

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 JLiteralFalse

- [java.lang.Object](#)
- [jminusminus.AST](#)
- [jminusminus.JStatement](#)
- [jminusminus.JExpression](#)
- [jminusminus.JLiteralFalse](#)

.

Assignment Project Exam Help
<https://powcoder.com>

Add WeChat powcoder

```
class JLiteralFalse  
extends JExpression
```

The AST node for the boolean "false" 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

```
JLiteralFalse(int line)
```

Construct an AST node for a "false" literal given its line number.

- **Method Summary**

Methods

Modifier and Type

Method and Description

JExpression	analyze (Context context) Analyzing a boolean literal is trivial.
void	codegen (CLEmitter output) Generating code for a boolean literal means generating code to push it onto the stack.
void	codegen (CLEmitter output, String targetLabel, boolean onTrue) Generating branch code for a boolean literal is trivial; it's either empty or an unconditional branch.
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**

- **JLiteralFalse**

publicJLiteralFalse(intline)

Construct an AST node for a "false" literal given its line number.

Parameters:

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

- **Method Detail**

- **analyze**

publicJExpressionanalyze(Contextcontext)

Analyzing a boolean 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)

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Generating code for a boolean 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).

- **codegen**

```
public void codegen(CLEmitter output,  
                  String targetLabel,  
                  boolean onTrue)
```

Generating branch code for a boolean literal is trivial; it's either empty or an unconditional 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?

- **writeToStdOut**

```
public void writeToStdOut(PrettyPrinter pp)
```

Description copied from class: `JAST`

Write the information pertaining to this AST to STDOUT.

Specified by:

`writeToStdOut` in class `JAST`

Parameters:

`pp` - for pretty printing with indentation.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

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

- **Frames**
- **No Frames**

- **All Classes**

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

- **Detail:**
- **Field |**
- **Constr |**
- **Method**