JavaScript is disabled on your browser.

jminusminus

Class JVariable

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

- All Implemented Interfaces:
  JLhs

```
 class JVariable
extends JExpression
implements JLhs
```

The AST node for an identifier used as a primary expression.

- **Field Summary**

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

  `isStatementExpression, type`

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

  `compilationUnit, line`

- **Constructor Summary**

Constructors
Constructor and Description
`JVariable(int line, String name)` Construct the AST node for a variable given its line number and name.

- **Method Summary**

Methods

Modifier and Type	Method and Description
`JExpression`	`analyze(Context context)` Analyzing identifiers involves resolving them in the context.
`JExpression`	`analyzeLhs(Context context)` Analyze the identifier as used on the lhs of an assignment.
`void`	`codegen(CLEmitter output)` Generate code to load value of variable on stack.
`void`	`codegen(CLEmitter output, String targetLabel, boolean onTrue)` The semantics of j-- require that we implement short-circuiting branching in implementing the identifier expression.
`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.
`IDefn`	`iDefn()` Return the identifier's definition.
`String`	`name()` Return the identifier name.
`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`

- ## Constructor Detail

  - ### JVariable

```
publicJVariable(intline,
         Stringname)
```

Construct the AST node for a variable given its line number and name.

Parameters:
: `line` - line in which the variable occurs in the source file.
: `name` - the name.

- ## Method Detail

  - ### name

```
publicStringname()
```

Return the identifier name.

Returns:
: the identifier name.

  - ### iDefn

```
publicIDefniDefn()
```

Return the identifier's definition.

Returns:
: the identifier's definition.

  - ### analyze

```
publicJExpressionanalyze(Contextcontext)
```

Analyzing identifiers involves resolving them in the context. Identifiers denoting fileds (with implicit targets) are rewritten as explicit field selection operations.

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 identifier as used on the lhs of an assignment.

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 code to load value of variable on stack.

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 the identifier expression.

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. Here, this requires nothing; all information is in the the store instruction.

Specified by:

`codegenLoadLhsLvalue` in interface `JLhs`

Parameters:

`output` - the emitter (an abstraction of 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 loading the Rvalue for the variable

Specified by:

`codegenLoadLhsRvalue` in interface `JLhs`

Parameters:

`output` - the emitter (an abstraction of 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 simply duplicating the value on the stack.

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. Here, this requires storing what's on the stack at the appropriate offset.

Specified by:

`codegenStore` in interface `JLhs`

Parameters:

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

- **writeToStdOut**

```
publicvoidwriteToStdOut(PrettyPrinterp)
```

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.