jminusminus

Class JLogicalNotOp

- java.lang.Object
  - jminusminus.JAST
    - jminusminus.JStatement
      - jminusminus.JExpression
        - jminusminus.JUnaryExpression
          - jminusminus.JLogicalNotOp


```
class JLogicalNotOp
extends JUnaryExpression
```

The AST node for a logical NOT (!) expression.

- **Field Summary**
- **Fields inherited from class jminusminus.JUnaryExpression**
  `arg`
- **Fields inherited from class jminusminus.JExpression**
  `isStatementExpression, type`
- **Fields inherited from class jminusminus.JAST**
  `compilationUnit, line`
- **Constructor Summary**

Constructors

Constructor and Description

```
JLogicalNotOp(int line, JExpression arg)
```

Construct an AST for a logical NOT expression given its line number, and the operand.

- **Method Summary**

Methods

Modifier and Type	Method and Description
JExpression	**analyze**(Context context) Analyzing a logical NOT operation means analyzing its operand, insuring it's a boolean, and setting the result to boolean.
void	**codegen**(CLEmitter output) Generate code for the case where we actually want a boolean value (true or false) computed onto the stack, eg for assignment to a boolean variable.
void	**codegen**(CLEmitter output, String targetLabel, boolean onTrue) The code generation necessary for branching simply flips the condition on which we branch.

- **Methods inherited from class jminusminus.JUnaryExpression**
  writeToStdOut

- **Methods inherited from class jminusminus.JExpression**
  isStatementExpression, type

- **Methods inherited from class jminusminus.JAST**
  line, parseCodegen

- **Methods inherited from class java.lang.Object**
  clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

- **Constructor Detail**

  - **JLogicalNotOp**

```
publicJLogicalNotOp(intline,
                    JExpressionarg)
```

Construct an AST for a logical NOT expression given its line number, and the operand.

Parameters:

`line` - line in which the logical NOT expression occurs in the source file.
`arg` - the operand.

- **Method Detail**

  - **analyze**

```
publicJExpressionanalyze(Contextcontext)
```

Analyzing a logical NOT operation means analyzing its operand, insuring it's a boolean, and setting the result to boolean.

Specified by:

`analyze` in class `JExpression`

Parameters:

`context` - context in which names are resolved.

Returns:

the analyzed (and possibly rewritten) AST subtree.

- **codegen**

```
publicvoidcodegen(CLEmitteroutput)
```

Generate code for the case where we actually want a boolean value (true or false) computed onto the stack, eg for assignment to a boolean variable.

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 code generation necessary for branching simply flips the condition on which we branch.

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?