JaML: XML Representation of Java source code

G. Fischer, J. Lusiardi
University of Würzburg
Department of Informatics
Am Hubland
D 97074 Würzburg
{fischer, lusiardi}@informatik.uni-wuerzburg.de

May 29, 2008

1 Introduction

As source code is one of the most important assets in software business, many people, and also more and more programs, produce it. And at least as many must be able to read and understand source code from other people or programs. This is the case in learning scenarios but also code reviewers may need assistence in understanding external code.

Code reviewing can be assisted by software metrics, that may point to critical sections in the source code. Detecting bad smells also supports identification of code section worth to improve. Learning is normally assisted by "easy to read code formatting", syntax highlighting and additional help tools.

Until now, all these different tools to support programmers, reviewers and learners were hard to implement because there was no direct access to the necessary information. There clearly was a lack of a standard interface to this data.

Our proposed way to make these tasks easier for Java is JaML, an XML application that is an abbreviation for **Ja**va **M**arkup **L**anguage. JaML is an abstract intermediate layer between raw source code and support tools. It enables standardised access to information about the code, which is normally nearly inaccessible.

2 Design Criteria for JaML

To be able to use JaML for many different applications, it contains as much information as made available by the parser. In addition, we evolved two main criteria, that had a deep impact on the design of JaML:

1. Include the complete source code in text nodes for easy restoration
2. Do not alter comments or white space

The first rule is important to locate the positions in original code when a problem was detected using the JaML representation. This rule is broken by many existing projects like JavaML [1] but is essential for our field of application.

The second rule contradicts the method of operation of nearly all parsers and compilers, since comments and whitespaces are normally assumed to be irrelevant for the resulting binary executable. Since comments can be a really important component of the source code, JaML was designed to preserve them. Imagine a

software to check if all parameters for a method are at least mentioned in the method's javadoc comment.

In addition to these two rules, simple and well known interfaces to access the generated JaML data should be available. This goal was achieved by implementing JaML as an XML application. The access to the JaML data can be done by standard tools like SAX, DOM or XPATH.

3 Related Work

Two approaches of related work in the field of representing languages will be discussed here.

The first is JavaML [1] by Greg J. Badros. JavaML is described as XML representation for Java and is in this aspect clearly like JaML. The author states that JavaML has most advantages of the classical representation as user readability and large toolbase. It overcomes some disadvantages of the classical text view as it simplifies access to many information in the source code. But JavaML has some flaws with respect to the two design rules stated above. The original source code is not stored in text nodes and in addition the code is also abstracted. For example, JavaML generalizes all loops to a single loop element. Types are also handled in a different way in JavaML. There is no exact information stored about references to type like the containing package. Last but not least, comments are regarded as not so important, they are attached only to some special elements like class, anonymous-class, method and a few more).

srcML [7] by Jonathan I. Maletic, Michael L. Collard and Andrian Marcus pays more attention to preserving comments and the original formatting of the source code than JavaML. One important feature of srcML are the various abstraction levels and views of the source code. The main disadvantage of srcML is the partial derivation. This has effects on the handling of expressions which are not parsed completly.

All these approaches above have their own special limititions in functionality and design that made them inappropriate for our purposes and so JaML was developed.

4 Detailed Description of JaML

As stated above, most compilers disregard comments and strip out white spaces. This would be a serious violation of the second design rule for JaML. Therefore the implementation of JaML needed a careful selection of the underlying compiler/parser.

The first implementation of JaML (before Java 1.5) relied on JTransform [4] by Holger Eichelberger. JTransform is a parser based on the Java-Cup compiler generator and builds a well annotated parse tree whis was used as fundament for JaML. It uses a visitor pattern with visitors that reassemble all information into a JaML file.

This solution to the problem of finding a suitable parser for JaML was only transient. JTransform supported only Java versions prior to 1.5. The introduction of Java 1.5 brought some very important features, namely generics and some other constructs like annotations and the for-each-loop. These features were not supported by JaML based on JTransform any more.

Due to this problem, the underlying parser was changed from JTransform to the Eclipse Project's JDT [8] compiler. This compiler preserves source code formatting and comments as it is also used for highlighting in the JDT project, it fulfills all requirements for JaML. It also gives an excellent interface to the generated parse

tree which is already annotated with much information about the source code. Another great advantage of the JDT is the very active community that drives the development and maintainance of the compiler.

The actual conversion scheme from Java to JaML was not modified while exchanging the underlying compiler/parser. JaML applies a visitor to the tree generated by the JDT compiler, modifies the data to form a valid JaML document and writes the data to disk.

A detailed description of the JaML XML format is given in chapters 6 and following. This documentation is generated from the JaML XML schema.

The JaML format was defined using an XML Schmema definition. Chapter 6 describes all XML elements, chapter 7 explains all defined complex type, chapter 8 presents all defined simple types and chapter 9 introduces used attribute and element groups.

The the headings above the explanations always mark the name of the defined element, type or group.

The **description**-section gives an informal description of the meaning and usage of a part of the JaML format.

Both sections, **attributes** and **contained attributes**, described defined attributes for an element, a complex type or an attribute group. Attributes are listed in a table, where the first row denotes the name of the attribute, the second row lists the usage, which could be `optional` and `required`. The last column gives the type of this attribute. If an index is given with a type, then this is a reference to the local definition of this simple type.

The **content**-section uses a Backus-Naur-form to describe which elements could be contained in this element or complex type. Indicies denote references to elements or element groups.

For elements, the section **type** may reference a type. This replaces the definition of attributes and contents for this element.

For simple types or elements of simple, the sections **fixed value** and **values**, can appear. **fixed value** means, that only one value is accepted for this type. **values** lists an enumeration of accepted values.

5 Fields of Application

JaML structures and enriches the source code with additional information and it offers easier access to this information than the raw source code. Therefore it can be applied in many fields. Some of them will be introduced now.

As described in the introduction, JaML facilitates source code comprehension. In addition, software analysis can be done in a much simpler way than on raw source code.

5.1 Support Generation of Different Views on Code

Traditionally code is viewed line by line. But for some purposes different views on the code may be more useful.

5.1.1 Traditional View

JaML supports the traditional “line-by-line” view with the following possibilities.

Pretty Printing This is eased by JaML because all source elements are accessible via the appropriate XML tags and can therefore be highlighted and formatted by simply applying a XSL transformation from JaML to HTML or anything alike.

Support of Source Code Navigation The view on the pretty printed source code can be further improved by adding links from method calls to their definition or from type references to their declaration. Tooltips can present information like metrics, types of variables or documentation taken from other JaML tags. Much of this is, of course, included in IDEs like Eclipse but can now easily be integrated in web sites as well. An application for such web sites may be the comprehensive field of e-Learning.

5.1.2 More Abstract Views

Sometimes views with a higher level of abstraction are more appropriate to answer certain questions. These views focus only on some aspects of the source code and emphasize special details of the code.

Selective View of Code Elements A view of this kind may, for example, hide all non-public elements and just show the viewer the elements that are callable from outside. This may reduce the vast amount of information by masking out unaccessible and therefore less relavant parts of the source code.

View of Class Hierarchy This view shows only the father-child dependencies of the used types. This may be useful to control the hierarchy depth and the use of object oriented techniques like method refining and polymorphism.

Graphoriented Views on the Source Code Programs are normally viewed in a linear style but for some applications the representation in form of a graph is more informative and appropriate. A call graph can be used for static analysis of code coverage, a control flow graph is an expression for the complexity of particular parts of the software. And finally a dependency graph will show weaknesses in the coupling between different parts of the software.

5.2 Applying Checks and Queries to Source Code

In JaML files there is much more information directly accessible than in raw source files. First this must be denoted to the enrichment which is done through the compiler/parser but the format of XML further improves the possibilty of querying information from JaML files and checking of source code.

Many different techniques are applyable to query JaML data, ranging from Prolog[5] over SQL to XML based technologies like XQuery[3] and XPath[2].

Possible checks, that may be performed, are tightly coupled to the following topic on metrics but also different types of checks may be performed. This will be covered in the e-Learning section.

5.3 Software Measurement by Metrics

There are some simple metrics that do not benefit from JaML. These are metrics like lines of code an source lines of code, which can be easily computed by using simple tools because no semantic information about the code is needed.

But many metrics depend on more and deeper information about the measured program which is hard to retrieve from raw source code. A small selection from the huge amount of metrics (see [6]) will be described and discussed in the next paragraphs.

Number of statements A simple metric whose computation may be easier through JaML is the *number of statements* metric. This is hard to compute directly from the source code, because this may depend on formatting of the source code. In JaML each statenment is encapsulated within a statement tag. To count these statements is an easy task, the following XPath does the job:

```
count(//jaml:statement)
```

It simply counts all statements in the current JaML file but this can be extended to match only a single type or method by specializing the selecting XPath expression.

Afferent coupling of a method A second metric that is nearly impossible to be calculated directly from source code is the *afferent coupling of a method* metric. It describes the number of times the method is called from other parts. This is of course a static metric and can be computed by quering all method-invocation JaML elements that reference the desired method in the method-declaration-ref attribute.

McCabe's Cyclomatic Complexity This metric has been defined for procedural code, but can also be applied to methods in object oriented code. Cyclomatic complexity is a measure for the number of continuous blocks of statements in a method. Each statement that introduces the possibility of branching increases the value of this metric by one. The computation of this metric is of course eased by JaML because it is only neccessary to sum up all branches via XPATH.

As laid out by these examples, the use of JaML makes it by far simpler to calculate metrices on Java programs than on the actual source code.

5.4 Support for e-Learning

e-Learning is already a important field and will become more and more significant in the future. JaML can support the electronic learning of Java by different techniques.

One important task is to support the students at comprehending the new language and programs written in it. The already mentioned pretty printing makes code nicely readable. Adding possibilities to navigate the code with links to retrieve information about used types and methods further increases the readability of the code.

Normaly submissions made by students are checked for correct functionality with tools like JUnit[9] or others. Most of these tools can only check code for correct runtime behaviour.

JaML extends this automatic code testing to cover an other important aspect. It allows to perform static checks on the source code the student turned in. So it is possible to check many constraints that were imposed by the assignment. This could be checks if

- forbidden classes were not used
- required classes were used
- constraints on method signatures hold
- constructs like recursion was used

and many more.

5.5 Foundation of the Ehrengast Framework

The Ehrengast (**E**clipse **h**osted **r**epresentation for **e**xtended **n**otation of **g**eneric **a**bstract **s**yntax **t**rees) Framework is the latest evolutionary step. This framework, founded on JaML, introduces another, more abstract view on Java called JAST (**J**ava **a**bstract **s**yntax **t**ree). JAST is a step towards a language independet representation of source code which is finally reached with GAST (**g**eneric **a**bstract **s**yntax **t**ree).

As Ehrengast was designed to be a language independent framework, more object oriented languages like C++, C#, Python and JavaScript were added to the frame work. With this support of languages, Ehrengast can be described as framework for creation of XML representations and transformation between language dependent source code and language independent representations.

6 Documentation for elements

6.1 JaML

description: represents the root node for this format
attributes:

name:	usage:	type:
`tag-newline`	optional	xs:boolean
`tag-whitespace`	optional	xs:boolean
`version`	optional	xs:string

content: $(\texttt{file}_{(6.85)})^*$

6.2 abstract

description: represents the java keyword **abstract**
type: $keywordType_{(7.4)}$
fixed value: **abstract**

6.3 ampersand

description: represents the symbol **&**.
type: $symbolType_{(7.6)}$
fixed value: **&**

6.4 ampersand-ampersand

description: represents the symbol **&&**.
type: $symbolType_{(7.6)}$
fixed value: **&&**

6.5 ampersand-equal

description: represents the symbol **&=**.
type: $symbolType_{(7.6)}$
fixed value: **&=**

6.6 annotation

description: represents the java keyword **annotation**.
type: $keywordType_{(7.4)}$
fixed value: **annotation**

6.7 annotation-definition

description: defines a declaration of an annotation
attributes:

name:	usage:	type:
`fully-qualified-name`	required	xs:string
`id`	optional	xs:string
`name`	required	xs:string
`package`	required	xs:string
`visibility`	required	visibility-kind$_{(8.11)}$

positions$_{(9.1)}$
content: (*ignoredGroup*$_{(9.5)}$, *modifiersGroup*$_{(9.6)}$, `at`$_{(6.21)}$, *ignoredGroup*$_{(9.5)}$, `interface`$_{(6.109)}$, *ignoredGroup*$_{(9.5)}$, `identifier`$_{(6.96)}$, *ignoredGroup*$_{(9.5)}$, `class-block`$_{(6.37)}$)

6.8 annotation-member-access

description: describes the access to a member of an annotation
attributes:

name:	usage:	type:
`declaration-ref`	optional	xs:string
`name`	optional	xs:string

positions$_{(9.1)}$
content: (`type-ref`$_{(6.223)}$, `identifier`$_{(6.96)}$)

6.9 annotation-member-declaration

description: describes the declaration of a member of an annotation
attributes:

name:	usage:	type:
`has-default`	optional	xs:boolean
`id`	optional	xs:string
`name`	optional	xs:string
`visibility`	required	visibility-kind$_{(8.11)}$

positions$_{(9.1)}$
content: (*ignoredGroup*$_{(9.5)}$, *modifiersGroup*$_{(9.6)}$, `type`$_{(6.211)}$, *ignoredGroup*$_{(9.5)}$, `identifier`$_{(6.96)}$, *ignoredGroup*$_{(9.5)}$, `left-parenthesis`$_{(6.122)}$, *ignoredGroup*$_{(9.5)}$, `right-parenthesis`$_{(6.180)}$, *ignoredGroup*$_{(9.5)}$, (`default-value`$_{(6.54)}$ [in annotation-member-declaration])? , `semicolon`$_{(6.183)}$)

6.10 annotation-modifier

description: represents a modifier that is an annotation
attributes:

name:	usage:	type:
`kind`	optional	xs:string

positions$_{(9.1)}$
content: (`type`$_{(6.211)}$)

6.11 anonymous-class-definition

description: represents the anonymous declaration of a class
attributes:

name:	usage:	type:
`id`	optional	xs:string
`is-generic`	optional	xs:boolean

positions$_{(9.1)}$
content: (`super-type-ref`$_{(6.192)}$ [in anonymous-class-definition], `class-block`$_{(6.37)}$)

6.12 array-access-expression

description: decribes the access to an array
attributes:

name:	usage:	type:
`dimension`	optional	xs:int
`dimensions`	optional	xs:int

positions$_{(9.1)}$
content: (`type-ref`$_{(6.223)}$, `array-qualifier`$_{(6.15)}$ [in array-access-expression], *ignoredGroup*$_{(9.5)}$, `left-bracket`$_{(6.119)}$, *ignoredGroup*$_{(9.5)}$, `expression`$_{(6.77)}$, *ignoredGroup*$_{(9.5)}$, `right-bracket`$_{(6.177)}$)

6.13 array-creation-expression

description: represents an expression to create an array
attributes:

name:	usage:	type:
`dimension`	optional	xs:int
`dimensions`	optional	xs:int

positions$_{(9.1)}$
content: (`type-ref`$_{(6.223)}$, `new`$_{(6.140)}$, *ignoredGroup*$_{(9.5)}$, `type`$_{(6.211)}$, *ignoredGroup*$_{(9.5)}$, (`left-bracket`$_{(6.119)}$, *ignoredGroup*$_{(9.5)}$, (`dimension-expression`$_{(6.55)}$ [in array-creation-expression], *ignoredGroup*$_{(9.5)}$)? , `right-bracket`$_{(6.177)}$)+ , *ignoredGroup*$_{(9.5)}$, (`expression`$_{(6.77)}$)?)

6.14 array-initializer-expression

description: element to represent an array initialiser, that means sth. like 1,2,3
attributes:

positions$_{(9.1)}$
content: (`type-ref`$_{(6.223)}$, `left-brace`$_{(6.118)}$, *ignoredGroup*$_{(9.5)}$, ((`expression-list`$_{(6.78)}$ [in array-initializer-expression], *ignoredGroup*$_{(9.5)}$)? | (`comma`$_{(6.42)}$, *ignoredGroup*$_{(9.5)}$)), `right-brace`$_{(6.176)}$, *ignoredGroup*$_{(9.5)}$)

6.15 array-qualifier [in array-access-expression]

description: used to qualify multidimensional arrays
attributes:
positions$_{(9.1)}$
content: (`expression`$_{(6.77)}$)

6.16 assert

description: represents the java keyword **assert**.
type: *keywordType*$_{(7.4)}$
fixed value: **assert**

6.17 assert-statement

description: describes an assert statement with the neccessary condition and the optional expression that will be outputted.
attributes:
positions$_{(9.1)}$
content: (`assert`$_{(6.16)}$, *ignoredGroup*$_{(9.5)}$, `condition`$_{(6.44)}$, *ignoredGroup*$_{(9.5)}$, (`colon`$_{(6.41)}$, *ignoredGroup*$_{(9.5)}$, `expression`$_{(6.77)}$, *ignoredGroup*$_{(9.5)}$)? , `semicolon`$_{(6.183)}$)

6.18 assignment-expression

description: represent an assignment expression
attributes:

name:	usage:	type:
gibt den Typ des Literals an.`type-ref`	optional	xs:string

positions$_{(9.1)}$
content: (`type-ref`$_{(6.223)}$, `left-hand-side`$_{(6.121)}$, *ignoredGroup*$_{(9.5)}$, *assignment-operators*$_{(9.2)}$, *ignoredGroup*$_{(9.5)}$, `right-hand-side`$_{(6.179)}$, *ignoredGroup*$_{(9.5)}$)

6.19 asterisk

description: represents the symbol *.
type: *symbolType*$_{(7.6)}$
fixed value: *

6.20 asterisk-equal

description: represents the symbol *=.
type: *symbolType*$_{(7.6)}$
fixed value: *=

6.21 at

description: represents the symbol @.
type: *symbolType*$_{(7.6)}$
fixed value: @

6.22 binary-expression

description: represents a binary expression
attributes:

name:	usage:	type:
gibt den Typ des Literals an.`type-ref`	optional	xs:string

positions$_{(9.1)}$
content: (`type-ref`$_{(6.223)}$, `expr-1`$_{(6.75)}$ [in binary-expression], *ignoredGroup*$_{(9.5)}$, *operatorGroup*$_{(9.7)}$, *ignoredGroup*$_{(9.5)}$, `expr-2`$_{(6.76)}$ [in binary-expression])

6.23 block

description: describes a sequence of statements
attributes:

positions$_{(9.1)}$
content: (`left-brace`$_{(6.118)}$, *ignoredGroup*$_{(9.5)}$, (`statement`$_{(6.185)}$, *ignoredGroup*$_{(9.5)}$)* , `right-brace`$_{(6.176)}$)

6.24 boolean

description: represents the java keyword **boolean**.
type: *keywordType*$_{(7.4)}$
fixed value: **boolean**

6.25 boolean-literal

description: represents a boolean literal
attributes:

content: simple content extending boolLiteralType

6.26 break

description: represents the java keyword **break**.
type: *keywordType*$_{(7.4)}$
fixed value: **break**

6.27 break-statement

description: representing a break statement with an optional label
attributes:

name:	usage:	type:
`label`	optional	xs:string

positions$_{(9.1)}$
content: (`break`$_{(6.26)}$, *ignoredGroup*$_{(9.5)}$, (`identifier`$_{(6.96)}$, *ignoredGroup*$_{(9.5)}$)? , `semicolon`$_{(6.183)}$)

6.28 byte

description: represents the java keyword **byte**.
type: *keywordType*$_{(7.4)}$
fixed value: **byte**

6.29 case

description:	represents the java keyword **case**.
type:	$keywordType_{(7.4)}$
fixed value:	**case**

6.30 case-statement

description:	represents the case clause in switch statements
attributes:	
	$positions_{(9.1)}$
content:	(`case`$_{(6.29)}$, $ignoredGroup_{(9.5)}$, `expression`$_{(6.77)}$, $ignoredGroup_{(9.5)}$, `colon`$_{(6.41)}$, $ignoredGroup_{(9.5)}$)

6.31 cast-expression

description:	represents a cast in the java form
attributes:	
	$positions_{(9.1)}$
content:	(`type-ref`$_{(6.223)}$, `left-parenthesis`$_{(6.122)}$, $ignoredGroup_{(9.5)}$, `type`$_{(6.211)}$, $ignoredGroup_{(9.5)}$, `right-parenthesis`$_{(6.180)}$, $ignoredGroup_{(9.5)}$, `expression`$_{(6.77)}$)

6.32 catch

description:	represents the java keyword **catch**.
type:	$keywordType_{(7.4)}$
fixed value:	**catch**

6.33 catch-clause

description:	represents the catch clause of a try-catch statement
attributes:	
	$positions_{(9.1)}$
content:	(`type-ref`$_{(6.223)}$, `catch`$_{(6.32)}$, $ignoredGroup_{(9.5)}$, `left-parenthesis`$_{(6.122)}$, $ignoredGroup_{(9.5)}$, `variable-declaration`$_{(6.227)}$, $ignoredGroup_{(9.5)}$, `right-parenthesis`$_{(6.180)}$, $ignoredGroup_{(9.5)}$, `block`$_{(6.23)}$)

6.34 char

description:	represents the java keyword **char**.
type:	$keywordType_{(7.4)}$
fixed value:	**char**

6.35 character-literal

description:	represents a character literal
attributes:	
content:	simple content extending charLiteralType

6.36 class

description: represents the java keyword **class**.
type: *keywordType*$_{(7.4)}$
fixed value: **class**

6.37 class-block

description: representing the body of a type
attributes:

positions$_{(9.1)}$

content: (`left-brace`$_{(6.118)}$, *ignoredGroup*$_{(9.5)}$, (`class-block-element`$_{(6.38)}$, *ignoredGroup*$_{(9.5)}$)* , `right-brace`$_{(6.176)}$)

6.38 class-block-element

description: this element contains a choice of all elements that can occur in a class-block
attributes:

positions$_{(9.1)}$

content: ((`annotation-member-declaration`$_{(6.9)}$ | `initializer-declaration`$_{(6.103)}$ | `field-declaration-statement`$_{(6.84)}$ | `constructor-declaration`$_{(6.46)}$ | `method-declaration`$_{(6.130)}$ | `type-definition`$_{(6.216)}$ | `enum-constants`$_{(6.69)}$), *ignoredGroup*$_{(9.5)}$, (`semicolon`$_{(6.183)}$)?)

6.39 class-definition

description: represensts the declaration of a class
attributes:

name:	usage:	type:
`fully-qualified-name`	required	xs:string
`id`	optional	xs:string
`is-generic`	optional	xs:boolean
`name`	required	xs:string
`package`	required	xs:string
`visibility`	required	visibility-kind$_{(8.11)}$

positions$_{(9.1)}$

content: (*ignoredGroup*$_{(9.5)}$, *modifiersGroup*$_{(9.6)}$, `class`$_{(6.36)}$, *ignoredGroup*$_{(9.5)}$, `identifier`$_{(6.96)}$, *ignoredGroup*$_{(9.5)}$, (`type-parameters`$_{(6.222)}$, *ignoredGroup*$_{(9.5)}$)? , (`super-class`$_{(6.190)}$, *ignoredGroup*$_{(9.5)}$)? , (`interfaces`$_{(6.111)}$, *ignoredGroup*$_{(9.5)}$)? , `class-block`$_{(6.37)}$)

6.40 class-qualifier

description: a qualifier for classes used in java doc comments
attributes:

name:	usage:	type:
`fqName`	optional	xs:string
`name`	optional	xs:string

content: simple content extending xs:string

6.41 colon

description: represents the symbol **:**.
type: $symbolType_{(7.6)}$
fixed value: **:**

6.42 comma

description: represents the symbol **,**.
type: $symbolType_{(7.6)}$
fixed value: **,**

6.43 comment

description: represents any kind of commant in java. the kind will be given by the attribute `kind`.
attributes:

name:	usage:	type:
`kind`	optional	comment-kind$_{(8.3)}$

positions$_{(9.1)}$
content: mixed content modell
(`newline`$_{(6.141)}$ | `whitespace`$_{(6.235)}$ | `formfeed`$_{(6.94)}$ | `javadoc-tag-element`$_{(6.115)}$ | `javadoc-text-element`$_{(6.116)}$)*

6.44 condition

description: represents a boolean expression and is used for various purposes
attributes:
positions$_{(9.1)}$
content: (`expression`$_{(6.77)}$)

6.45 conditional-expression

description: represents the conditional expression, that means "expression ? then : else".
attributes:

name:	usage:	type:
`type-ref`	optional	xs:string

positions$_{(9.1)}$
content: (`type-ref`$_{(6.223)}$, `condition`$_{(6.44)}$, *ignoredGroup*$_{(9.5)}$, `question`$_{(6.170)}$, *ignoredGroup*$_{(9.5)}$, `then-clause`$_{(6.201)}$ [in conditional-expression], *ignoredGroup*$_{(9.5)}$, `colon`$_{(6.41)}$, *ignoredGroup*$_{(9.5)}$, `else-clause`$_{(6.65)}$ [in conditional-expression])

6.46 constructor-declaration

description: represents a declaration of a constructor
attributes:

name:	usage:	type:
`id`	optional	xs:string
`is-generic`	required	xs:boolean
`name`	required	xs:string
`signature`	required	xs:string
`visibility`	required	visibility-kind$_{(8.11)}$

positions$_{(9.1)}$

content: (*ignoredGroup*$_{(9.5)}$, *modifiersGroup*$_{(9.6)}$, `identifier`$_{(6.96)}$, *ignoredGroup*$_{(9.5)}$, `parameters`$_{(6.155)}$, *ignoredGroup*$_{(9.5)}$, (`left-bracket`$_{(6.119)}$, *ignoredGroup*$_{(9.5)}$, `right-bracket`$_{(6.177)}$, *ignoredGroup*$_{(9.5)}$)* , (`exceptions`$_{(6.73)}$, *ignoredGroup*$_{(9.5)}$)? , (`block`$_{(6.23)}$ | `semicolon`$_{(6.183)}$))

6.47 constructor-ref

description: represents a reference to a constructor
attributes:

name:	usage:	type:
`declaring-class-ref`	optional	xs:string
`id`	optional	xs:string
`is-constructor`	optional	xs:boolean
`is-default-constructor`	optional	xs:boolean
`signature`	optional	xs:string

positions$_{(9.1)}$

content: ((`return-type`$_{(6.175)}$)* , `parameters`$_{(6.156)}$ [in constructor-ref])

6.48 continue

description: represents the java keyword **continue**.
type: *keywordType*$_{(7.4)}$
fixed value: **continue**

6.49 continue-statement

description: representing a continue statement with optional label
attributes:

name:	usage:	type:
`label`	optional	xs:string

positions$_{(9.1)}$

content: (`continue`$_{(6.48)}$, *ignoredGroup*$_{(9.5)}$, (`identifier`$_{(6.96)}$, *ignoredGroup*$_{(9.5)}$)? , `semicolon`$_{(6.183)}$)

6.50 current-type-ref [in parameter]

description: represents the type that was actually used for this parameter
attributes:

positions$_{(9.1)}$

content: (`type-ref`$_{(6.223)}$)

6.51 declared-type-ref [in parameter]

description:	represents the type that was used in the declaration of this parameter
attributes:	
	$positions_{(9.1)}$
content:	(`type-ref`$_{(6.223)}$)

6.52 default

description:	represents the java keyword **default**.
type:	$keywordType_{(7.4)}$
fixed value:	**default**

6.53 default-statement

description:	the default part of a switch statement
attributes:	
	$positions_{(9.1)}$
content:	(`default`$_{(6.52)}$, $ignoredGroup_{(9.5)}$, `colon`$_{(6.41)}$, $ignoredGroup_{(9.5)}$)

6.54 default-value [in annotation-member-declaration]

description:	represents a default value for a member of an annotation
attributes:	
	$positions_{(9.1)}$
content:	(`default`$_{(6.52)}$, $ignoredGroup_{(9.5)}$, `expression`$_{(6.77)}$)

6.55 dimension-expression [in array-creation-expression]

description:	expression to determine the size of an array dimension
attributes:	
	$positions_{(9.1)}$
content:	(`expression`$_{(6.77)}$)

6.56 divide

description:	represents the symbol /.
type:	$symbolType_{(7.6)}$
fixed value:	/

6.57 divide-equal

description:	represents the symbol /=.
type:	$symbolType_{(7.6)}$
fixed value:	/=

6.58 do

description:	represents the java keyword **do**.
type:	$keywordType_{(7.4)}$
fixed value:	**do**

6.59 do-statement

description:	represents the do-while loop of java
attributes:	
	positions$_{(9.1)}$
content:	(`do`$_{(6.58)}$, *ignoredGroup*$_{(9.5)}$, `statement`$_{(6.185)}$, *ignoredGroup*$_{(9.5)}$, `while`$_{(6.233)}$, *ignoredGroup*$_{(9.5)}$, `left-parenthesis`$_{(6.122)}$, *ignoredGroup*$_{(9.5)}$, `condition`$_{(6.44)}$, *ignoredGroup*$_{(9.5)}$, `right-parenthesis`$_{(6.180)}$, *ignoredGroup*$_{(9.5)}$, `semicolon`$_{(6.183)}$)

6.60 dot

description:	represents the symbol **.**
type:	*symbolType*$_{(7.6)}$
fixed value:	**.**

6.61 double

description:	represents the java keyword **double**.
type:	*keywordType*$_{(7.4)}$
fixed value:	**double**

6.62 ellipsis

description:	represents the symbol **...**.
type:	*symbolType*$_{(7.6)}$
fixed value:	**...**

6.63 else

description:	represents the java keyword **else**.
type:	*keywordType*$_{(7.4)}$
fixed value:	**else**

6.64 else-clause

description:	represents the else part of a conditional expression
attributes:	
	positions$_{(9.1)}$
content:	(`statement`$_{(6.185)}$)

6.65 else-clause [in conditional-expression]

description:	represents the else part of a conditional expression
attributes:	
	positions$_{(9.1)}$
content:	(`expression`$_{(6.77)}$)

6.66 empty-statement

description: represents the empty statement in java, that is only a single ';'
attributes:

positions$_{(9.1)}$

content: (`semicolon`$_{(6.183)}$)

6.67 enum

description: represents the java keyword **enum**.
type: *keywordType*$_{(7.4)}$
fixed value: **enum**

6.68 enum-constant-declaration

description: represents a declaration of a java enum constant
attributes:

name:	usage:	type:
`id`	optional	xs:string
`name`	optional	xs:string

positions$_{(9.1)}$

content: (`constructor-ref`$_{(6.47)}$, *ignoredGroup*$_{(9.5)}$, `identifier`$_{(6.96)}$, (`parameters`$_{(6.155)}$)?)

6.69 enum-constants

description: describes a comma separated list of enum constants followed by an optional semicolon
attributes:

positions$_{(9.1)}$

content: ((`enum-constant-declaration`$_{(6.68)}$, *ignoredGroup*$_{(9.5)}$, (`comma`$_{(6.42)}$, *ignoredGroup*$_{(9.5)}$)?)+ , (`semicolon`$_{(6.183)}$)?)

6.70 enum-definition

description: represents the declaration of an enumeration
attributes:

name:	usage:	type:
`fully-qualified-name`	required	xs:string
`id`	optional	xs:string
`name`	required	xs:string
`package`	required	xs:string
`visibility`	required	visibility-kind$_{(8.11)}$

positions$_{(9.1)}$

content: (*ignoredGroup*$_{(9.5)}$, *modifiersGroup*$_{(9.6)}$, `enum`$_{(6.67)}$, *ignoredGroup*$_{(9.5)}$, `identifier`$_{(6.96)}$, *ignoredGroup*$_{(9.5)}$, `class-block`$_{(6.37)}$)

6.71 equal

description: represents the symbol =.
type: $symbolType_{(7.6)}$
fixed value: =

6.72 equalequal

description: represents the symbol ==.
type: $symbolType_{(7.6)}$
fixed value: ==

6.73 exceptions

description: represents a comma separated list of exceptions to be used in the signature of methods
attributes:
$positions_{(9.1)}$
content: (`throws`$_{(6.206)}$, $ignoredGroup_{(9.5)}$, `type`$_{(6.211)}$, ($ignoredGroup_{(9.5)}$, `comma`$_{(6.42)}$, $ignoredGroup_{(9.5)}$, `type`$_{(6.211)}$)*)

6.74 explicit-constructor-invocation-statement

description: represent an explicit invocation of a super contructor or another constructor from the same class
attributes:

name:	usage:	type:
`kind`	optional	xs:string

$positions_{(9.1)}$
content: (`constructor-ref`$_{(6.47)}$, (`qualifier`$_{(6.168)}$)? , (`super`$_{(6.189)}$ | `this`$_{(6.202)}$), $ignoredGroup_{(9.5)}$, `parameters`$_{(6.155)}$, $ignoredGroup_{(9.5)}$, `semicolon`$_{(6.183)}$)

6.75 expr-1 [in binary-expression]

description: representing the first expression of a binary expression
type: $expressionType_{(7.2)}$

6.76 expr-2 [in binary-expression]

description: representing the second expression of a binary expression
type: $expressionType_{(7.2)}$

6.77 expression

description: represents an element that contains one child out of a choice of concrete expressions
attributes:
positions$_{(9.1)}$
content: (`array-access-expression`$_{(6.12)}$ |
`array-creation-expression`$_{(6.13)}$ |
`array-initializer-expression`$_{(6.14)}$ |
`assignment-expression`$_{(6.18)}$ | `binary-expression`$_{(6.22)}$ |
`cast-expression`$_{(6.31)}$ | `conditional-expression`$_{(6.45)}$ |
`field-access-expression`$_{(6.81)}$ | `literal-expression`$_{(6.126)}$ |
`method-invocation`$_{(6.131)}$ | `super-method-invocation`$_{(6.191)}$ |
`instance-creation`$_{(6.105)}$ | `instanceof-expression`$_{(6.107)}$ |
`variable-access-expression`$_{(6.226)}$ |
`parenthesis-expression`$_{(6.158)}$ | `postfix-expression`$_{(6.162)}$ |
`this-expression`$_{(6.203)}$ | `prefix-expression`$_{(6.163)}$ |
`variable-declaration-expression`$_{(6.228)}$)

6.78 expression-list [in array-initializer-expression]

description: describes a comma separated list of expressions
attributes:
positions$_{(9.1)}$
content: ((`expression`$_{(6.77)}$, *ignoredGroup*$_{(9.5)}$, (`comma`$_{(6.42)}$, *ignoredGroup*$_{(9.5)}$)?)+)

6.79 expression-statement

description: an expression statement embedds an expression and ignores the value
attributes:
positions$_{(9.1)}$
content: (`expression`$_{(6.77)}$, *ignoredGroup*$_{(9.5)}$, `semicolon`$_{(6.183)}$)

6.80 extends

description: represents the java keyword **extends**.
type: *keywordType*$_{(7.4)}$
fixed value: **extends**

6.81 field-access-expression

description: describes the access to a field

attributes:

name:	usage:	type:
`declaration-ref`	optional	xs:string
`is-local`	optional	xs:boolean
`name`	optional	xs:string

positions$_{(9.1)}$

content: (`type-ref`$_{(6.223)}$, (`qualifier`$_{(6.168)}$, *ignoredGroup*$_{(9.5)}$)? , (`identifier`$_{(6.96)}$ | `expression`$_{(6.77)}$), *ignoredGroup*$_{(9.5)}$)

6.82 field-declaration

description: represents the declaration of a single field in a field declaration statement

attributes:

name:	usage:	type:
`dimensions`	optional	xs:int
`id`	optional	xs:string
`is-vargs`	optional	xs:boolean
`name`	optional	xs:string
`visibility`	required	visibility-kind$_{(8.11)}$

positions$_{(9.1)}$

content: (`type-ref`$_{(6.223)}$, *ignoredGroup*$_{(9.5)}$, `identifier`$_{(6.96)}$, *ignoredGroup*$_{(9.5)}$, (`left-bracket`$_{(6.119)}$, *ignoredGroup*$_{(9.5)}$, `right-bracket`$_{(6.177)}$, *ignoredGroup*$_{(9.5)}$)* , (`initializer`$_{(6.102)}$, *ignoredGroup*$_{(9.5)}$)?)

6.83 field-declaration-list [in field-declaration-statement]

description: represents a comma separated list of field declarations

attributes:

positions$_{(9.1)}$

content: (`field-declaration`$_{(6.82)}$, (`comma`$_{(6.42)}$, *ignoredGroup*$_{(9.5)}$, `field-declaration`$_{(6.82)}$, *ignoredGroup*$_{(9.5)}$)*)

6.84 field-declaration-statement

description: represents the declaration of a list of fields

attributes:

positions$_{(9.1)}$

content: (*ignoredGroup*$_{(9.5)}$, *modifiersGroup*$_{(9.6)}$, `type`$_{(6.211)}$, *ignoredGroup*$_{(9.5)}$, `field-declaration-list`$_{(6.83)}$ [in field-declaration-statement], *ignoredGroup*$_{(9.5)}$, `semicolon`$_{(6.183)}$, *ignoredGroup*$_{(9.5)}$)

6.85 file

description: represent a java source file
attributes:

name:	usage:	type:
`name`	required	xs:string
`package`	required	xs:string
`path`	required	xs:string

positions$_{(9.1)}$
content: (*ignoredGroup*$_{(9.5)}$, (`package-declaration`$_{(6.150)}$, *ignoredGroup*$_{(9.5)}$)? , (`import-declaration`$_{(6.101)}$, *ignoredGroup*$_{(9.5)}$, (`semicolon`$_{(6.183)}$, *ignoredGroup*$_{(9.5)}$)*)* , (`type-definition`$_{(6.216)}$, *ignoredGroup*$_{(9.5)}$, (`semicolon`$_{(6.183)}$, *ignoredGroup*$_{(9.5)}$)*)*)

6.86 final

description: represents the java keyword **final**.
type: *keywordType*$_{(7.4)}$
fixed value: **final**

6.87 finally

description: represents the java keyword **finally**.
type: *keywordType*$_{(7.4)}$
fixed value: **finally**

6.88 finally-clause

description: the finally clause of the throw statements
attributes:

positions$_{(9.1)}$
content: (`finally`$_{(6.87)}$, *ignoredGroup*$_{(9.5)}$, `block`$_{(6.23)}$)

6.89 float

description: represents the java keyword **float**.
type: *keywordType*$_{(7.4)}$
fixed value: **float**

6.90 for

description: represents the java keyword **for**.
type: *keywordType*$_{(7.4)}$
fixed value: **for**

6.91 for-init-statement [in forStatementGroup]

description: the init part of the classical for-statement
attributes:

positions$_{(9.1)}$
content: (`expression`$_{(6.77)}$, (`comma`$_{(6.42)}$, *ignoredGroup*$_{(9.5)}$, `expression`$_{(6.77)}$, *ignoredGroup*$_{(9.5)}$)*)

6.92 for-statement

description: represents any for-statement. for- and foreach-stament are differentiated by different content modells

attributes:

positions$_{(9.1)}$

content: (`for`$_{(6.90)}$, *ignoredGroup*$_{(9.5)}$, `left-parenthesis`$_{(6.122)}$, *ignoredGroup*$_{(9.5)}$, (*forStatementGroup*$_{(9.4)}$ | *enhancedForStatementGroup*$_{(9.3)}$), `right-parenthesis`$_{(6.180)}$, *ignoredGroup*$_{(9.5)}$, `statement`$_{(6.185)}$)

6.93 for-update-statement [in forStatementGroup]

description: the update part of the classical for-statement

attributes:

positions$_{(9.1)}$

content: (`expression`$_{(6.77)}$, *ignoredGroup*$_{(9.5)}$, (`comma`$_{(6.42)}$, *ignoredGroup*$_{(9.5)}$, `expression`$_{(6.77)}$, *ignoredGroup*$_{(9.5)}$)*)

6.94 formfeed

description: represents the formfeed characters that appeared in some few file in src.zip

attributes:

name:	usage:	type:
`length`	optional	xs:int

content: simple content extending xs:string

6.95 greater-equal

description: represents the symbol >=.

type: *symbolType*$_{(7.6)}$

fixed value: >=

6.96 identifier

description: represents an identifier

type: *identifierType*$_{(7.3)}$

6.97 if

description: represents the java keyword **if**.

type: *keywordType*$_{(7.4)}$

fixed value: **if**

6.98 if-statement

description: represents an if-statement with a condition, a then clause and an optional else clause

attributes:

positions$_{(9.1)}$

content: (`if`$_{(6.97)}$, *ignoredGroup*$_{(9.5)}$, `left-parenthesis`$_{(6.122)}$, *ignoredGroup*$_{(9.5)}$, `condition`$_{(6.44)}$, *ignoredGroup*$_{(9.5)}$, `right-parenthesis`$_{(6.180)}$, *ignoredGroup*$_{(9.5)}$, `then-clause`$_{(6.200)}$, *ignoredGroup*$_{(9.5)}$, (`else`$_{(6.63)}$, *ignoredGroup*$_{(9.5)}$, `else-clause`$_{(6.64)}$)?)

6.99 implements

description: represents the java keyword **implements**.

type: *keywordType*$_{(7.4)}$

fixed value: **implements**

6.100 import

description: represents the java keyword **import**.

type: *keywordType*$_{(7.4)}$

fixed value: **import**

6.101 import-declaration

description: represents an import declaration

attributes:

name:	usage:	type:
`full-name`	required	xs:string
`is-static`	required	xs:boolean
`kind`	required	xs:string
`name`	required	xs:string
`prefix`	required	xs:string
`ref`	required	xs:string

positions$_{(9.1)}$

content: (`import`$_{(6.100)}$, *ignoredGroup*$_{(9.5)}$, (`static`$_{(6.186)}$, *ignoredGroup*$_{(9.5)}$)? , ((`package-name`$_{(6.151)}$, *ignoredGroup*$_{(9.5)}$, `dot`$_{(6.60)}$, *ignoredGroup*$_{(9.5)}$, `asterisk`$_{(6.19)}$, *ignoredGroup*$_{(9.5)}$) | (`type-name`$_{(6.218)}$, *ignoredGroup*$_{(9.5)}$, (`dot`$_{(6.60)}$, *ignoredGroup*$_{(9.5)}$, (`asterisk`$_{(6.19)}$ | `identifier`$_{(6.96)}$), *ignoredGroup*$_{(9.5)}$)?)), `semicolon`$_{(6.183)}$)

6.102 initializer

description: represents the initializer for a variable or field

attributes:

positions$_{(9.1)}$

content: (`equal`$_{(6.71)}$, *ignoredGroup*$_{(9.5)}$, `expression`$_{(6.77)}$)

6.103 initializer-declaration

description: represents an initializer (either static or normal)
attributes:

name:	usage:	type:
`is-static`	optional	xs:boolean

positions$_{(9.1)}$
content: (*ignoredGroup*$_{(9.5)}$, (`modifier`$_{(6.137)}$ [in initializer-declaration], *ignoredGroup*$_{(9.5)}$)? , `block`$_{(6.23)}$)

6.104 instance

description: embedds the expression to access an instance
attributes:

positions$_{(9.1)}$
content: (`expression`$_{(6.77)}$)

6.105 instance-creation

description: represents a creation of a new instance of a class. an qualifier and an anonymous class declaration are optional.
attributes:

positions$_{(9.1)}$
content: (`type-ref`$_{(6.223)}$, `constructor-ref`$_{(6.47)}$, `qualifier`$_{(6.168)}$, `new`$_{(6.140)}$, *ignoredGroup*$_{(9.5)}$, `type`$_{(6.211)}$, *ignoredGroup*$_{(9.5)}$, `parameters`$_{(6.155)}$, *ignoredGroup*$_{(9.5)}$, (`anonymous-class-definition`$_{(6.11)}$)?)

6.106 instanceof

description: represents the java keyword **instanceof**.
type: *keywordType*$_{(7.4)}$
fixed value: **instanceof**

6.107 instanceof-expression

description: reprensents an instanceof expression
attributes:

name:	usage:	type:
`type`	optional	xs:string
`type-ref`	optional	xs:string

positions$_{(9.1)}$
content: (`type-ref`$_{(6.223)}$, `instance`$_{(6.104)}$, *ignoredGroup*$_{(9.5)}$, `instanceof`$_{(6.106)}$, *ignoredGroup*$_{(9.5)}$, `type`$_{(6.211)}$)

6.108 int

description: represents the java keyword **int**.
type: *keywordType*$_{(7.4)}$
fixed value: **int**

6.109 interface

description: represents the java keyword **interface**.
type: *keywordType*$_{(7.4)}$
fixed value: **interface**

6.110 interface-definition

description: represents the declaration of an interface
attributes:

name:	usage:	type:
`fully-qualified-name`	required	xs:string
`id`	optional	xs:string
`is-generic`	optional	xs:boolean
`name`	required	xs:string
`package`	required	xs:string
`visibility`	required	visibility-kind$_{(8.11)}$

positions$_{(9.1)}$

content: (*ignoredGroup*$_{(9.5)}$, *modifiersGroup*$_{(9.6)}$, `interface`$_{(6.109)}$, *ignoredGroup*$_{(9.5)}$, `identifier`$_{(6.96)}$, *ignoredGroup*$_{(9.5)}$, (`type-parameters`$_{(6.222)}$, *ignoredGroup*$_{(9.5)}$)? , (`interfaces`$_{(6.111)}$, *ignoredGroup*$_{(9.5)}$)? , `class-block`$_{(6.37)}$)

6.111 interfaces

description: represents a list of interface, either with keyword implements or extends
attributes:

positions$_{(9.1)}$

content: ((`implements`$_{(6.99)}$ | `extends`$_{(6.80)}$), *ignoredGroup*$_{(9.5)}$, `type`$_{(6.211)}$, *ignoredGroup*$_{(9.5)}$, (`comma`$_{(6.42)}$, *ignoredGroup*$_{(9.5)}$, `type`$_{(6.211)}$, *ignoredGroup*$_{(9.5)}$)*)

6.112 javadoc-member-ref

description: a reference to a member of a type used within a java doc
attributes:

positions$_{(9.1)}$

content: mixed content modell
(*ignoredGroup*$_{(9.5)}$, (`class-qualifier`$_{(6.40)}$, *ignoredGroup*$_{(9.5)}$)? , `member-name`$_{(6.128)}$)*

6.113 javadoc-method-ref

description: represents a reference to a method within a java doc
attributes:

positions$_{(9.1)}$

content: mixed content modell
(*ignoredGroup*$_{(9.5)}$, (`class-qualifier`$_{(6.40)}$, *ignoredGroup*$_{(9.5)}$)? , (`member-name`$_{(6.128)}$, *ignoredGroup*$_{(9.5)}$)+ , (`javadoc-method-ref-parameter`$_{(6.114)}$, *ignoredGroup*$_{(9.5)}$)*)

6.114 javadoc-method-ref-parameter

description: represents a java doc element to represent a parameter to a method ref

attributes:

positions$_{(9.1)}$

content: mixed content modell
(*ignoredGroup*$_{(9.5)}$, (`type-ref`$_{(6.223)}$, *ignoredGroup*$_{(9.5)}$)? , (`parameter-name`$_{(6.154)}$)?)?

6.115 javadoc-tag-element

description: represents any tagged element within a java doc

attributes:

name:	usage:	type:
`kind`	optional	xs:string

positions$_{(9.1)}$

content: mixed content modell
((*ignoredGroup*$_{(9.5)}$ | `javadoc-text-element`$_{(6.116)}$ | `javadoc-tag-element`$_{(6.115)}$ | `javadoc-member-ref`$_{(6.112)}$ | `javadoc-method-ref`$_{(6.113)}$ | `class-qualifier`$_{(6.40)}$ | `member-name`$_{(6.128)}$))*

6.116 javadoc-text-element

description: represents any text element in a java doc comment

attributes:

positions$_{(9.1)}$

content: mixed content modell
(*ignoredGroup*$_{(9.5)}$)*

6.117 labeled-statement

description: prefixes a statement with a label

attributes:

name:	usage:	type:
`label`	optional	xs:string

positions$_{(9.1)}$

content: (`identifier`$_{(6.96)}$, *ignoredGroup*$_{(9.5)}$, `colon`$_{(6.41)}$, *ignoredGroup*$_{(9.5)}$, `statement`$_{(6.185)}$)

6.118 left-brace

description: represents the symbol **{**.

type: *symbolType*$_{(7.6)}$

fixed value: **{**

6.119 left-bracket

description: represents the symbol [.

type: *symbolType*$_{(7.6)}$

fixed value: [

6.120 left-chevron

description: represents the symbol <.
type: *symbolType*$_{(7.6)}$
fixed value: <

6.121 left-hand-side

description: represents the left operand of a binary expression
attributes:
positions$_{(9.1)}$
content: (`expression`$_{(6.77)}$)

6.122 left-parenthesis

description: represents the symbol (.
type: *symbolType*$_{(7.6)}$
fixed value: (

6.123 left-shift

description: represents the symbol <<.
type: *symbolType*$_{(7.6)}$
fixed value: <<

6.124 left-shift-equal

description: represents the symbol <<=.
type: *symbolType*$_{(7.6)}$
fixed value: <<=

6.125 less-equal

description: represents the symbol <=.
type: *symbolType*$_{(7.6)}$
fixed value: <=

6.126 literal-expression

description: a literal in expression context
attributes:

name:	usage:	type:
`kind`	optional	xs:string
`literal`	optional	xs:string
`type`	optional	xs:string
`type-ref`	optional	xs:string

positions$_{(9.1)}$
content: (`type-ref`$_{(6.223)}$, `literal`$_{(??)}$ [in literal-expression])

6.127 long

description: represents the java keyword **long**.
type: *keywordType*$_{(7.4)}$
fixed value: **long**

6.128 member-name

description: represents the name of a member in a java doc comment
attributes:

content: simple content extending xs:string

6.129 member-value-pair

description: represents a pair between member and value used in annotations
attributes:

positions$_{(9.1)}$

content: (`annotation-member-access`$_{(6.8)}$, *ignoredGroup*$_{(9.5)}$, `equal`$_{(6.71)}$, *ignoredGroup*$_{(9.5)}$, `expression`$_{(6.77)}$)

6.130 method-declaration

description: represents a declaration of a method
attributes:

name:	usage:	type:
`id`	optional	xs:string
`is-generic`	required	xs:boolean
`name`	required	xs:string
`signature`	required	xs:string
`visibility`	required	visibility-kind$_{(8.11)}$

positions$_{(9.1)}$

content: (*ignoredGroup*$_{(9.5)}$, *modifiersGroup*$_{(9.6)}$, ((`type-parameters`$_{(6.222)}$, *ignoredGroup*$_{(9.5)}$)? , `return-type`$_{(6.175)}$, *ignoredGroup*$_{(9.5)}$)? , `identifier`$_{(6.96)}$, *ignoredGroup*$_{(9.5)}$, `parameters`$_{(6.155)}$, *ignoredGroup*$_{(9.5)}$, (`left-bracket`$_{(6.119)}$, *ignoredGroup*$_{(9.5)}$, `right-bracket`$_{(6.177)}$, *ignoredGroup*$_{(9.5)}$)* , (`exceptions`$_{(6.73)}$, *ignoredGroup*$_{(9.5)}$)? , (`block`$_{(6.23)}$ | `semicolon`$_{(6.183)}$))

6.131 method-invocation

description: represents an invocation of a constructor
attributes:

name:	usage:	type:
`kind`	optional	xs:string
`method-declaration-ref`	optional	xs:string
`name`	optional	xs:string
`signature`	optional	xs:string

positions$_{(9.1)}$

content: (`type-ref`$_{(6.223)}$, `method-ref`$_{(6.132)}$, (`qualifier-expr`$_{(6.169)}$, *ignoredGroup*$_{(9.5)}$, `dot`$_{(6.60)}$, *ignoredGroup*$_{(9.5)}$)? , (`type-arguments`$_{(6.213)}$, *ignoredGroup*$_{(9.5)}$)? , `identifier`$_{(6.96)}$, *ignoredGroup*$_{(9.5)}$, `parameters`$_{(6.155)}$)

6.132 method-ref

description: represents a reference to a method
attributes:

name:	usage:	type:
`declaring-class-ref`	optional	xs:string
`id`	optional	xs:string
`is-constructor`	optional	xs:boolean
`is-default-constructor`	optional	xs:boolean
`signature`	optional	xs:string

positions$_{(9.1)}$
content: (`return-type`$_{(6.175)}$, `parameters`$_{(6.157)}$ [in method-ref])

6.133 minus

description: represents the symbol `-`.
type: *symbolType*$_{(7.6)}$
fixed value: `-`

6.134 minus-equal

description: represents the symbol `-=`.
type: *symbolType*$_{(7.6)}$
fixed value: `-=`

6.135 minus-minus

description: represents the symbol −.
type: *symbolType*$_{(7.6)}$
fixed value: −

6.136 modifier

description: represents the static modifier for the initializer
attributes:
positions$_{(9.1)}$
content: (`native`$_{(6.139)}$ | `public`$_{(6.167)}$ | `private`$_{(6.165)}$ | `transient`$_{(6.207)}$ | `static`$_{(6.186)}$ | `synchronized`$_{(6.198)}$ | `protected`$_{(6.166)}$ | `final`$_{(6.86)}$ | `abstract`$_{(6.2)}$ | `annotation-modifier`$_{(6.10)}$ | `volatile`$_{(6.232)}$ | `strictfp`$_{(6.187)}$)

6.137 modifier [in initializer-declaration]

description: represents the static modifier for the initializer
attributes:
positions$_{(9.1)}$
content: (`static`$_{(6.186)}$)

6.138 modifiers

description: represents a whitespace separated list of modifiers
attributes:

positions$_{(9.1)}$

content: (`modifier`$_{(6.136)}$, *ignoredGroup*$_{(9.5)}$)+

6.139 native

description: represents the java keyword **native**.
type: *keywordType*$_{(7.4)}$
fixed value: **native**

6.140 new

description: represents the java keyword **new**.
type: *keywordType*$_{(7.4)}$
fixed value: **new**

6.141 newline

description: represents a new line of any kind
attributes:

name:	usage:	type:
`kind`	optional	newline-kind$_{(8.6)}$

content: simple content extending newline-value

6.142 not

description: represents the symbol **!**.
type: *symbolType*$_{(7.6)}$
fixed value: **!**

6.143 not-equal

description: represents the symbol **!=**.
type: *symbolType*$_{(7.6)}$
fixed value: **!=**

6.144 null-literal

description: represents a null literal
attributes:

content: simple content extending nullLiteralType

6.145 number-literal

description: represents a number literal
attributes:

content: simple content extending numberLiteralType

6.146 or

description: represents the symbol |.
type: $symbolType_{(7.6)}$
fixed value: |

6.147 or-equal

description: represents the symbol |=.
type: $symbolType_{(7.6)}$
fixed value: |=

6.148 or-or

description: represents the symbol ||.
type: $symbolType_{(7.6)}$
fixed value: ||

6.149 package

description: represents the java keyword **package**.
type: $keywordType_{(7.4)}$
fixed value: **package**

6.150 package-declaration

description: represents the java package declaration
attributes:

name:	usage:	type:
`name`	required	xs:string

$positions_{(9.1)}$
content: (*ignoredGroup*$_{(9.5)}$, `package`$_{(6.149)}$, *ignoredGroup*$_{(9.5)}$, `package-name`$_{(6.151)}$, *ignoredGroup*$_{(9.5)}$, `semicolon`$_{(6.183)}$)

6.151 package-name

description: represents a java package name
attributes:

name:	usage:	type:
`name`	optional	xs:string

$positions_{(9.1)}$
content: (`identifier`$_{(6.96)}$, *ignoredGroup*$_{(9.5)}$, (`dot`$_{(6.60)}$, *ignoredGroup*$_{(9.5)}$, `identifier`$_{(6.96)}$, *ignoredGroup*$_{(9.5)}$)*)

6.152 parameter

description: represents a parameter to an invocation of either a method or a constructor
attributes:
$positions_{(9.1)}$
content: ((`declared-type-ref`$_{(6.51)}$ [in parameter], `current-type-ref`$_{(6.50)}$ [in parameter], `expression`$_{(6.77)}$) | `expression`$_{(6.77)}$)

6.153 parameter-declaration

description: represents the creation of a parameter
attributes:

name:	usage:	type:
`dimensions`	optional	xs:int
`id`	optional	xs:string
`is-vargs`	optional	xs:boolean
`name`	optional	xs:string

positions$_{(9.1)}$
content: (`type-ref`$_{(6.223)}$, *modifiersGroup*$_{(9.6)}$, `type`$_{(6.211)}$, *ignoredGroup*$_{(9.5)}$, (`ellipsis`$_{(6.62)}$, *ignoredGroup*$_{(9.5)}$)? , `identifier`$_{(6.96)}$, *ignoredGroup*$_{(9.5)}$, (`left-bracket`$_{(6.119)}$, *ignoredGroup*$_{(9.5)}$, `right-bracket`$_{(6.177)}$, *ignoredGroup*$_{(9.5)}$)*)

6.154 parameter-name

description: defines the name of a parameter
attributes:

content: simple content extending xs:string

6.155 parameters

description: describes the types of the parameters
attributes:

positions$_{(9.1)}$
content: (`left-parenthesis`$_{(6.122)}$, *ignoredGroup*$_{(9.5)}$, (*parametersContent*$_{(9.8)}$, *ignoredGroup*$_{(9.5)}$, (`comma`$_{(6.42)}$, *ignoredGroup*$_{(9.5)}$, *parametersContent*$_{(9.8)}$, *ignoredGroup*$_{(9.5)}$)*)* , `right-parenthesis`$_{(6.180)}$)

6.156 parameters [in constructor-ref]

description: describes the types of the parameters
attributes:

positions$_{(9.1)}$
content: (`type-ref`$_{(6.223)}$)*

6.157 parameters [in method-ref]

description: describes the types of the parameters
attributes:

positions$_{(9.1)}$
content: (`type-ref`$_{(6.223)}$)*

6.158 parenthesis-expression

description: represents an expression surrounded with parenthesis
attributes:

name:	usage:	type:
`type`	optional	xs:string
`type-ref`	optional	xs:string

positions$_{(9.1)}$
content: (`type-ref`$_{(6.223)}$, `left-parenthesis`$_{(6.122)}$, *ignoredGroup*$_{(9.5)}$, `expression`$_{(6.77)}$, *ignoredGroup*$_{(9.5)}$, `right-parenthesis`$_{(6.180)}$)

6.159 plus

description: represents the symbol +.
type: *symbolType*$_{(7.6)}$
fixed value: +

6.160 plus-equal

description: represents the symbol +=.
type: *symbolType*$_{(7.6)}$
fixed value: +=

6.161 plus-plus

description: represents the symbol ++.
type: *symbolType*$_{(7.6)}$
fixed value: ++

6.162 postfix-expression

description: represents a postfix expression
attributes:

name:	usage:	type:
gibt den Typ des Literals an.`type-ref`	optional	xs:string

positions$_{(9.1)}$
content: (`type-ref`$_{(6.223)}$, `expression`$_{(6.77)}$, *ignoredGroup*$_{(9.5)}$, (`plus-plus`$_{(6.161)}$ | `minus-minus`$_{(6.135)}$))

6.163 prefix-expression

description: represents a prefix expression
attributes:

name:	usage:	type:
gibt den Typ des Literals an.`type-ref`	optional	xs:string

positions$_{(9.1)}$
content: (`type-ref`$_{(6.223)}$, (`plus-plus`$_{(6.161)}$ | `plus`$_{(6.159)}$ | `minus-minus`$_{(6.135)}$ | `minus`$_{(6.133)}$ | `not`$_{(6.142)}$ | `twiddle`$_{(6.210)}$), *ignoredGroup*$_{(9.5)}$, `expression`$_{(6.77)}$)

6.164 primitive-type [in type]

description: represents any primitive type
attributes:
positions$_{(9.1)}$
content: ((`void`$_{(6.231)}$ | `char`$_{(6.34)}$ | `byte`$_{(6.28)}$ | `short`$_{(6.184)}$ | `int`$_{(6.108)}$ | `long`$_{(6.127)}$ | `float`$_{(6.89)}$ | `double`$_{(6.61)}$ | `boolean`$_{(6.24)}$), *ignoredGroup*$_{(9.5)}$, (`left-bracket`$_{(6.119)}$, *ignoredGroup*$_{(9.5)}$, `right-bracket`$_{(6.177)}$, *ignoredGroup*$_{(9.5)}$)*)

6.165 private

description: represents the java keyword **private**.
type: *keywordType*$_{(7.4)}$
fixed value: **private**

6.166 protected

description: represents the java keyword **protected**.
type: *keywordType*$_{(7.4)}$
fixed value: **protected**

6.167 public

description: represents the java keyword **public**.
type: *keywordType*$_{(7.4)}$
fixed value: **public**

6.168 qualifier

description: represents a qualifier
attributes:
positions$_{(9.1)}$
content: ((`package-name`$_{(6.151)}$, *ignoredGroup*$_{(9.5)}$, `dot`$_{(6.60)}$) | (`type`$_{(6.211)}$, *ignoredGroup*$_{(9.5)}$, `dot`$_{(6.60)}$) | (`expression`$_{(6.77)}$, *ignoredGroup*$_{(9.5)}$, `dot`$_{(6.60)}$) | (`type-ref`$_{(6.223)}$, (`super`$_{(6.189)}$ | `this`$_{(6.202)}$ | `type`$_{(6.211)}$ | `expression`$_{(6.77)}$), *ignoredGroup*$_{(9.5)}$, `dot`$_{(6.60)}$, *ignoredGroup*$_{(9.5)}$, (`super`$_{(6.189)}$, *ignoredGroup*$_{(9.5)}$, `dot`$_{(6.60)}$, *ignoredGroup*$_{(9.5)}$)?))

6.169 qualifier-expr

description: represents an expression that works like a qualifier
attributes:
positions$_{(9.1)}$
content: (`type-ref`$_{(6.223)}$, (`type`$_{(6.211)}$ | `expression`$_{(6.77)}$))

6.170 question

description: represents the symbol **?**.
type: *symbolType*$_{(7.6)}$
fixed value: **?**

6.171 remainder

description:	represents the symbol %.
type:	$symbolType_{(7.6)}$
fixed value:	%

6.172 remainder-equal

description:	represents the symbol %=.
type:	$symbolType_{(7.6)}$
fixed value:	%=

6.173 return

description:	represents the java keyword **return**.
type:	$keywordType_{(7.4)}$
fixed value:	**return**

6.174 return-statement

description:	represents a return statement
attributes:	
	$positions_{(9.1)}$
content:	(`return`$_{(6.173)}$, $ignoredGroup_{(9.5)}$, (`expression`$_{(6.77)}$, $ignoredGroup_{(9.5)}$)? , `semicolon`$_{(6.183)}$)

6.175 return-type

description:	represents a return type for methods
attributes:	
	$positions_{(9.1)}$
content:	(`type-ref`$_{(6.223)}$, `type`$_{(6.211)}$)

6.176 right-brace

description:	represents the symbol }.
type:	$symbolType_{(7.6)}$
fixed value:	}

6.177 right-bracket

description:	represents the symbol].
type:	$symbolType_{(7.6)}$
fixed value:	]

6.178 right-chevron

description:	represents the symbol >.
type:	$symbolType_{(7.6)}$
fixed value:	>

6.179 right-hand-side

description:	represents the right side of a binary expression
attributes:	
	$positions_{(9.1)}$
content:	(`expression`$_{(6.77)}$)

6.180 right-parenthesis

description:	represents the symbol).
type:	$symbolType_{(7.6)}$
fixed value:	)

6.181 right-shift

description:	represents the symbol >>.
type:	$symbolType_{(7.6)}$
fixed value:	>>

6.182 right-shift-equal

description:	represents the symbol >>=.
type:	$symbolType_{(7.6)}$
fixed value:	>>=

6.183 semicolon

description:	represents the symbol **;**.
type:	$symbolType_{(7.6)}$
fixed value:	**;**

6.184 short

description:	represents the java keyword **short**.
type:	$keywordType_{(7.4)}$
fixed value:	**short**

6.185 statement

description:	represents an element that may contain any possible statement
type:	$statementType_{(??)}$

6.186 static

description:	represents the java keyword **static**.
type:	$keywordType_{(7.4)}$
fixed value:	**static**

6.187 strictfp

description:	represents the java keyword **strictfp**.
type:	$keywordType_{(7.4)}$
fixed value:	**strictfp**

6.188 string-literal

description: represents a string literal
attributes:

content: simple content extending stringLiteralType

6.189 super

description: represents the java keyword **super**.
type: $keywordType_{(7.4)}$
fixed value: **super**

6.190 super-class

description: represents the optional super class reference in class declarations
attributes:
$positions_{(9.1)}$
content: (`extends`$_{(6.80)}$, $ignoredGroup_{(9.5)}$, `type`$_{(6.211)}$, $ignoredGroup_{(9.5)}$)

6.191 super-method-invocation

description: represents the invocation of a method of the superclass
attributes:

name:	usage:	type:
`kind`	optional	xs:string
`method-declaration-ref`	optional	xs:string
`name`	optional	xs:string
`signature`	optional	xs:string

$positions_{(9.1)}$
content: (`type-ref`$_{(6.223)}$, `method-ref`$_{(6.132)}$, $ignoredGroup_{(9.5)}$, ((`identifier`$_{(6.96)}$, $ignoredGroup_{(9.5)}$, `dot`$_{(6.60)}$, $ignoredGroup_{(9.5)}$)* | `qualifier`$_{(6.168)}$), `super`$_{(6.189)}$, $ignoredGroup_{(9.5)}$, `dot`$_{(6.60)}$, $ignoredGroup_{(9.5)}$, `identifier`$_{(6.96)}$, $ignoredGroup_{(9.5)}$, `parameters`$_{(6.155)}$)

6.192 super-type-ref [in anonymous-class-definition]

description: represents a reference to a supertype
attributes:
$positions_{(9.1)}$
content: (`type-ref`$_{(6.223)}$)

6.193 super-types [in type-ref]

description: represents a list of supertypes
attributes:
$positions_{(9.1)}$
content: (`type-ref`$_{(6.223)}$)+

6.194 switch

description: represents the java keyword **switch**.
type: *keywordType*$_{(7.4)}$
fixed value: **switch**

6.195 switch-block [in switch-statement]

description: describes the block of a switch statement
type: *caseBlockType*$_{(7.1)}$

6.196 switch-label

description: represents a choice for one of the two labels allowed in switch statements
attributes:
positions$_{(9.1)}$
content: (`case-statement`$_{(6.30)}$ | `default-statement`$_{(6.53)}$)

6.197 switch-statement

description: represents a switch statement
attributes:
positions$_{(9.1)}$
content: (`switch`$_{(6.194)}$, *ignoredGroup*$_{(9.5)}$, `left-parenthesis`$_{(6.122)}$, *ignoredGroup*$_{(9.5)}$, `condition`$_{(6.44)}$, *ignoredGroup*$_{(9.5)}$, `right-parenthesis`$_{(6.180)}$, *ignoredGroup*$_{(9.5)}$, `switch-block`$_{(6.195)}$ [in switch-statement], *ignoredGroup*$_{(9.5)}$)

6.198 synchronized

description: represents the java keyword **synchronized**.
type: *keywordType*$_{(7.4)}$
fixed value: **synchronized**

6.199 synchronized-statement

description: represents a synchronized statement including the synchronzed block
attributes:
positions$_{(9.1)}$
content: (`synchronized`$_{(6.198)}$, *ignoredGroup*$_{(9.5)}$, `left-parenthesis`$_{(6.122)}$, *ignoredGroup*$_{(9.5)}$, `expression`$_{(6.77)}$, *ignoredGroup*$_{(9.5)}$, `right-parenthesis`$_{(6.180)}$, *ignoredGroup*$_{(9.5)}$, `block`$_{(6.23)}$)

6.200 then-clause

description: represents the then part of a conditional expression
attributes:
positions$_{(9.1)}$
content: (`statement`$_{(6.185)}$)

6.201 then-clause [in conditional-expression]

description:	represents the then part of a conditional expression
attributes:	
	positions$_{(9.1)}$
content:	(`expression`$_{(6.77)}$)

6.202 this

description:	represents the java keyword **this**.
type:	*keywordType*$_{(7.4)}$
fixed value:	**this**

6.203 this-expression

description:	represents the expression referencing the current instance
attributes:	
	positions$_{(9.1)}$
content:	(`type-ref`$_{(6.223)}$, (`qualifier`$_{(6.168)}$, *ignoredGroup*$_{(9.5)}$)? , `this`$_{(6.202)}$, *ignoredGroup*$_{(9.5)}$)

6.204 throw

description:	represents the java keyword **throw**.
type:	*keywordType*$_{(7.4)}$
fixed value:	**throw**

6.205 throw-statement

description:	represents a statement to throw an exception
attributes:	
	positions$_{(9.1)}$
content:	(`throw`$_{(6.204)}$, *ignoredGroup*$_{(9.5)}$, `expression`$_{(6.77)}$, *ignoredGroup*$_{(9.5)}$, `semicolon`$_{(6.183)}$)

6.206 throws

description:	represents the java keyword **throws**.
type:	*keywordType*$_{(7.4)}$
fixed value:	**throws**

6.207 transient

description:	represents the java keyword **transient**.
type:	*keywordType*$_{(7.4)}$
fixed value:	**transient**

6.208 try

description:	represents the java keyword **try**.
type:	*keywordType*$_{(7.4)}$
fixed value:	**try**

6.209 try-statement

description: represents the try-catch-statement
attributes:
positions$_{(9.1)}$
content: (`try`$_{(6.208)}$, *ignoredGroup*$_{(9.5)}$, `block`$_{(6.23)}$, *ignoredGroup*$_{(9.5)}$, (`catch-clause`$_{(6.33)}$, *ignoredGroup*$_{(9.5)}$)* , `finally-clause`$_{(6.88)}$)

6.210 twiddle

description: represents the symbol ∼.
type: *symbolType*$_{(7.6)}$
fixed value: ∼

6.211 type

description: represents a type
attributes:

name:	usage:	type:
`dimensions`	optional	xs:int
`kind`	optional	xs:string

positions$_{(9.1)}$
content: ((`at`$_{(6.21)}$, *ignoredGroup*$_{(9.5)}$)? , `type-ref`$_{(6.223)}$, (`wildcard-type`$_{(6.236)}$ | ((`at`$_{(6.21)}$, *ignoredGroup*$_{(9.5)}$)? , `type-name`$_{(6.218)}$, *ignoredGroup*$_{(9.5)}$, (`type-arguments`$_{(6.213)}$, *ignoredGroup*$_{(9.5)}$)? , (`left-bracket`$_{(6.119)}$, *ignoredGroup*$_{(9.5)}$, `right-bracket`$_{(6.177)}$, *ignoredGroup*$_{(9.5)}$)* , (`parameters`$_{(6.155)}$)?) | `primitive-type`$_{(6.164)}$ [in type]))

6.212 type-argument

description: represents a type argument
attributes:
positions$_{(9.1)}$
content: (`type`$_{(6.211)}$)

6.213 type-arguments

description: represents a list of type arguments used to instanciate the type
attributes:
positions$_{(9.1)}$
content: (`left-chevron`$_{(6.120)}$, *ignoredGroup*$_{(9.5)}$, `type-argument`$_{(6.212)}$, *ignoredGroup*$_{(9.5)}$, (`comma`$_{(6.42)}$, *ignoredGroup*$_{(9.5)}$, `type-argument`$_{(6.212)}$, *ignoredGroup*$_{(9.5)}$)* , `right-chevron`$_{(6.178)}$)

6.214 type-bounds [in type-ref]

description: represents bounds on the associated type parameter
attributes:

positions$_{(9.1)}$

content: (`type-ref`$_{(6.223)}$)+

6.215 type-bounds [in type-parameter]

description: represents bounds on the associated type parameter
attributes:

positions$_{(9.1)}$

content: (`extends`$_{(6.80)}$, *ignoredGroup*$_{(9.5)}$, `type`$_{(6.211)}$, *ignoredGroup*$_{(9.5)}$, (`ampersand`$_{(6.3)}$, *ignoredGroup*$_{(9.5)}$, `type`$_{(6.211)}$, *ignoredGroup*$_{(9.5)}$)*)

6.216 type-definition

description: an element surroundig all 4 type declaration elements
attributes:

name:	usage:	type:
`kind`	optional	kind-type$_{(8.5)}$

positions$_{(9.1)}$

content: ((`interface-definition`$_{(6.110)}$ | `enum-definition`$_{(6.70)}$ | `class-definition`$_{(6.39)}$ | `annotation-definition`$_{(6.7)}$), *ignoredGroup*$_{(9.5)}$)

6.217 type-literal

description: represents a type literal
attributes:

positions$_{(9.1)}$

content: (`type`$_{(6.211)}$, *ignoredGroup*$_{(9.5)}$, `dot`$_{(6.60)}$, *ignoredGroup*$_{(9.5)}$, `class`$_{(6.36)}$)

6.218 type-name

description: represents a name for a type with optional qualification
attributes:

positions$_{(9.1)}$

content: ((`qualifier`$_{(6.168)}$, *ignoredGroup*$_{(9.5)}$)? , `identifier`$_{(6.96)}$)

6.219 type-parameter [in type-parameters]

description: represents the declaration of a type parameter
attributes:

name:	usage:	type:
`ref`	optional	xs:string

positions$_{(9.1)}$
content: (`type-ref`$_{(6.223)}$)

6.220 type-parameter

description: represents the declaration of a type parameter
attributes:

name:	usage:	type:
`id`	optional	xs:string
`name`	optional	xs:string

positions$_{(9.1)}$
content: (`type-ref`$_{(6.223)}$, *ignoredGroup*$_{(9.5)}$, `identifier`$_{(6.96)}$, *ignoredGroup*$_{(9.5)}$, `type-bounds`$_{(6.215)}$ [in type-parameter])

6.221 type-parameters [in type-ref]

description: represents a comma separated list of typeparameters within chevrons
attributes:

name:	usage:	type:
`ref`	optional	xs:string

positions$_{(9.1)}$
content: (`type-parameter`$_{(6.219)}$ [in type-parameters])+

6.222 type-parameters

description: represents a comma separated list of typeparameters within chevrons
attributes:

positions$_{(9.1)}$
content: (`left-chevron`$_{(6.120)}$, *ignoredGroup*$_{(9.5)}$, `type-parameter`$_{(6.220)}$, *ignoredGroup*$_{(9.5)}$, (`comma`$_{(6.42)}$, *ignoredGroup*$_{(9.5)}$, `type-parameter`$_{(6.220)}$, *ignoredGroup*$_{(9.5)}$)* , `right-chevron`$_{(6.178)}$)

6.223 type-ref

description: represents a reference to a type
attributes:

name:	usage:	type:
`class-name`	optional	xs:string
`dimensions`	optional	xs:int
`fully-qualified-class-name`	optional	xs:string
`id`	optional	xs:string
`is-anonymous`	optional	xs:boolean
`is-array`	optional	xs:boolean
`is-bounded`	optional	xs:boolean
`is-enum`	optional	xs:boolean
`is-interface`	optional	xs:boolean
`is-type-variable`	optional	xs:boolean
`is-varargs`	optional	xs:boolean
`is-wildcard`	optional	xs:boolean
`kind`	optional	xs:string
`name`	optional	xs:string
`package`	optional	xs:string
`pure-name`	optional	xs:string
`ref`	optional	xs:string
`type-ref`	optional	xs:string

positions$_{(9.1)}$

content: (`super-types`$_{(6.193)}$ [in type-ref], `type-parameters`$_{(6.221)}$ [in type-ref], `type-bounds`$_{(6.214)}$ [in type-ref])

6.224 unsigned-right-shift

description: represents the symbol >>>.
type: *symbolType*$_{(7.6)}$
fixed value: >>>

6.225 unsigned-right-shift-equal

description: represents the symbol >>>=.
type: *symbolType*$_{(7.6)}$
fixed value: >>>=

6.226 variable-access-expression

description: represents the access to a variable
attributes:

name:	usage:	type:
`declaration-ref`	optional	xs:string
`name`	optional	xs:string

positions$_{(9.1)}$

content: (`type-ref`$_{(6.223)}$, `identifier`$_{(6.96)}$)

6.227 variable-declaration

description: the declaration of a simple variable with optional initializer

attributes:

name:	usage:	type:
`dimensions`	optional	xs:int
`id`	optional	xs:string
`is-vargs`	optional	xs:boolean
`name`	optional	xs:string

positions$_{(9.1)}$

content: (`type-ref`$_{(6.223)}$, *ignoredGroup*$_{(9.5)}$, *modifiersGroup*$_{(9.6)}$, (`type`$_{(6.211)}$, *ignoredGroup*$_{(9.5)}$)? , `identifier`$_{(6.96)}$, *ignoredGroup*$_{(9.5)}$, (`left-bracket`$_{(6.119)}$, *ignoredGroup*$_{(9.5)}$, `right-bracket`$_{(6.177)}$, *ignoredGroup*$_{(9.5)}$)* , (`initializer`$_{(6.102)}$, *ignoredGroup*$_{(9.5)}$)?)

6.228 variable-declaration-expression

description: represents an expression that declares one or more variables of the same type

attributes:

positions$_{(9.1)}$

content: (`type-ref`$_{(6.223)}$, *modifiersGroup*$_{(9.6)}$, `type`$_{(6.211)}$, *ignoredGroup*$_{(9.5)}$, `variable-declaration-list`$_{(6.229)}$)

6.229 variable-declaration-list

description: represents a comma separated list of variable declarations

attributes:

positions$_{(9.1)}$

content: (`variable-declaration`$_{(6.227)}$, *ignoredGroup*$_{(9.5)}$, (`comma`$_{(6.42)}$, *ignoredGroup*$_{(9.5)}$, `variable-declaration`$_{(6.227)}$, *ignoredGroup*$_{(9.5)}$)*)

6.230 variable-declaration-statement

description: represents a statement that declares one or more variables of the same type

attributes:

positions$_{(9.1)}$

content: (*ignoredGroup*$_{(9.5)}$, *modifiersGroup*$_{(9.6)}$, `type`$_{(6.211)}$, *ignoredGroup*$_{(9.5)}$, `variable-declaration-list`$_{(6.229)}$, `semicolon`$_{(6.183)}$)

6.231 void

description: represents the java keyword **void**.

type: *keywordType*$_{(7.4)}$

fixed value: **void**

6.232 volatile

description: represents the java keyword **volatile**.
type: *keywordType*$_{(7.4)}$
fixed value: **volatile**

6.233 while

description: represents the java keyword **while**.
type: *keywordType*$_{(7.4)}$
fixed value: **while**

6.234 while-statement

description: represents the while loop
attributes:
positions$_{(9.1)}$
content: (`while`$_{(6.233)}$, *ignoredGroup*$_{(9.5)}$, `left-parenthesis`$_{(6.122)}$, *ignoredGroup*$_{(9.5)}$, `condition`$_{(6.44)}$, *ignoredGroup*$_{(9.5)}$, `right-parenthesis`$_{(6.180)}$, *ignoredGroup*$_{(9.5)}$, `statement`$_{(6.185)}$)

6.235 whitespace

description: represents some whitespace characters. The length of the sequence is stored in an attribute.
attributes:

name:	usage:	type:
`length`	optional	xs:int

content: simple content extending xs:string

6.236 wildcard-type

description: represents a wildcard type that may be bounded by an optional type either from top or bottom
attributes:
name: usage: type:
positions$_{(9.1)}$
content: (`question`$_{(6.170)}$, (*ignoredGroup*$_{(9.5)}$, (`super`$_{(6.189)}$ | `extends`$_{(6.80)}$), *ignoredGroup*$_{(9.5)}$, `type`$_{(6.211)}$)?)

6.237 xor

description: represents the symbol ^.
type: *symbolType*$_{(7.6)}$
fixed value: ^

6.238 xor-equal

description: represents the symbol^=.
type: *symbolType*$_{(7.6)}$
fixed value: ^ =

7 complex types

7.1 caseBlockType

description: represents an element that may contain any possible statement
attributes:
positions$_{(9.1)}$
content: (`left-brace`$_{(6.118)}$, *ignoredGroup*$_{(9.5)}$, ((`switch-label`$_{(6.196)}$ | `statement`$_{(6.185)}$), *ignoredGroup*$_{(9.5)}$)* , `right-brace`$_{(6.176)}$)

7.2 expressionType

description: describes the embedding of a **expression** element
attributes:
positions$_{(9.1)}$
content: (`expression`$_{(6.77)}$)

7.3 identifierType

description: used to represent identifiers in JaML
content: simple content extending identifierTypeContent

7.4 keywordType

description: used to represent java keywords
content: simple content extending xs:string

7.5 literal-type

description: offers a choice of different types of literals
attributes:

name:	usage:	type:
`kind`	optional	xs:string

positions$_{(9.1)}$
content:

7.6 symbolType

description: used to represent symbols
content: simple content extending xs:string

8 simple types

8.1 boolLiteralType

description: describes the java literals for the boolean type.
values: base type for restriction *xs:string*
enumerated values: `true`, `false`

8.2 charLiteralType

description: describes literals for the datatype *char*.
values: base type for restriction *xs:string*

8.3 comment-kind

description: describes the kind of the java comment
values: base type for restriction *xs:string*
enumerated values: `line`, `block`, `javadoc`

8.4 identifierTypeContent

description: type to represent the indentifiers of java in attributes
values: base type for restriction *xs:string*
restricting patterns:

- `[$a-zA-Z_][$a-zA-Z_0-9]*`

8.5 kind-type

description: defines the kind of typedeclaration
values: base type for restriction *xs:string*
enumerated values: `class-definition`, `annotation-definition`, `interface-definition`, `enum-definition`

8.6 newline-kind

description: specifies the operating system of the line break
values: base type for restriction *xs:string*
enumerated values: `unix`, `mac`, `win`

8.7 newline-value

description: Specifies the character sequences representing the newlines ('`0x10`', '`0x13`' and '`0x10 0x13`')
values:

8.8 nullLiteralType

description: describes the **null** literal
values: base type for restriction *xs:string*
enumerated values: `null`

8.9 numberLiteralType

description: describing literals for numbers
values: base type for restriction *xs:string*

8.10 stringLiteralType

description: describes the literals for the datatype *String*
values: base type for restriction *xs:string*
minimal length: 2

8.11 visibility-kind

description: defines the visibility of java element
values: base type for restriction *xs:string*
enumerated values: `public`, `private`, `protected`, `default`

9 groups

9.1 positions

description: defines the positions of the element in the original source code.
contained attributes:

name:	usage:	type:
`pos`	optional	xs:integer
`line`	optional	xs:integer
`col`	optional	xs:integer

9.2 assignment-operators

description: represents a choice of all possible assignment operators
content: (`ampersand-equal`$_{(6.5)}$ | `asterisk-equal`$_{(6.20)}$ | `divide-equal`$_{(6.57)}$ | `equal`$_{(6.71)}$ | `left-shift-equal`$_{(6.124)}$ | `minus-equal`$_{(6.134)}$ | `or-equal`$_{(6.147)}$ | `plus-equal`$_{(6.160)}$ | `remainder-equal`$_{(6.172)}$ | `right-shift-equal`$_{(6.182)}$ | `unsigned-right-shift-equal`$_{(6.225)}$ | `xor-equal`$_{(6.238)}$)

9.3 enhancedForStatementGroup

description: a group to represent a iterating for-statement
content: (`variable-declaration`$_{(6.227)}$, *ignoredGroup*$_{(9.5)}$, `colon`$_{(6.41)}$, *ignoredGroup*$_{(9.5)}$, `expression`$_{(6.77)}$, *ignoredGroup*$_{(9.5)}$)

9.4 forStatementGroup

description: a group to represent a counting for-statement
content: ((`for-init-statement`$_{(6.91)}$ [in forStatementGroup], *ignoredGroup*$_{(9.5)}$)? , `semicolon`$_{(6.183)}$, *ignoredGroup*$_{(9.5)}$, (`condition`$_{(6.44)}$, *ignoredGroup*$_{(9.5)}$)? , `semicolon`$_{(6.183)}$, *ignoredGroup*$_{(9.5)}$, (`for-update-statement`$_{(6.93)}$ [in forStatementGroup], *ignoredGroup*$_{(9.5)}$)?)

9.5 ignoredGroup

description: this group represents all kind of ignorable whitespace in the JaML format
content: (((`newline`$_{(6.141)}$ | `comment`$_{(6.43)}$ | `whitespace`$_{(6.235)}$ | `formfeed`$_{(6.94)}$))*)

9.6 modifiersGroup

description: a group containing an optioal modifiers element
content: ((`modifiers`$_{(6.138)}$, *ignoredGroup*$_{(9.5)}$)?)

9.7 operatorGroup

description:	represents a choice of all possible operators
content:	(`asterisk`$_{(6.19)}$ | `ampersand`$_{(6.3)}$ | `ampersand-ampersand`$_{(6.4)}$ | `ampersand-equal`$_{(6.5)}$ | `divide`$_{(6.56)}$ | `equalequal`$_{(6.72)}$ | `greater-equal`$_{(6.95)}$ | `left-chevron`$_{(6.120)}$ | `left-shift`$_{(6.123)}$ | `less-equal`$_{(6.125)}$ | `minus`$_{(6.133)}$ | `minus-equal`$_{(6.134)}$ | `minus-minus`$_{(6.135)}$ | `not`$_{(6.142)}$ | `not-equal`$_{(6.143)}$ | `or-or`$_{(6.148)}$ | `or`$_{(6.146)}$ | `plus`$_{(6.159)}$ | `remainder`$_{(6.171)}$ | `right-chevron`$_{(6.178)}$ | `right-shift`$_{(6.181)}$ | `unsigned-right-shift`$_{(6.224)}$ | `xor`$_{(6.237)}$)

9.8 parametersContent

description:	a choice of possible parameter contents
content:	(`parameter-declaration`$_{(6.153)}$ | `parameter`$_{(6.152)}$ | `member-value-pair`$_{(6.129)}$ | `expression`$_{(6.77)}$)

References

[1] Greg J. Badros. Javaml: a markup language for java source code. In *Proceedings of the 9th international World Wide Web conference on Computer networks : the international journal of computer and telecommunications networking*, pages 159–177. North-Holland Publishing Co. Amsterdam, The Netherlands, The Netherlands, 2000.

[2] W3C World Wide Web Consortium. Xml path language (xpath) version 1.0. online. http://www.w3org/TR/xpath.

[3] W3C World Wide Web Consortium. Xquery 1.0: An xml query language. online. http://www.w3org/TR/xquery.

[4] H. Eichelberger. Jtransform - a java source code transformation framework. Technical Report 303, University of Würzburg, Institute for Computer Sience, December 2002.

[5] M. Hopfner, D. Seipel, and J. Wolff von Gudenberg. *Comprehending and Visualising Source Code based on XML Representation and Call Graphs.*

[6] Semmle Limited. Semmlecode metric library. online. http://semmle.com/documentation/libraries-of-queries/metrics-library/.

[7] Jonathan I. Maletic, Michael L. Collard, and Adrian Marcus. Source code files as structured documents. In *Program Comprehension, 2002. Proceedings. 10th International Workshop on*, page 289. IEEE Computer Society Washington, DC, USA, 2002.

[8] JDT/Core team. Eclipse java development tools subproject. webpage. `http://www.eclipse.org/jdt/`.

[9] JUnit team. Junit.org - resources for test driven development. online. `http://junit.org/`.