

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

JIfStatement.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 an if-statement.
9   */
10
11 class JIfStatement extends JStatement {
12
13     /** Test expression. */
14     private JExpression condition;
15
16     /** Then clause. */
17     private JStatement thenPart;
18
19     /** Else clause. */
20     private JStatement elsePart;
21
22     /**
23      * Construct an AST node for an if-statement given its line number, the test
24      * expression, the consequent, and the alternate.
25      *
26      * @param line
27      *     line in which the if statement occurs in the source file.
28      * @param condition
29      *     test expression.
30      * @param thenPart
31      *     then clause.
32      * @param elsePart
33      *     else clause.
34      */
35
36     public JIfStatement(int line, JExpression condition, JStatement thenPart,
37         JStatement elsePart) {
38         super(line);
39         this.condition = condition;
40         this.thenPart = thenPart;
41         this.elsePart = elsePart;
42     }
43
44     /**
45      * Analyzing the if-statement means analyzing its components and checking
46      * that the test is boolean.
47      *
48      * @param context
49      *     context in which names are resolved.
50      * @return the analyzed (and possibly rewritten) AST subtree.
51      */
52
53     public JStatement analyze(Context context) {
54         condition = (JExpression) condition.analyze(context);
55         condition.type().mustMatchExpected(line(), Type.BOOLEAN);
56         thenPart = (JStatement) thenPart.analyze(context);
57         if (elsePart != null) {
58             elsePart = (JStatement) elsePart.analyze(context);
59         }
60         return this;
61     }
62
63     /**
64      * Code generation for an if-statement. We generate code to branch over the
65      * consequent if !test; the consequent is followed by an unconditional branch
66      * over (any) alternate.
```

```

67      *
68      * @param output
69      *      the code emitter (basically an abstraction for producing the
70      *      .class file).
71      */
72
73      public void codegen(CLEmitter output) {
74          String elseLabel = output.createLabel();
75          String endLabel = output.createLabel();
76          condition.codegen(output, elseLabel, false);
77          thenPart.codegen(output);
78          if (elsePart != null) {
79              output.addBranchInstruction(GOTO, endLabel);
80          }
81          output.addLabel(elseLabel);
82          if (elsePart != null) {
83              elsePart.codegen(output);
84              output.addLabel(endLabel);
85          }
86      }
87
88      /**
89       * @inheritDoc
90       */
91
92      public void writeToStdOut(PrettyPrinter p) {
93          p.printf("<JIfStatement line=\"%d\">\n", line());
94          p.indentRight();
95          p.printf("<TestExpression>\n");
96          p.indentRight();
97          condition.writeToStdOut(p);
98          p.indentLeft();
99          p.printf("</TestExpression>\n");
100         p.printf("<ThenClause>\n");
101         p.indentRight();
102         thenPart.writeToStdOut(p);
103         p.indentLeft();
104         p.printf("</ThenClause>\n");
105         if (elsePart != null) {
106             p.printf("<ElseClause>\n");
107             p.indentRight();
108             elsePart.writeToStdOut(p);
109             p.indentLeft();
110             p.printf("</ElseClause>\n");
111         }
112         p.indentLeft();
113         p.printf("</JIfStatement>\n");
114     }
115 }
116 }
117

```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder