laSpecialized Computing and Robotics Language (SCARL) Grammar *(fourth draft)*
indiciates revisions from previous draft

---

program → *statement_list*

statement_list → *statement* | *statement statement_list*

statement → ~~*device_declarator_statement*~~ | *variable_definition_statement* | *function_definition_statement* | *class_definition_statement* | *constant_definition_statement* | *variable_declaration_statement*

block_statment → **‘{‘** *statement_list_block_level* **‘}’**

statement_list_block_level → *statement_block_level* | *statement_block_level statement_list_block_level*

statement_block_level → *variable_definition_statement* | *statement_block_statement* | *variable_set_statement* | *function_invocation_statement* | *if_block_statement* | *while_block_statement* | *return_statement* | *method_invocation_statement* | *variable_declaration_statement* | *delete_statement*

~~*device_declarator_statement* → *device_type* **‘IDENTIFIER’ ‘;’**~~

type_declarator → *any_type* **'IDENTIFIER'**

any_type → *array_type* | *pointer_type* | *primitive_type* | *user_type* | *device_type*

array_type → *any_type* **'['** *arithmetic_expression* **']'**

pointer_type → *any_type* **'pointer'**

user_type → **‘IDENTIFIER’**

variable_definition_statement → *type_declarator* **‘=’** *expression* **‘;’**

variable_declaration_statement → *type_declarator* **';'**

delete_statement → **'delete' 'IDENTIFIER' ';'**

constant_definition_statement → **'constant'** *type_declarator* **'='** *expression* **';'**

class_definition_statement → *class_declarator class_body*

class_declarator → *class_name class_extending* | *class_name*

class_name → **'class' 'IDENTIFIER'**

class_extending → **'extends' 'IDENTIFIER'**

class_body → **'{'** *class_attribute_list* **'}'**

class_attribute_list → *class_attribute* | *class_attribute class_attribute_list*

class_attribute → *function_definition_statement* | *variable_declaration_statement*

function_definition_statement → *any_type* **'IDENTIFIER'** **'('** *formal_parameter_list* **')'**
block_statement

variable_set_statement → **'IDENTIFIER'** **'='** *expression* **';'** | *class_attribute_identifier* **'='**
expression **';'**

return_statement → **'return'** *expression* **';'** | **'return'** **';'**

function_invocation → **'IDENTIFIER'** **'('** *parameter_list* **')'**

function_invocation_statement → *function_invocation* **';'**

method_invocation → *class_attribute_identifier* **'('** *parameter_list* **')'**

method_invocation_statement → *method_invocation* **';'**

class_attribute_identifier → **'IDENTIFIER'** **'.'** **'IDENTIFIER'** | **'IDENTIFIER'** **'.'**
class_attribute_identifier

if_block_statement → **'if'** **'('** *expression* **')'** *block_statement* |
'if' **'('** *expression* **')'** *block_statement* **'else'** *block_statement*

while_block_statement → **'while'** **'('** *expression* **')'** *block_statement*

formal_parameter_list → *type_declarator* |
type_declarator **','** *formal_parameter_list* |

parameter_list → *expression* | *expression* **','** *parameter_list* |

expression → *logical_expression* | *allocation_expression*

allocation_expression → **'allocate'** *allocation_invocation*

allocation_invocation → *primitive_type* **'('** *parameter_list* **')'** | **'IDENTIFIER'** **'('**
parameter_list **')'**

logical_expression → *logical_and_expression* |
logical_expression **'||'** *logical_and_expression*

logical_and_expression → *equality_expression* |
logical_and_expression **'&&'** *equality_expression*

equality_expression → *relational_expression*|
equality_expression **'=='** *relational_expression* |
equality_expression **'!='** *relational_expression*

relational_expression → *bool_expression* |
relational_expression **'>'** *bool_expression* |
relational_expression **'<'** *bool_expression* |
relational_expression **'>='** *bool_expression* |
relational_expression **'<='** *bool_expression*

bool_expression → *arithmetic_expression* |
‘!’ *arithmetic_expression*

arithmetic_expression → *arithmetic_factor* |
arithmetic_expression **‘+’** *arithmetic_factor* |
arithmetic_expression **‘-‘** *arithmetic_factor*

arithmetic_factor → *arithmetic_unary* |
arithmetic_factor **’*’** *arithmetic_unary* |
arithmetic_factor **‘/’** *arithmetic_unary*

arithmetic_unary → *unit* | **‘-‘** *arithmetic_unary* | **‘(‘** *arithmetic_expression* **‘)’**

unit → **‘IDENTIFIER’** | *integer_value* | *bool_value* | *function_invocation* | *char_value* |
array_accessor_unit | *method_invocation* | *dereferenced_pointer* |
array_value

array_accessor_unit → **'IDENTIFIER' '['** *arithmetic_expression* **']'**

dereferenced_pointer → **'deref' 'IDENTIFIER'**

array_value → **'STRING_LITERAL'** | *array_initializer*

array_initializer → **'{'** *parameter_list* **'}'**

integer_value → **‘DECIMAL’** | **‘OCTAL’** | **‘HEX’** | **‘BINARY’**

char_value → **'CHAR_LITERAL'**

bool_value → **‘true’** | **‘false’**

primitive_type → **‘bool’** | **‘int’** | **‘char’** | **‘void’**

device_type → **‘LightActuator’** | **‘ServoActuator’** | **’SoundSensor’** |
‘LightSensor’ | **‘DistanceSensor’** | **‘TemperatureSensor’** |
'SerialSensor' | **'SerialActuator'**