

JLiteralTrue.java

```
1  // Copyright 2013 Bill Campbell, Swami Iyer and Bahar Akbal-Delibas
2
3  package jminusminus;
4
5  import static jminusminus.CLConstants.*;
6
7  /**
8   * The AST node for the boolean "true" literal.
9   */
10
11 class JLiteralTrue extends JExpression {
12
13     /**
14      * Construct an AST node for a "true" literal given its line number.
15      *
16      * @param line
17      *         line in which the literal occurs in the source file.
18      */
19
20     public JLiteralTrue(int line) {
21         super(line);
22     }
23
24     /**
25      * Analyzing a boolean literal is trivial.
26      *
27      * @param context
28      *         context in which names are resolved (ignored here).
29      * @return the analyzed (and possibly rewritten) AST subtree.
30      */
31
32     public JExpression analyze(Context context) {
33         type = Type.BOOLEAN;
34         return this;
35     }
36
37     /**
38      * Generating code for a boolean literal means generating code to push it
39      * onto the stack.
40      *
41      * @param output
42      *         the code emitter (basically an abstraction for producing the
43      *         .class file).
44      */
45
46     public void codegen(CLEmitter output) {
47         output.addNoArgInstruction(ICONST_1);
48     }
49
50     /**
51      * Generating branch code for a boolean literal is trivial; it's either
52      * empty or an unconditional branch.
53      *
54      * @param output
55      *         the code emitter (basically an abstraction for producing the
56      *         .class file).
57      * @param targetLabel
58      *         the label to which we should branch.
59      * @param onTrue
60      *         do we branch on true?
61      */
62
63     public void codegen(CLEmitter output, String targetLabel, boolean onTrue) {
64         if (onTrue) {
65             output.addBranchInstruction(GOTO, targetLabel);
66         }
67     }
68 }
```

```
67     }
68
69     /**
70     * @inheritDoc
71     */
72
73     public void writeToStdOut(PrettyPrinter p) {
74         p.printf("<JLiteralTrue line=\"%d\" type=\"%s\"/>\n", line(),
75             ((type == null) ? "" : type.toString()));
76     }
77
78 }
79
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder