Grammar Productions

empty

February 16, 2016

1 Grammar Productions

program	::=	*classDecl**
classDecl	::=	class *id* '{' *fieldDecl** *methDecl** '}'
fieldDecl	::=	*type id* (',' *id*)* ';'
methDecl	::=	(*type* | void) *id* '(' [*formals*] ')' *block*
formals	::=	*type id* (',' *type id*)*
type	::=	*type* '[]' | int | boolean | string | id
block	::=	'{' *varDecl** *stmt** '}'
varDecl	::=	*type id* ';'
stmt	::=	skip ';'
	|	*id* '=' *expr* ';'
	|	*expr* '[' *expr* ']' '=' *expr* ';'
	|	*location* '=' *expr* ';'
	|	*id* '(' [*actuals*] ')' ';'
	|	*location* '(' [*actuals*] ')' ';'
	|	return [*expr*] ';'
	|	if '(' *expr* ')' then *block* else *block*
	|	while '(' *expr* ')' *block*
location	::=	*expr* '.' *id*
actuals	::=	*expr* (',' *expr*)*
expr	::=	*id*
	|	*location*
	|	*call*
	|	*new type* '[' *expr* ']'
	|	*expr* '[' *expr* ']'
	|	*expr* '.' length
	|	this
	|	new *id* '(' [*actuals*] ')'
	|	*expr binary expr*
	|	*unary expr*
	|	*literal*
	|	'(' *expr* ')'
binary	::=	'+' | '−' | '∗' | '/' | '&&' | '||' | '%'
	|	'<' | '<=' | '>' | '>=' | '==' | '! ='
unary	::=	'−' | '!'
literal	::=	*integer* − *literal* | *string* − *literal* | true | false