

Zur Verwendung der Java-Syntax-Beschreibung

Die folgenden Tabellen sollen als Beispiel für die Dokumentation der Verwendung der Java-Sprachbeschreibung dienen. Die zugehörige .tex-Quelle kann darüberhinaus als Anhaltspunkt verwendet werden, wie die Aufgabe 1 der Serie 5 abgegeben werden soll.

Dabei ist zu beachten:

Ebene soll wiedergeben wie die Zeilen der Tabelle zueinander stehen.

Der Übergang von **x.i** nach **x.i.1** bedeutet die Anwendung einer weiteren Produktion (im allgemeinen ohne Lesen von Teilen des Programmfragmentes). Die Verwendung von **x.i**, **x.i+1**, bedeutet, dass zu der Produktion mehrere Teilproduktionen zu betrachten sind.

Programmfragment ist der Teil der zu parsenden Eingabe, der noch zu lesen/erkennen ist.

Produktion ist das Nichtterminal auf das geprüft werden soll. “(*Produktion*)” bedeutet dabei den Verweis dass wir für die Verarbeitung auf diese Produktion schauen müssen, dass aber nur noch ein Rest der zugehörigen rechten Seite gelesen werden muss.

Begründung dokumentiert warum ein Übergang zu der jeweiligen Zeile möglich ist.

1. Parsen von **for (;;);**

Ebene	Programmfragment	Produktion	Begründung
1	for (;;);	ForStatement	
1.1	;;);	ForInit _{opt}	for (erkannt
1	);	(ForStatement)	erkannt, daß kein ForInit; erstes ; gelesen
1.2	);	Expression _{opt}	
1	);	(ForStatement)	erkannt, daß keine Expression, zweites ; gelesen
1.3	);	ForUpdate _{opt}	
1	;	(ForStatement)	erkannt, daß kein ForUpdate, lese)
1.4	;	Statement	parse Rumpf
1.4.1			leeres Statement erkannt

2. Parsen von `for (i=j=5;;)`

Ebene	Programmfragment	Produktion	Begründung
1	<code>for (i=j=5;;)</code>	ForStatement	
1.1	<code>i=j=5;;)</code>	ForInit _{opt}	for erkannt, parse Init
1.1.1	<code>i=j=5;;)</code>	StatementExpressionList	keine LocalVariableDeclaration
1.1.1.1	<code>i=j=5;;)</code>	StatementExpression	nur 1 Statement
1.1.1.1.1	<code>i=j=5;;)</code>	Assignment	richtigen Fall rausfinden
1.1.1.1.1.1	<code>i=j=5;;)</code>	LeftHandSide	1. Teilproduktion
1.1.1.1.1.1.1	<code>i=j=5;;)</code>	ExpressionName	i wird als ExpressionName erkannt
1.1.1.1.1.1.2	<code>=j=5;;)</code>	AssignmentOperator	= wird als AssignmentOperator erkannt
1.1.1.1.1.1.3	<code>j=5;;)</code>	AssignmentExpression	
1.1.1.1.1.1.3.1	<code>j=5;;)</code>	Assignment	ConditionalExpression passt nicht
1.1.1.1.1.1.3.1.1	<code>j=5;;)</code>	LeftHandSide	1. Teilproduktion
1.1.1.1.1.1.3.1.1.1	<code>j=5;;)</code>	ExpressionName	j wird als ExpressionName erkannt
1.1.1.1.1.1.3.1.2	<code>=5;;)</code>	AssignmentOperator	= wird als AssignmentOperator erkannt
1.1.1.1.1.1.3.1.3	<code>5;;)</code>	AssignmentExpression	5 wird als ConditionalExpression erkannt
1	<code>);</code>	(ForStatement)	Ende ForInit erkannt, ; gesehen
1.2	<code>)</code>	Expression _{opt}	; gelesen, erkannt, daß keine Expression
1	<code>)</code>	(ForStatement)	erkannt, dass keine Expression, Fehler: erkannt, daß; fehlt