باشد:

```
------------  nested : 39  ------------
----------------------------------------
```

- در هنگام ذخیره سازی هر یک از اجزا در Symbol table نیاز است نوع آن را در کنار نام آن ذخیره کنید به عنوان مثال در قطعه کد زیر نیاز است کلاس Base را به صورت class_Base و متد set را به صورت method_set در قسمت key ذخیره کنید.

```
class Base{
  private int set() {
  }
}
```

فاز سوم:

در این فاز می‌خواهیم با استفاده از جدول علائم به بررسی خطاهای معنایی موجود در برنامه بپردازیم.

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.

شما باید دو نوع خطایی که در ادامه آورده شده است پیاده‌سازی کنید.

۱. **خطای تعریف دوباره متد/خصیصه**

- تعریف دوباره متد:

```
Error102 : in line [line:column] , method [name] has been defined already
```

- تعریف دوباره خصیصه:

```
Error104 : 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 ویا فیلد اسم آن را عوض میکنیم و به سیمبل تیبل اضافه میکنیم و اسم آن را به این صورت ذخیره میکنیم: name_line_column .
    بعنوان مثال اگر متغیر d دوباره تعریف شود آن را به صورت d_۳۴_۴۸ ذخیره می نماییم.
  - 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.

۲. **خطای استفاده از متغیر/کلاس تعریف نشده:**

- استفاده از کلاس تعریف نشده در یک اسکوپ:

```
Error105 : in line [line:column], cannot find  class [className]
```

- استفاده از متغیر یا خصیصه تعریف نشده در اسکوپ:

```
Error106 : in line [line:column], Can not find Variable [name]
```

بخش نمره اضافه :

برای بخش نمره اضافه می‌توانید خطاهای زیر را پیاده سازی کنید.

۳. **عدم تطابق نوع بازگشتی متد با نوع بازگشتی تعریف شده توسط متد:**

```
Error210 : in line [line:column], ReturnType of this method must be
[MethodReturnType]
```

۴. **تعداد و نوع پارامترهای متد در هنگام فراخوانی با تعداد پارامترهای رسمی در هنگام تعریف برابر نباشد:**

```
Error220: in line [line:column], Mismatch arguments.
```

۵. **در دستورات انتساب نوع عملوند چپ و راست یکی نباشد:**

```
Error 230 : in line [line:column], Incompatible types : [LeftType] can not be
converted to [RightType]
```

نکته : برای چاپ کردن ارور ها از ساختار زیر استفاده کنید :

```
public String toString(){
        return "Error" + type + " : in line " + line + ":" + column + " , " +
text;
    }
```

نکات تحویل :

- تمام فایل های مورد نیاز برای تحویل را در یک فایل فشرده به فرمت نام زیر تحویل دهید.

CDProject_PhaseFinal_StundetName_StudentId_TeamMateName_TeamMateStudentID.zip

در صورت تحویل تک نفره به فرمت زیر :

CDProject_PhaseFinal_StundetName_StudentId.zip

مثال :

CDProject_PhaseFinal_AdelJalalAhmadi_97*******_ElaheMottaghin_97******.zip

یا

CDProject_PhaseFinal_AdelJalalAhmadi_97*******.zip

- در صورت اثبات تقلب به طرفین **نمره ۱۰۰-** تعلق می‌گیرد.

موفق باشید