Entwurfsdokument

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

Praxis der Softwareentwicklung
Projekt 3:
Automatisches Prüfen der Korrektheit von Programmen
Gruppe 1

WS 2011/2012

Inhaltsverzeichnis

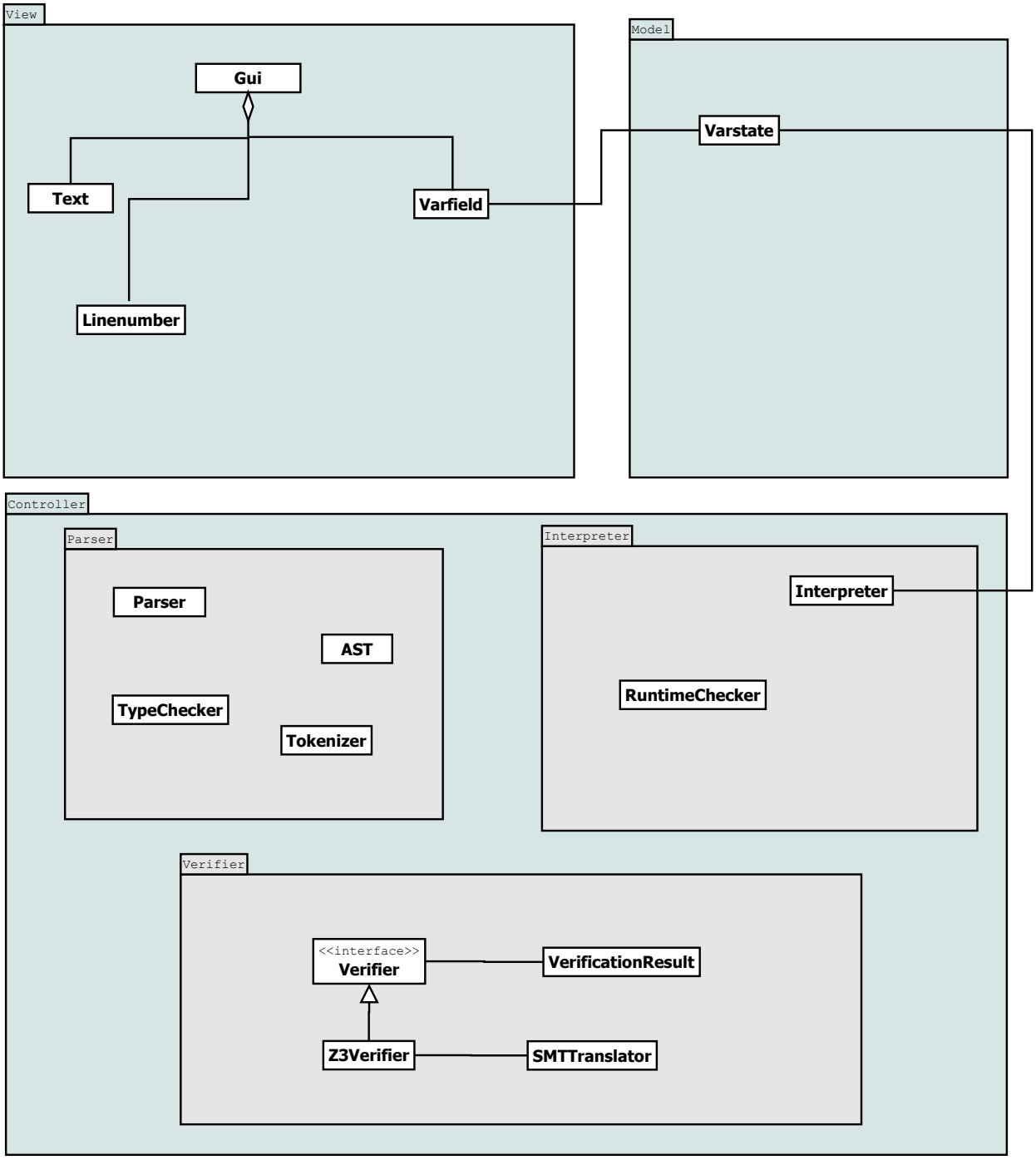
1 Klassendiagramme **3**
1.1 Übersicht . . . 3
1.2 Feinstruktur der Komponenten . . . 4
1.2.1 Parser . . . 4
1.2.2 Interpreter . . . 5
1.2.3 Beweiser . . . 6
1.2.4 GUI . . . 7

2 Verhaltensdiagramme **8**
2.1 Aktivitätsdiagramme . . . 8
2.1.1 Parser/Type-Checker . . . 8
2.2 Zustandsdiagramm . . . 9

3 Syntax der While-Sprache **10**
3.1 Übersicht der Schlüsselwörter und Sonderzeichen . . . 10
3.2 Startsymbol . . . 11
3.3 Produktionsregeln . . . 11

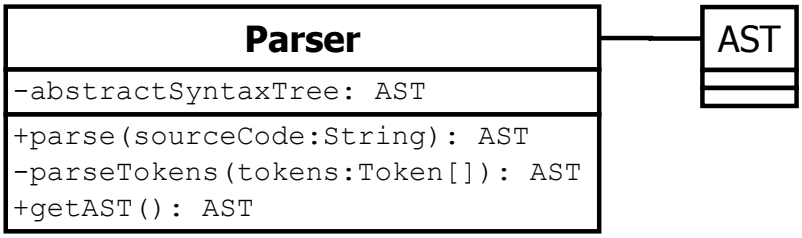
1 Klassendiagramme

1.1 Übersicht

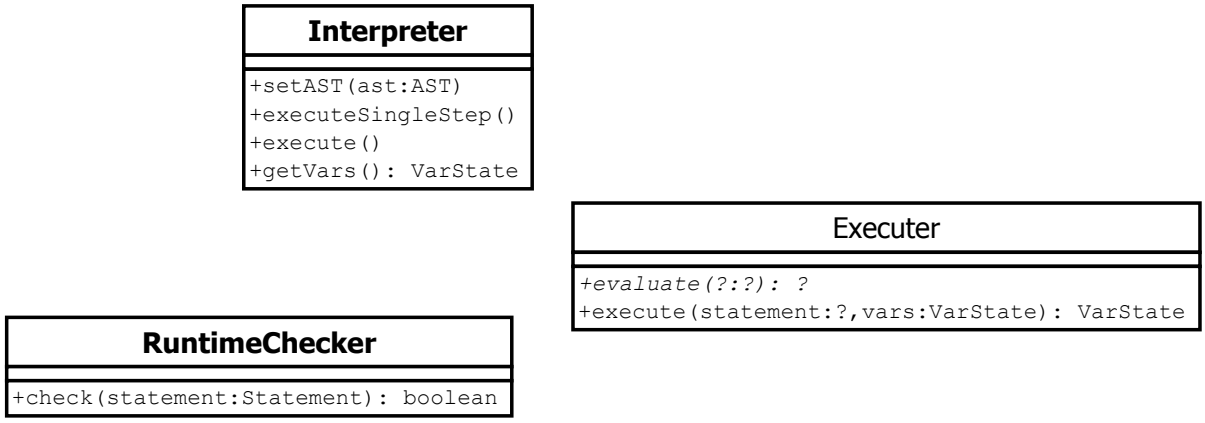


1.2 Feinstruktur der Komponenten

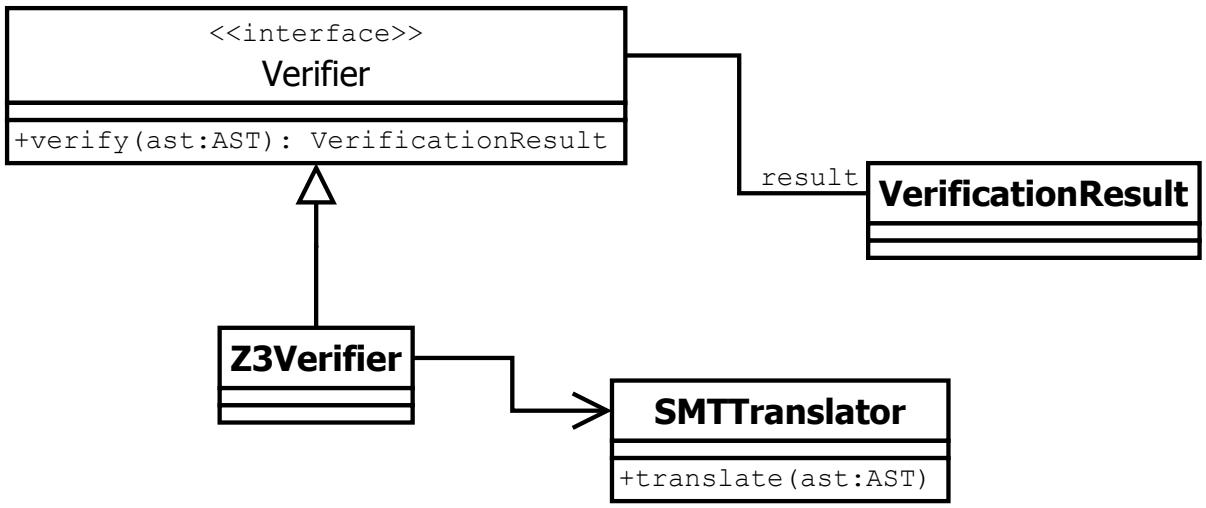
1.2.1 Parser



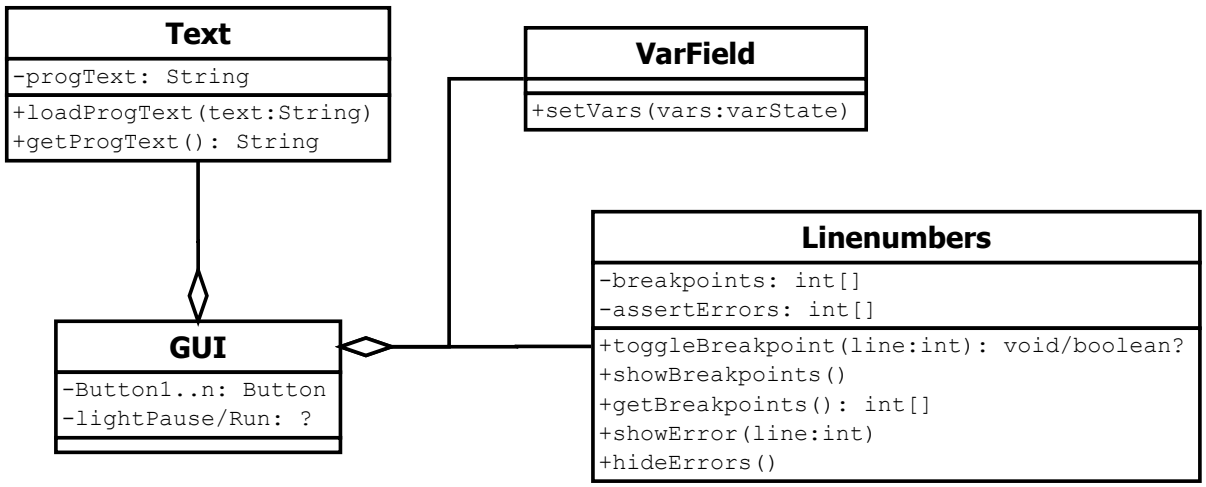
1.2.2 Interpreter



1.2.3 Beweiser



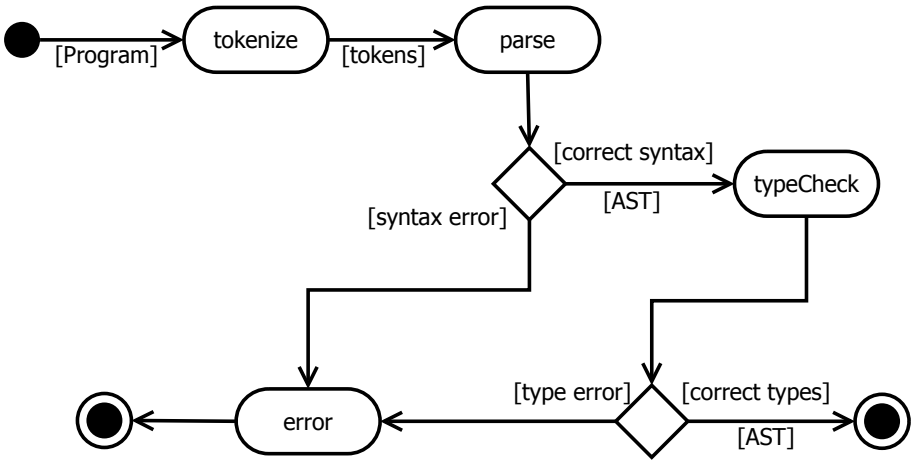
1.2.4 GUI


Text
-progText: String
+loadProgText(text:String)
+getProgText(): String
VarField
+setVars(vars:varState)
Linenumbers
-breakpoints: int[]
-assertErrors: int[]
+toggleBreakpoint(line:int): void/boolean?
+showBreakpoints()
+getBreakpoints(): int[]
+showError(line:int)
+hideErrors()
GUI
-Button1..n: Button
-lightPause/Run: ?

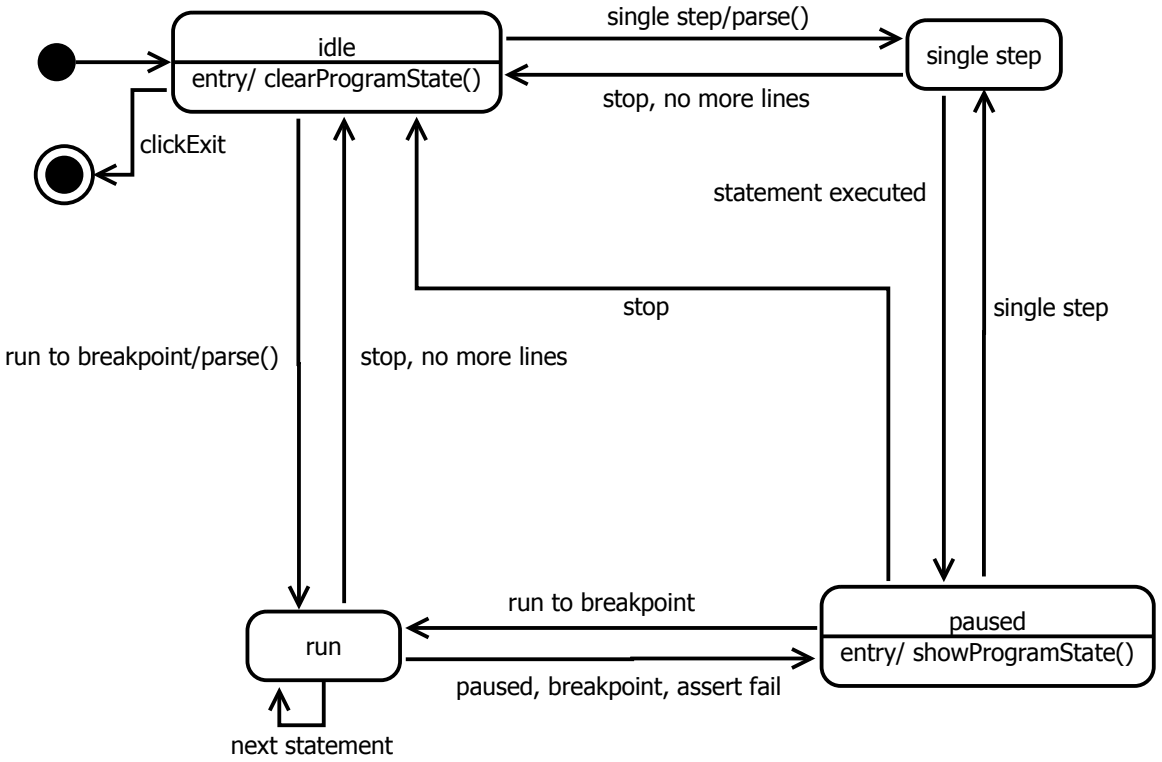
2 Verhaltensdiagramme

2.1 Aktivitätsdiagramme

2.1.1 Parser/Type-Checker



2.2 Zustandsdiagramm



3 Syntax der While-Sprache

3.1 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

3.2 Startsymbol

```
compilation_unit
```

3.3 Produktionsregeln

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

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

compilation_unit ::= { field_declaration }

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

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

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

if_statement ::= "if" "(" expression ")" statement_block [ "else" statement_block ]

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

literal_expression ::= integer_literal

logical_expression ::= ( "!" expression )
                       | ( 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"
                 | "int"

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

variable_declarator ::= identifier [ "=" variable_initializer ]

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

while_statement ::= "while" "(" expression ")" statement_block
```