

JVariableDeclarator.java

```
1  // Copyright 2013 Bill Campbell, Swami Iyer and Bahar Akbal-Delibas
2
3  package jminusminus;
4
5  /**
6   * The AST node for a variable declarator, which declares a name, its type and
7   * (possibly) provides an initialization.
8   */
9
10 class JVariableDeclarator extends JAST {
11
12     /** Variable name. */
13     private String name;
14
15     /** Variable type. */
16     private Type type;
17
18     /** Variable initializer. */
19     private JExpression initializer;
20
21     public boolean isInitialized;
22
23     /**
24      * Construct an AST node for a variable declarator given the line number,
25      * variable name, its type, and the initializer.
26      *
27      * @param line line in which the variable occurs in the source file.
28      * @param name variable name.
29      * @param type variable type.
30      * @param initializer initializer.
31      */
32     public JVariableDeclarator(int line, String name, Type type,
33                               JExpression initializer)
34     {
35         super(line);
36         this.name = name;
37         this.type = type;
38         this.initializer = initializer;
39     }
40
41     /**
42      * Return the variable name.
43      *
44      * @return the variable name.
45      */
46     public String name() {
47         return name;
48     }
49
50     /**
51      * Return the variable type.
52      *
53      * @return the variable type.
54      */
55     public Type type() {
56         return type;
57     }
58
59 }
```

```

67  /**
68   * Set the declarator's type.
69   *
70   * @param type
71   *       the new type
72   */
73
74  public void setType(Type type) {
75      this.type = type;
76  }
77
78  /**
79   * Return the variable initializer.
80   *
81   * @return the variable initializer.
82   */
83
84  public JExpression initializer() {
85      return initializer;
86  }
87
88  /**
89   * Set the variable initializer.
90   *
91   * @param initial
92   *       initializer.
93   */
94
95  public void setInitializer(JExpression initial) {
96      this.initializer = initial;
97  }
98
99  /**
100   * No analysis is done here.
101   *
102   * @param context
103   *       context in which names are resolved.
104   * @return the analyzed (and possibly rewritten) AST subtree.
105   */
106
107  public JVariableDeclarator analyze(Context context) {
108      // Not used. Analysis is done further up the tree.
109      return this;
110  }
111
112  /**
113   * No code generation is done here.
114   *
115   * @param output
116   *       the code emitter (basically an abstraction for producing the
117   *       .class file).
118   */
119
120  public void codegen(CLEmitter output) {
121      // No code generation here -- possibly up the tree
122      // (for initializations).
123  }
124
125  /**
126   * @inheritDoc
127   */
128
129  public void writeToStdOut(PrettyPrinter p) {
130      p.printf("<JVariableDeclarator line=\"%d\" name=\"%s\" \"
131          + \"type=\"%s\">\n", line(), name, type.toString());
132      p.indentRight();
133      if (initializer != null) {
134          p.println("<Initializer>");
135          p.indentRight();

```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```
136         initializer.writeToStdOut(p);
137         p.indentLeft();
138         p.println("</Initializer>");
139     }
140     p.indentLeft();
141     p.println("</JVariableDeclarator>");
142 }
143
144 }
145
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder