

## JConstructorDeclaration.java

```
1  // Copyright 2013 Bill Campbell, Swami Iyer and Bahar Akbal-Delibas
2
3  package jminusminus;
4
5  import java.util.ArrayList;
6  import static jminusminus.CLConstants.*;
7
8  /**
9   * The AST node for a constructor declaration. A constructor looks very much
10  * like a method.
11  */
12
13  class JConstructorDeclaration extends JMethodDeclaration implements JMember {
14
15      /** Does this constructor invoke this(...) or super(...)? */
16      private boolean invokesConstructor;
17
18      /** Defining class */
19      JClassDeclaration definingClass;
20
21      /**
22       * Construct an AST node for a constructor declaration given the line
23       * number, modifiers, constructor name, formal parameters, and the
24       * constructor body.
25       *
26       * @param line
27       *     line in which the constructor declaration occurs in the source
28       *     file.
29       * @param mods
30       *     modifiers.
31       * @param name
32       *     constructor name.
33       * @param params
34       *     the formal parameters.
35       * @param body
36       *     constructor body.
37       */
38
39      public JConstructorDeclaration(int line, ArrayList<String> mods,
40                                   String name, ArrayList<JFormalParameter> params, JBlock body)
41      {
42          super(line, mods, name, Type.CONSTRUCTOR, params, body);
43      }
44
45      /**
46       * Declare this constructor in the parent (class) context.
47       *
48       * @param context
49       *     the parent (class) context.
50       * @param partial
51       *     the code emitter (basically an abstraction for producing the
52       *     partial class).
53       */
54
55      public void preAnalyze(Context context, CLEmitter partial) {
56          super.preAnalyze(context, partial);
57          if (isStatic) {
58              JAST.compilationUnit.reportSemanticError(line(),
59                                                         "Constructor cannot be declared static");
60          } else if (isAbstract) {
61              JAST.compilationUnit.reportSemanticError(line(),
62                                                         "Constructor cannot be declared abstract");
63          }
64          if (body.statements().size() > 0
65              && body.statements().get(0) instanceof JStatementExpression) {
```

```

67         JStatementExpression first = (JStatementExpression) body
68             .statements().get(0);
69         if (first.expr instanceof JSuperConstruction) {
70             ((JSuperConstruction) first.expr).markProperUseOfConstructor();
71             invokesConstructor = true;
72         } else if (first.expr instanceof JThisConstruction) {
73             ((JThisConstruction) first.expr).markProperUseOfConstructor();
74             invokesConstructor = true;
75         }
76     }
77 }
78
79 /**
80  * Analysis for a constructor declaration is very much like that for a
81  * method declaration.
82  *
83  * @param context
84  *      context in which names are resolved.
85  * @return the analyzed (and possibly rewritten) AST subtree.
86  */
87
88 public JAST analyze(Context context) {
89     // Record the defining class declaration.
90     definingClass =
91         (JClassDeclaration) context.classContext().definition();
92     MethodContext methodContext =
93         new MethodContext(context, isStatic, returnType);
94     this.context = methodContext;
95
96     if (isStatic()) {
97         // offset 0 is used to address "this"
98         this.context.nextOffset();
99     }
100
101     // Declare the parameters. We consider a formal parameter
102     // to be always initialized, via a function call.
103     for (JFormalParameter param : params) {
104         LocalVariableDefn defn =
105             new LocalVariableDefn(param.type(),
106                 this.context.nextOffset());
107         defn.initialize();
108         this.context.addEntry(param.line(), param.name(), defn);
109     }
110     if (body != null) {
111         body = body.analyze(this.context);
112     }
113     return this;
114 }
115
116
117 /**
118  * Add this constructor declaration to the partial class.
119  *
120  * @param context
121  *      the parent (class) context.
122  * @param partial
123  *      the code emitter (basically an abstraction for producing the
124  *      partial class).
125  */
126
127 public void partialCodegen(Context context, CLEmitter partial) {
128     partial.addMethod(mods, "<init>", descriptor, null, false);
129     if (!invokesConstructor) {
130         partial.addNoArgInstruction(ALOAD_0);
131         partial.addMemberAccessInstruction(INVOKE_SPECIAL,
132             ((JTypeDecl) context.classContext().definition())
133                 .superType().jvmName(), "<init>", "()V");
134     }
135     partial.addNoArgInstruction(RETURN);

```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```

136 }
137
138 /**
139  * Generate code for the constructor declaration.
140  *
141  * @param output
142  *      the code emitter (basically an abstraction for producing the
143  *      .class file).
144  */
145
146 public void codegen(CLEmitter output) {
147     output.addMethod(mods, "<init>", descriptor, null, false);
148     if (!invokesConstructor) {
149         output.addNoArgInstruction(ALOAD_0);
150         output.addMemberAccessInstruction(INVOKEVIRTUAL,
151             ((JTypeDecl) context.classContext().definition())
152                 .superType().jvmName(), "<init>", "()V");
153     }
154     // Field initializations
155     for (JFieldDeclaration field : definingClass
156         .instanceFieldInitializations()) {
157         field.codegenInitializations(output);
158     }
159     // And then the body
160     body.codegen(output);
161     output.addNoArgInstruction(RETURN);
162 }
163
164 /**
165  * write to stdout
166  */

```

```

167
168 public void writeToStdOut(PrettyPrinter p) {
169     p.printf("/<JConstructorDeclaration line=\"%d\" " + "name=\"%s\">\n",
170         line(), name);
171     p.indentRight();
172     if (context != null) {
173         context.writeToStdOut(p);
174     }
175     if (mods != null) {
176         p.println("<Modifiers>");
177         p.indentRight();
178         for (String mod : mods) {
179             p.printf("<Modifier name=\"%s\"/>\n", mod);
180         }
181         p.indentLeft();
182         p.println("</Modifiers>");
183     }
184     if (params != null) {
185         p.println("<FormalParameters>");
186         for (JFormalParameter param : params) {
187             p.indentRight();
188             param.writeToStdOut(p);
189             p.indentLeft();
190         }
191         p.println("</FormalParameters>");
192     }
193     if (body != null) {
194         p.println("<Body>");
195         p.indentRight();
196         body.writeToStdOut(p);
197         p.indentLeft();
198         p.println("</Body>");
199     }
200     p.indentLeft();
201     p.println("</JConstructorDeclaration>");
202 }
203
204 }

```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder