LSPS Expression Language 3.2

1 Types and values

1.1 Simple

String	"string" #ASCII_CODE #"not-localized string" $localizedString(argument1, argument2,...)
Boolean	**true** **false**
Integer	1 -100
Decimal	24.86 6.63E-34 **Decimal**(scale) **Decimal**(scale, rounding)
Date	d'yyyy-MM-dd HH:mm:ss.SSS' **date**(year, month, day) **date**(year, month, day, hours, minutes, seconds, millis) **date**(string) **date**(string, pattern)
Object	*any expression*
Null	**null**

1.2 Containers

Set<T>	{ element1,... }
List<T>	[element1,...]
List<Integer>	-5..81 1000..0
Map<K,V>	[key1 -> value1,...] empty map: [->]

1.3 Other

Record	**new** record_name() **new** record_name(value1, value2,...) **new** record_name(field2 -> value2, field4 -> value4,...)
Reference<T>	&variable
Closure {T,...: U}	{ -> expression } { x:Type, y -> expression }
Type	**type**(type_definition) record_name

2 Operators

2.1 Assignment

Assignment	variable := expression

2.2 Arithmetic

Addition	+
Subtraction	-
Increment	++var var++
Decrement	--var var--
Multiplication	*
Division	/
Modulo	%
Exponentiation	**

2.3 Boolean

Equality	== != <>
Comparison	< <= > >= <=> instanceof
Negation	not !
Conjunction	and &&
Disjunction	or ||
Exclusive disjunction	xor
Pattern matching	string **like** pattern
Collection containment	item **in** collection

2.4 Other

String concatenation	+
Parenthetical grouping (precedence)	( )

2.5 Compound Assignment

Addition	+=
Subtraction	-=
Multiplication	*=
Division	/=
Modulo	%=

3 Access

Set element	set[index]
List element	list[index]
Map element	map[key]
Record field	record.field
Record safe field	record?.field
Record reference field	reference.field
Record ref. safe field	reference?.field
Enumeration literal	enum.literal
Dereferencing	*reference
Namespace separator	::
Function call	function|T1,...|(argument1, argument2,...) function|T1,...|(name1->value1, name2->value2,...)
Closure invocation	closure(argument1, argument2,...)
Method call	object.method(argument1, argument2,...)

4 Special constructs

Local variable	**def** type name with assignment: **def** type name := expression
Comments	// single-line comment /* multi-line */
Expression chaining	expr1 ; expr2 ; expr3 ; ...
Block	**begin** expression **end**
Error throw	**error**(string_expression)
Try-catch	**try** expression **catch** exception,... -> expression ... **end**
Cast	expression **as** type

5 Control flow

If-then
If-then-else
If-then-elsif-then
 	
If-then-elsif-then-e
	if condition1 **then**
	expression1
	elsif condition2 **then**
	expression2
	...
	else
	expressionN
	end

Ternary conditional	condition ? expression1 : expression2
Null-coalescing	expression1 ?? expression2
Switch	**switch** arg_expression **case** c1, c2,... -> expression1 ... **default** -> defaultExpression **end**
While looping	**while** condition **do** expression **end**
For looping	**for**(init; condition; update) **do** expression **end**
Collection iteration	**foreach** type iterator **in** collection **do** expression **end**
Looping break (in while,for and foreach)	**break**
Looping iteration skipping (in while, for and foreach)	**continue**

6 Function definition file

Function	/** description */ *annotations* *visibility* <T,...> ReturnType function(Type1 param1,...) *body*
Function body	expression: { expression } native: **native** java_method_path
Annotations	**@SideEffect** **@Deprecated** **@Disabled** **@Status**(statusName) **@Meta**(key1->value1, key2->value2,...)
Visibility	**public** **private**
Function parameter	optional: Type name mandatory: Type name* variadic: Type... name default value: Type name = defaultValueExpression

7 Method definition file

Record methods	Record {
	methods
	}

Method	/** description */ *annotations* *visibility modifiers* <T,...> ReturnType function(Type1 param1, Type2 param2*,...) *body* Note: Annotations and parameters follow syntax of function.
Constructor	/** description */ *annotations* *visibility* Record(Type1 param1, Type2 param2*,...) *body* Note: Annotations, parameters and body follow syntax of function; visibility follows syntax of method.
Method body	expression: { expression } native: **native** java_method_path abstract: ;
Visibility	**public** **private** **protected**
Modifiers	**abstract** **static**

7.1 Access (only in method)

This instance	**this**
Super-type members	**super**.method(argument,...)

7.2 Access (only as first statement of constructor)

Calling constructor of this record	**this**(argument,...)
Calling constructor of supertype	**super**(argument,...)