JavaScript is disabled on your browser.

- **Prev Class**
- **Next Class**

- Frames
- No Frames

- All Classes

- Summary:
- Nested |
- Field |
- Constr |
- Method

- Detail:
- Field |
- Constr |
- Method

jminusminus

Class JLiteralChar

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

-

```
class JLiteralChar
extends JExpression
```

The AST node for a char literal.

- **Field Summary**

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

  `isStatementExpression, type`

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

  `compilationUnit, line`

- **Constructor Summary**

Constructors
Constructor and Description
`JLiteralChar``(int line, String text)` Construct an AST node for a char literal given its line number and text representation.

- **Method Summary**

Methods

Modifier and Type	Method and Description
JExpression	**analyze**(Context context) Analyzing a char literal is trivial.
void	**codegen**(CLEmitter output) Generating code for a char literal means generating code to push it onto the stack.
void	**writeToStdOut**(PrettyPrinter p) Write the information pertaining to this AST to STDOUT.

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

  `codegen, 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**

  - **JLiteralChar**

```
publicJLiteralChar(intline,
                   Stringtext)
```

Construct an AST node for a char literal given its line number and text representation.

Parameters:

`line` - line in which the literal occurs in the source file.

`text` - string representation of the literal.

- **Method Detail**

  - **analyze**

```
publicJExpressionanalyze(Contextcontext)
```

Analyzing a char literal is trivial.

Specified by:

`analyze` in class `JExpression`

Parameters:

`context` - context in which names are resolved (ignored here).

Returns:

the analyzed (and possibly rewritten) AST subtree.

  - **codegen**

```
publicvoidcodegen(CLEmitteroutput)
```

Generating code for a char literal means generating code to push it onto the stack.

Specified by:

`codegen` in class `JAST`

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.