jminusminus

Class JFieldSelection

- java.lang.Object
  - jminusminus.JAST
    - jminusminus.JStatement
      - jminusminus.JExpression
        - jminusminus.JFieldSelection

- All Implemented Interfaces:
  JLhs

```
class JFieldSelection
extends JExpression
implements JLhs
```

The AST node for a field selection operation. It has a target object, a field name, and the Field it defines.

- **Field Summary**

Fields

Modifier and Type	Field and Description
protected JExpression	**target** The target expression.

- **Fields inherited from class jminusminus.JExpression**
  `isStatementExpression, type`
- **Fields inherited from class jminusminus.JAST**
  `compilationUnit, line`

- **Constructor Summary**

Constructors

Constructor and Description
JFieldSelection(int line, AmbiguousName ambiguousPart, JExpression target, String fieldName) Construct an AST node for a field selection having an ambiguous part.
JFieldSelection(int line, JExpression target, String fieldName) Construct an AST node for a field selection without an ambiguous part.

- **Method Summary**

Methods

Modifier and Type	Method and Description
JExpression	**analyze**(Context context) Analyzing a field selection expression involves, (1) reclassifying any ambiguous part, (2) analyzing the target, (3) treating "length" field of arrays specially, or computing the Field object, (4) checking the access rules, and (5) computing the resultant type.
JExpression	**analyzeLhs**(Context context) Analyze the field selection expression for use on the lhs of an assignment.
void	**codegen**(CLEmitter output) Generate the code necessary to load the Rvalue for this field selection.
void	**codegen**(CLEmitter output, String targetLabel, boolean onTrue) The semantics of j-- require that we implement short-circuiting branching in implementing field selections.
void	**codegenDuplicateRvalue**(CLEmitter output) Generate the code required for duplicating the Rvalue that is on the stack becuase it is to be used in a surrounding expression, as in a[i] = x = or x = y--.
void	**codegenLoadLhsLvalue**(CLEmitter output) Generate the code required for setting up an Lvalue, eg, for use in an assignment.
void	**codegenLoadLhsRvalue**(CLEmitter output) Generate the code required for loading an Rvalue for this variable, eg for use in a +=.
void	**codegenStore**(CLEmitter output) Generate the code required for doing the actual assignment.
void	**writeToStdOut**(PrettyPrinter p) Write the information pertaining to this AST to STDOUT.

- **Methods inherited from class jminusminus.JExpression**

  `isStatementExpression, type`

- **Methods inherited from class jminusminus.JAST**

  `line, partialCodegen`

- **Methods inherited from class java.lang.Object**

  `clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

- **Field Detail**

  - **target**

```
protectedJExpression target
```

The target expression.

- **Constructor Detail**

  - **JFieldSelection**

```
publicJFieldSelection(intline,
                      JExpressiontarget,
                      StringfieldName)
```


Construct an AST node for a field selection without an ambiguous part.

Parameters:

`line` - the line number of the selection.
`target` - the target of the selection.
`fieldName` - the field name.

  - **JFieldSelection**


```
publicJFieldSelection(intline,
                      AmbiguousNameambiguousPart,
                      JExpressiontarget,
                      StringfieldName)
```

Construct an AST node for a field selection having an ambiguous part.

Parameters:

`line` - line in which the field selection occurs in the source file.
`ambiguousPart` - the ambiguous part.
`target` - the target of the selection.
`fieldName` - the field name.

- **Method Detail**

  - **analyze**

```
publicJExpressionanalyze(Contextcontext)
```

Analyzing a field selection expression involves, (1) reclassifying any ambiguous part, (2) analyzing the target, (3) treating "length" field of arrays specially, or computing the Field object, (4) checking the access rules, and (5) computing the resultant type.

Specified by:

`analyze` in class `JExpression`

Parameters:

`context` - context in which names are resolved.

Returns:

the analyzed (and possibly rewritten) AST subtree.

- **analyzeLhs**

```
publicJExpressionanalyzeLhs(Contextcontext)
```

Analyze the field selection expression for use on the lhs of an assignment. Although the final keyword is not in j--, we do make use of the Java api and so must repect its constraints.

Specified by:

`analyzeLhs` in interface `JLhs`

Parameters:

`context` - context in which names are resolved.

Returns:

the analyzed (and possibly rewritten) AST subtree.

- **codegen**

```
publicvoidcodegen(CLEmitteroutput)
```

Generate the code necessary to load the Rvalue for this field selection.

Specified by:

`codegen` in class `JAST`

Parameters:

`output` - the code emitter (basically an abstraction for producing the .class file).

- **codegen**

```
publicvoidcodegen(CLEmitteroutput,
                  Stringtargetlabel,
                  booleanonTrue)
```


The semantics of j-- require that we implement short-circuiting branching in implementing field selections.

Overrides:

`codegen` in class `JExpression`

Parameters:

`output` - the code emitter (basically an abstraction for producing the .class file).
`targetLabel` - the label to which we should branch.
`onTrue` - do we branch on true?

- **codegenLoadLhsLvalue**

```
publicvoidcodegenLoadLhsLvalue(CLEmitteroutput)
```

Generate the code required for setting up an Lvalue, eg, for use in an assignment.

Specified by:

`codegenLoadLhsLvalue` in interface `JLhs`

Parameters:

`output` - the code emitter (basically an abstraction for producing the .class file).

- **codegenLoadLhsRvalue**

```
publicvoidcodegenLoadLhsRvalue(CLEmitteroutput)
```

Generate the code required for loading an Rvalue for this variable, eg for use in a +=. Here, this requires either a getstatic or getfield.

Specified by:

`codegenLoadLhsRvalue` in interface `JLhs`

Parameters:

`output` - the code emitter (basically an abstraction for producing the .class file).

- **codegenDuplicateRvalue**

```
publicvoidcodegenDuplicateRvalue(CLEmitteroutput)
```

Generate the code required for duplicating the Rvalue that is on the stack becuase it is to be used in a surrounding expression, as in a[i] = x = or x = y--. Here this means copying it down

Specified by:

`codegenDuplicateRvalue` in interface `JLhs`

Parameters:

`output` - the code emitter (basically an abstraction for producing the .class file).

- **codegenStore**

```
publicvoidcodegenStore(CLEmitteroutput)
```

Generate the code required for doing the actual assignment.

Specified by:

`codegenStore` in interface `JLhs`

Parameters:

`output` - the code emitter (basically an abstraction for producing the .class file).

- **writeToStdOut**

```
publicvoidwriteToStdOut(PrettyWriterp)
```

Description copied from class: JAST

Write the information pertaining to this AST to STDOUT.

Specified by:

`writeToStdOut` in class `JAST`

Parameters:

p - for pretty printing with indentation.