سند زبان برنامهنویسی Smoola

نسخه ۲.۰.۲

۱. مقدمه

زبان Smoola یک زبان شی گرا مشابه زبان Java است و برخی از ویژگیهای زبانهای شی گرا 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 می شوند، همانند زبان Java، در هنگام اجرا دستورات درون این متد را اجرا می کنند.

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
- o 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{
   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 كلى 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 می آیند.

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

۳-۲. قواعد نام گذاری 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، ـ 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 كلاس، يكتاست.
- نام هر متد در یک کلاس، یکتاست. در واقع، Method Overloading در این زبان وجود ندارد.
- نام هر متغیر در یک Scope یکتاست اما می توان در Scopeهای درونی تر 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.

٣. كلاس و متد

کلاس اصلی، تنها از یک متد main بدون آرگومان و با مقدار بازگشتی از نوع int تشکیل شده است و همچنین، متغیری داخل این کلاس و یا متد main، قابل تعریف نیست. این کلاس، همواره اولین کلاس تعریف شده در یک برنامه است. همچنین، کلاس اصلی فرزند هیچ کلاس دیگری نیست و هر کلاسی می تواند از حداکثر یک کلاس دیگر ارث بری کند(با استفاده از کلیدواژه extends) و از فیلدها و متدهای پدرش استفاده کند.

برخلاف کلاس اصلی، سایر کلاسها می توانند یک یا چند متغیر و متد داشته باشند. تعریف متغیرها در کلاس، در ابتدای آن و قبل از ابتدای آن و قبل از تعریف متغیرها در ابتدای آن و قبل از تمام دستورات انجام می شود. همچنین در این زبان، قابلیت overriding برای متدهای یک کلاس وجود ندارد. همه متدها باید مقدار بازگشتی داشته باشند و آخرین دستور هر متد نیز باید یک دستور return باشد. به علاوه، دستور return نمی تواند در جایی غیر از آخرین خط یک متد بیاید. همچنین، تعریف یک متد به صورت زیر انجام می شود:

```
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 می توان یک متد از یک کلاس را در یک خط جدا 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 می کنیم:

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

در مثال بالا، توجه داشته باشید که instance اشاره گری به نوع ClassName است، به عبارتی همه متغیرهای non-primitive همانند جاوا اشاره گر 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;
    }
}
```

۴. کلیدواژه this

کلیدواژه this به کلاسی که در آن قرار داریم اشاره می کند. در این زبان، از this تنها برای فراخوانی یک متد استفاده می شود (برای دسترسی به فیلدهای یک کلاس نمی توان از آن استفاده کرد). به مثال زیر دقت کنید:

```
class MainClass{
    def main(): int {
        return 0;
    }
}
class Test1{
    var i: int;
    def salam(): string{
        return "salam";
    }
    def testMethod(): int{
        this.i = 10; # error
        i = 10; # correct
        writeln(this.salam()); # ok
        return 0;
    }
}
```

٥. انواع داده

در زبان Smoola، سه تایپ پایه string ، int و boolean وجود دارند. علاوه بر آنها، هر متغیر می تواند از جنس یکی از کلاسهایی باشد که در برنامه تعریف شده است.

در این زبان، یک نوع آرایه نیز تعریف شده است. این آرایه، یک بعدی و از جنس پایه ی int می باشد. تایپ آن به صورت []int می باشد.

۶. متغیرها

تعریف متغیرها به صورت زیر می باشد:

```
var identifier: type;
```

در این زبان نمی توان 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 می شود. 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 داده شود.

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 می شوند 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 نمی تواند صفر یا عددی منفی باشد.

٧. عملگرها

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

۱-۷.عملگرهای حسابی

این دسته از عملگرها تنها روی اعداد عمل می کنند، لیست این عملگرها در جدول زیر آمده است. در مثالهای استفاده شده A برابر D و D را برابر D در نظر بگیرید:

مثال	توضيح	شرکتپذیری	عملگر
A+B=30	جمع	چپ	+
A-B=10	تفريق	چپ	-
A*B=200	ضرب	چپ	*
A/B=2 B/A=0	تقسيم	چپ	/
-A=-20	منفى تكعملوندى	راست	-

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 == < > نیز باید تایپ عملوندها یکسان باشند و در صورت آرایه بودن، اندازه ی 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 D و مقدار D را برابر D برابر برابر D بر

مثال	توضيح	شرکتپذیری	عملگر
(A == B) = false	تساوى	چپ	==
(A <> B) = true	عدم تساوی	چپ	<>
(A < B) = false	كوچكتر	چپ	<
(A > B) = true	بزرگتر	چپ	>

۳-۷. عملگرهای منطقی

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

مثال	توضيح	شرکتپذیری	عملگر
(A && B) = false	عطف منطقى	چپ	&&

$(A \parallel B) = true$	فصل منطقى	چپ	
(!A) = false	نقيض منطقى	راست	!

۷-۴. عملگر تخصیص

این عملگر که به صورت = نمایش داده می شود وظیفه ی تخصیص را بر عهده دارد. عملگر تخصیص مقدار عملوند سمت راست به سمت راست را به عملوند سمت چپ اختصاص می دهد (برای آرایه ها مقدار تک تک عناصر عملوند سمت راست به عناصر متناظر سمت چپ تخصیص می یابد). 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 Ivalue باشد. مفهوم lvalue و rvalue در زبان Smoola مشابه زبان C است. عبارات Ivalue عباراتی هستند که به یک مکان در حافظه اشاره می کنند، در مقابل عبارات rvalue به مکان خاصی در حافظه اشاره نمی کنند و صرفاً یک عبارت دارای مقدار هستند. به عنوان مثال یک متغیر یک عبارت lvalue است اما عبارت اما عبارت 30+10 یک عبارت rvalue محسوب می شود. در زبان Smoola عبارات rvalue تنها می توانند سمت راست عملگر تخصیص قرار بگیرند.

٥-٧. اولويت عملگرهااولويت عملگرها طبق جدول زير است:

شرکتپذیری	عملگرها	دسته	اولويت
چپ	()	پرانتز	1
چپ	[]	دسترسی به عناصر آرایه	*
راست	! -	تک عملوندی	٣
چپ	/ *	ضرب و تقسیم	۴
چپ	-+	جمع و تفريق	۵

چپ	< >	رابطهای	۶
چپ	<> ==	مقایسهی تساوی	٧
چپ	&&	عطف منطقى	٨
چپ		فصل منطقى	٩
راست	=	تخصيص	١.
چپ به راست	,	کاما(ورودی متدها)	11

۸. ساختار تصمیم گیری

در زبان Smoola تنها ساختار تصمیم گیری، if... else است:

همچنین ساختار if می تواند بدون else استفاده گردد.

٩. ساختار تكرار

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

مثال زیر نحوهی استفاده از این ساختار را نشان میدهد:

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

۱۰. قوانین Scopeها

Scope ۱۰-۱های موجود در زبان

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

١- خطوط كد داخل يك كلاس.

۲- پارامترها و خطوط کد داخل یک متد.

۲-۱۰. قوانین Scopeها

نكات زير در مورد Scopeها وجود دارد:

- ❖ تعریف کلاسها در بیرونی ترین Scope است.
- ❖ متغیرهایی که داخل یک Scope تعریف می شوند در Scopeهای بیرون آن دسترس پذیر 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 می شود.

٣-١٠. قوانين خطوط برنامه

قوانین خطوط این زبان مشابه زبان Java میباشد. تنها نکته قابل توجه 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.

۱۱-۱ تابع writeln

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

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

۱۱-۲. فیلد length

این فیلد تنها برای آرایهها تعریف می شود و طول یک آرایه را بازمی گرداند. به عنوان مثال:

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

١٢. مثالها

در ادامه، چندین کد نمونه از این زبان آمدهاست:

```
class MainClass{
  def main(): int {
     writeln(new SecondMain().main());
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 y1: int;
      var name: string;
      def constructor(x: int, y: int): int{
             y1 = y;
      def area(): int{
```