Entwurf

Simon Bischof Jan Haag Adrian Herrmann Lin Jin Tobias Schlumberger
Matthias Schnetz

29. November 2011

1 Klassendiagramme

2 Zustandsdiagramme

3 Aktivtätsdiagramme

4 Syntax der While-Sprache

4.1 Übersicht der Schlüsselwörter und Sonderzeichen

boolean	→ type_specifier
else	→ if_statement
false	→ logical_expression
if	→ if_statement
int	→ type_specifier
return	→ statement
true	→ logical_expression
while	→ while_statement
0..9	→ integer_literal
a..z,A..Z,_	→ identifier
&	→ logical_expression
|	→ logical_expression
!	→ logical_expression
!=	→ testing_expression
==	→ testing_expression
<	→ testing_expression
<=	→ testing_expression
>	→ testing_expression
>=	→ testing_expression
+	→ numeric_expression
-	→ numeric_expression
*	→ numeric_expression
/	→ numeric_expression
%	→ numeric_expression
,	→ arglist → parameter_list → variable_declaration → variable_initializer
;	→ statement → variable_declaration
=	→ variable_declarator
(	→ expression → if_statement → methode_declaration → while_statement
)	→ expression → if_statement → methode_declaration → while_statement
[	→ expression → type
]	→ expression → type
{	→ statement_block → variable_initializer
}	→ statement_block → variable_initializer
#	→ comment

4.2 Startsymbol

compilation_unit

4.3 Produktionsregeln

```
arglist ::= expression { ",ëxpression }

comment ::= "#... text ..."

compilation_unit ::= { field_declaration }

expression ::= numeric_expression
                | testing_expression
                | literal_expression
                | logical_expression
                | identifier
                | ( "(ëxpression ")")
                | ( expression ( ( "("[ arglist ] ")")
                                  | ( "[ëxpression "]") ) )

field_declaration ::= ( [ comment ] ( method_declaration
                                      | variable_declaration ) )

identifier ::= ä..z,A..Z,_"{ ä..z,A..Z,_,0..9"}

if_statement ::= ïf(ëxpression ")ßtatement_block [ ëlseßtatement_block ]

integer_literal ::= ( "0..9"{ "0..9"} )

literal_expression ::= integer_literal

logical_expression ::= ( "!ëxpression )
                        | ( expression ( "&"
                                         | "
                                         | ( "&&")
                                         | ( |") ) expression )
                        | "true"
                        | "false"

method_declaration ::= type identifier "("[ parameter_list ] ")"( statement_block )

numeric_expression ::= ( ( "+"
                             | ") expression )
                        | ( expression ( "+"
                                          | "
                                          | "*"
                                          | "/"
                                          | "%") expression )

parameter ::= type identifier

parameter_list ::= parameter { ","parameter }

statement ::= variable_declaration
                | ( expression ";")
                | ( statement_block )
                | ( if_statement )
                | ( while_statement )
                | ( "return"[ expression ] ";")
                | ( ";")

statement_block ::= "{"{ statement } "}"

testing_expression ::= ( expression ( ≫"
                                      | ≪"
                                      | ≫="
                                      | ≪="
                                      | -="
                                      | "!=") expression )

type ::= type_specifier { "[]"}

type_specifier ::= "boolean"
                     | ïnt"

variable_declaration ::= type variable_declarator { ","variable_declarator } ";"

variable_declarator ::= identifier [ -"variable_initializer ]

variable_initializer ::= expression
                          | ( "{"[ variable_initializer { ","variable_initializer } ] "}")

while_statement ::= "while(ëxpression ")ßtatement_block
```