سند زبان برنامه‌نویسی

Smoola

نسخه ۲۰۰۱

۱. مقدمه

زبان Smoola یک زبان شی‌گرا مشابه زبان Java است و برخی از ویژگی‌های زبان‌های شی‌گرا نظیر ارث‌بری را داراست.
در این زبان، یک کلاس اصلی وجود دارد که در آن فقط یک متد main پیاده‌سازی شده‌است. برنامه‌هایی که در این زبان نوشته می‌شوند، همانند زبان Java، در هنگام اجرا دستورات درون این متد را اجرا می‌کنند.

در این زبان، کد برنامه درون یک فایل با پسوند sml. قرار دارد. این فایل شامل یک یا چند کلاس است و هر کلاس شامل تعدادی متغیر و متد است.

۲. ساختار کلی

یک برنامه به زبان Smoola از قسمت های زیر تشکیل شده است:

- یک کلاس اصلی
  - یک متد main
- کلاس‌های دیگر
  - تعریف متغیر
  - تعریف متدها

یک نمونه از کد در این زبان به صورت زیر است:

```
class Test{
  def main(): int{
     writeln((new BabyTest()).testMethod(1,2));
     return 0;
  }
}
class BabyTest{
  var test1: int[];
  var test2: boolean;
  def testMethod(f1: int, f2: int) : int{
    var i: int;
    i = 0;
    test1 = new int[10];

    while(i <> 10){
       test1[i] = i;
    }
    if(test1[1] == 1) then
       test2 = true;
    else{
       test2 = false;
    }
    return test2;
  }
}
```

۲-۱. قواعد کلی نحو

زبان Smoola به بزرگ و کوچک بودن حروف حساس است. در این زبان، وجود کاراکترهای Tab و Space تاثیری در خروجی برنامه ندارند. جزئیات مربوط به Scope ها و خطوط برنامه در ادامه به طور مفصل توضیح داده خواهد شد.

۲-۲. کامنت‌ها

در این زبان، کامنت‌ها تنها تک‌خطی هستند و تمامی کاراکترهای بعد از # تا انتهای خط کامنت به حساب می‌آیند.

```
class Test{
  def main(): int{
     writeln((new Class1()).testMethod(“hi there!”));
     return 0;
     # This is a comment!
  }
}
```

۲-۳. قواعد نام‌گذاری کلاس‌ها، متدها و متغیرها

اسامی انتخابی برای نام‌گذاری کلاس‌ها، متدها و متغیرها باید از قواعد زیر پیروی کنند:

- تنها از کاراکترهای a..z،A..Z، _ و ارقام تشکیل شده باشند.
- با رقم شروع نشوند.
- معادل کلیدواژه‌ها نباشند. در جدول زیر تمامی کلیدواژه‌های زبان Smoola آمده‌است:

boolean	string	int	class	def
then	if	writeln	extends	var
this	false	true	while	else
			return	new

- نام هر کلاس، یکتاست.
- نام هر متد در یک کلاس، یکتاست. در واقع، Method Overloading در این زبان وجود ندارد.
- نام هر متغیر در یک Scope یکتاست اما می‌توان در Scopeهای درونی‌تر از نام‌های متغیرهای بیرونی استفاده کرد که در نتیجه، در طول آن Scope متغیر درونی هنگام استفاده ارجحیت دارد. همچنین، Property Overloading نیز در این زبان وجود ندارد.

۳. کلاس و متد

کلاس اصلی، تنها از یک متد main بدون آرگومان و با مقدار بازگشتی از نوع int تشکیل شده‌است و همچنین، متغیری داخل این کلاس و یا متد main، قابل تعریف نیست. این کلاس، همواره اولین کلاس تعریف‌شده در یک برنامه است. همچنین، کلاس اصلی فرزند هیچ کلاس دیگری نیست و هر کلاسی می‌تواند از حداکثر یک کلاس دیگر ارث‌بری کند(با استفاده از کلیدواژه extends) و از فیلدها و متدهای پدرش استفاده کند.
برخلاف کلاس اصلی، سایر کلاس‌ها می‌توانند یک یا چند متغیر و متد داشته باشند. تعریف متغیرها در کلاس، در ابتدای آن و قبل از تعریف تمام متدها انجام می‌شود. به علاوه، در هر متد نیز تعریف متغیرها در ابتدای آن و قبل از تمام دستورات انجام می‌شود. همچنین در این زبان، قابلیت overriding برای متدهای یک کلاس وجود ندارد.
همه متدها باید مقدار بازگشتی داشته‌باشند و آخرین دستور هر متد نیز باید یک دستور return باشد. همچنین، تعریف یک متد به صورت زیر انجام می‌شود:

```
def method(arg1: int, arg2: boolean): int{
        # body
}
```

دقت کنید که فراخوانی متدها یک Expression است و نه یک Statement(قابلیت فراخوانی یک متد در یک خط جدا در این زبان وجود ندارد!). تنها در متد main می‌توان یک متد از یک کلاس را فراخوانی کرد.

در این زبان، فراخوانی متد به صورت call-by-value است.

برای ساختن یک نمونه(instance) جدید از یک کلاس، به صورت زیر از کلید واژه new برای آن استفاده می‌کنیم:

```
var instance: ClassName;
instance = new ClassName();
```

در مثال بالا، توجه داشته باشید که instance اشاره‌گری به نوع ClassName است، به عبارتی همه متغیرهای non-primitive همانند جاوا اشاره‌گر هستند.

نمونه‌ای از تعریف کلاس‌ها به صورت زیر است:

```
class MainClass{
  def main(): int {
       return new Test2().method2();
  }
}
class Test1{
       var i: int;
       def method1(): string{
              var j: string;
              j = "hello world!";
              return j;
       }
}
class Test2 extends Test1{
       def method2(): int{
              i = 10;
              return i;
       }
}
```

۵. انواع داده

در زبان Smoola، سه تایپ پایه int، string و boolean وجود دارند. علاوه بر آن‌ها، هر متغیر می‌تواند از جنس یکی از کلاس‌هایی باشد که در برنامه تعریف شده‌است.
در این زبان، یک نوع آرایه نیز تعریف شده‌است. این آرایه، یک‌بعدی و از جنس پایه‌ی int می‌باشد. تایپ آن به صورت int[] می‌باشد.

۶. متغیرها

تعریف متغیرها به صورت زیر می‌باشد:

```
var identifier: type;
```

در این زبان نمی‌توان چندین متغیر را در یک خط تعریف کرد و یا در هنگام تعریف یک متغیر، آن را مقداردهی کرد.
در صورتی که به متغیری مقداری نسبت داده نشود، مقدار آن برابر با مقدار پیش‌فرض تایپ خود در نظر گرفته می‌شود. مقادیر پیش‌فرض تایپ‌های مختلف در جدول زیر آمده‌است:

int	0
boolean	false
string	""

در صورتی که متغیر از جنس یک کلاس یا آرایه باشد، مقدار اولیه ندارد و در صورت استفاده، باید خطای مناسب به کاربر داده‌شود.

تایپ‌های موجود در این زبان در جدول زیر آمده‌است:

int
string
boolean
int[ ]
Class

در این زبان، تایپ‌های int، string و boolean از نوع primitive هستند.(خود مقادیر در آن‌ها ذخیره می‌شوند نه پوینتری به خانه‌ای از حافظه) و تایپ‌های دیگر، از نوع non-primitive هستند و در آن‌ها پوینتری به خانه‌ای از حافظه وجود دارد.

پس از تعریف متغیری از جنس آرایه و یا کلاس، باید با استفاده از کلیدواژه new اندازه آن را به صورت زیر مشخص کرد:

```
variable = new int[10];
variable = new ClassName();
```

لازم به ذکر است که اندازه‌ی یک آرایه نمی‌تواند صفر یا عددی منفی باشد.

۷. عملگرها

عملگرها در زبان Smoola به چهار دسته‌ی عملگرهای حسابی، مقایسه‌ای، منطقی، عملگر تخصیص تقسیم می‌شوند.

۷-۱. عملگرهای حسابی

این دسته از عملگرها تنها روی اعداد عمل می‌کنند، لیست این عملگرها در جدول زیر آمده است. در مثال‌های استفاده شده A برابر 20 و B را برابر 10 در نظر بگیرید:

عملگر	شرکت‌پذیری	توضیح	مثال
+	چپ	جمع	A+B=30
-	چپ	تفریق	A-B=10
*	چپ	ضرب	A*B=200
/	چپ	تقسیم	A/B=2 B/A=0
-	راست	منفی تک‌عمل‌وندی	-A=-20

۷-۲. عملگرهای مقایسه‌ای

این عملگرها وظیفه‌ی مقایسه را دارند، پس نتیجه‌ی آن‌ها باید مقدار صحیح یا غلط (true, false) باشد. با این حساب خروجی این عملگرها یک Boolean است.

توجه داشته باشید که عملوند عملگرهای < و > تنها از جنس عدد صحیح هستند. همچنین برای عملگرهای == و <> نیز باید تایپ عملوندها یکسان باشند و در صورت آرایه بودن، اندازه‌ی آن‌ها نیز برابر باشد؛ در غیر اینصورت باید خطای کامپایل گرفته شود.

لیست عملگرهای مقایسه‌ای در جدول زیر آمده است. در مثال‌های استفاده شده مقدار A را برابر 20 و مقدار B را برابر 10 بگیرید:

عملگر	شرکت‌پذیری	توضیح	مثال
==	چپ	تساوی	(A == B) = false
< >	چپ	عدم تساوی	(A < > B) = true
<	چپ	کوچکتر	(A < B) = false
>	چپ	بزرگتر	(A > B) = true

۷-۳. عملگرهای منطقی

در زبان Smoola عملیات منطقی تنها روی نوع داده‌ی Boolean قابل اعمال است.این عملگرها در جدول زیر لیست شده‌اند. در مثال‌های استفاده شده A را برابر true و B را برابر false در نظر بگیرید:

عملگر	شرکت‌پذیری	توضیح	مثال
&&	چپ	عطف منطقی	(A && B) = false
||	چپ	فصل منطقی	(A || B) = true
!	راست	نقیض منطقی	(!A) = false

۷-۴. عملگر تخصیص

این عملگر که به صورت = نمایش داده می‌شود وظیفه‌ی تخصیص را بر عهده دارد. عملگر تخصیص مقدار عملوند سمت راست را به عملوند سمت چپ اختصاص می‌دهد (برای آرایه‌ها مقدار تک‌تک عناصر عملوند سمت راست به عناصر متناظر سمت چپ تخصیص می‌یابد). مقدار خروجی این عملگر برابر با مقدار تخصیص داده شده به عملوند سمت چپ آن است.

دقت داشته باشید که عملوند سمت چپ باید حتماً از نوع lvalue باشد. مفهوم lvalue و rvalue در زبان Smoola مشابه زبان C است. عبارات lvalue عباراتی هستند که به یک مکان در حافظه اشاره می‌کنند، در مقابل

عبارات rvalue به مکان خاصی در حافظه اشاره نمی‌کنند و صرفاً یک عبارت دارای مقدار هستند. به عنوان مثال یک متغیر یک عبارت lvalue است اما عبارت 30+10 یک عبارت rvalue محسوب می‌شود. در زبان Smoola عبارات rvalue تنها می‌توانند سمت راست عملگر تخصیص قرار بگیرند.

۵-۷. اولویت عملگرها

اولویت عملگرها طبق جدول زیر است:

اولویت	دسته	عملگرها	شرکت‌پذیری
۱	پرانتز	()	چپ
۲	دسترسی به عناصر آرایه	[ ]	چپ
۳	تک عملوندی	! -	راست
۴	ضرب و تقسیم	* /	چپ
۵	جمع و تفریق	+-	چپ
۶	رابطه‌ای	< >	چپ
۷	مقایسه‌ی تساوی	== < >	چپ
۸	عطف منطقی	&&	چپ
۹	فصل منطقی	||	چپ
۱۰	تخصیص	=	راست
۱۱	کاما(ورودی متدها)	,	چپ به راست

۸. ساختار تصمیم‌گیری

در زبان Smoola تنها ساختار تصمیم‌گیری، if... else است:

```
if(a == 2) then
	b = true;
else
	b = false;
```

همچنین ساختار if می‌تواند بدون else استفاده گردد.

۹. ساختار تکرار

تنها ساختار تکرار در این زبان while می‌باشد، که یک expression با تایپ boolean را می‌گیرد و تا زمانی که مقدار آن برابر true باشد، حلقه را تکرار می‌کند.

مثال زیر نحوه‌ی استفاده از این ساختار را نشان می‌دهد:

```
while(a <> 0){
	a = a - 1;
}
```

۱۰. قوانین Scopeها

۱۰-۱ Scopeهای موجود در زبان

به طور کلی در زبان Smoola موارد زیر در اسکوپ جدیدی قرار دارند:

۱- خطوط کد داخل یک کلاس.

۲- پارامترها و خطوط کد داخل یک متد.

۱۰-۲. قوانین Scopeها

نکات زیر در مورد Scopeها وجود دارد:

- تعریف کلاس‌ها در بیرونی‌ترین Scope است.
- متغیرهایی که داخل یک Scope تعریف می‌شوند در Scopeهای بیرون آن دسترس‌پذیر نیستند و صرفاً در Scopeهای درون آن قابل دسترسی هستند.

❖ امکان تعریف متغیر با نام یکسان در یک Scope وجود ندارد امّا در Scopeهای درونی آن امکان تعریف مجدد وجود دارد و تا زمان خروج از Scope درونی، نزدیکترین تعریف به آن استفاده می‌شود.

۳-۱۰. قوانین خطوط برنامه

قوانین خطوط این زبان مشابه زبان Java می‌باشد. تنها نکته قابل‌توجه این است که تمامی دستورات، در انتهای خود یک کاراکتر ; دارند.

۱۱. توابع و فیلدهای پیش‌فرض

در زبان Smoola، تنها یک تابع پیش‌فرض وجود دارد و آن، تابع writeln است.

۱-۱۱. تابع writeln

این تابع به صورت ضمنی تعریف شده است و می‌تواند یک آرایه(با هر طولی) و یا یک مقدار int یا string دریافت کند و آن را در کنسول چاپ کند. نمونه‌ای از این دستور به صورت زیر است:

```
writeln("Hello Kiki!");
```

۲-۱۱. فیلد length

این فیلد تنها برای آرایه‌ها تعریف می‌شود و طول یک آرایه را بازمی‌گرداند. به عنوان مثال:

```
var arr : int[];
arr = new int[666];
writeln(arr.length); # the output is 666
```

۱۲. مثال‌ها

در ادامه، چندین کد نمونه از این زبان آمده‌است:

```
class MainClass{
  def main(): int {
     writeln(new Math().factorial(5));
     return 0;
  }
}
class Math{
       var i: int;
       def factorial(f: int): int{
              var j: int;
              j = 1;
              while(f <> 0){
                     j = j * f;
              }
              return j;
       }
}
```

```
class IDENTIFIER{
  def main(): int {
     return new SecondMain().main();
  }
}

class SecondMain{
       var arr: int[];
       var x: int;
       var y: boolean;
       def main(): int{
              x = (2 + (3 || 4)) / 7;
              arr = new int[10];
              if(arr.length == 10 * x) then
                     y = true;
              else
                     y = false;
              return y;
       }
}
```

```
class MainClass{
  def main(): int {
     writeln(SecondMain().main());
     return 0;
  }
}
class SecondMain{
       var s: Rectangle;
       def main(): int{
              var x: int;
              s = new Rectangle();
              x = s.constructor(10,5);
              return s.area();
       }
}
class Rectangle{
       var x: int;
       var y: int;
       var name: string;
       def constructor(x: int, y: int): int{
              this.x = x;
              this.y = y;
              return 0;
       }
       def area(): int{
              return x * y;
       }
}
```