JVariableDeclarator.java

```java
// Copyright 2013 Bill Campbell, Swami Iyer and Bahar Akbal-Delibas

package jminusminus;

/**
 * The AST node for a variable declarator, which declares a name, its type and
 * (possibly) provides an initialization.
 */

class JVariableDeclarator extends JAST {

    /** Variable name. */
    private String name;

    /** Variable type. */
    private Type type;

    /** Variable initializer. */
    private JExpression initializer;

    public boolean isInitialized;

    /**
     * Construct an AST node for a variable declarator given the line number,
     * variable name, its type, and the initializer.
     * 
     * @param line
     *            line in which the variable occurs in the source file.
     * @param name
     *            variable name.
     * @param type
     *            variable type.
     * @param initializer
     *            initializer.
     */

    public JVariableDeclarator(int line, String name, Type type,
            JExpression initializer)

    {
        super(line);
        this.name = name;
        this.type = type;
        this.initializer = initializer;
    }

    /**
     * Return the variable name.
     * 
     * @return the variable name.
     */

    public String name() {
        return name;
    }

    /**
     * Return the variable type.
     * 
     * @return the variable type.
     */

    public Type type() {
        return type;
    }

    /**
     * Set the declarator's type.
     *
     * @param type
     *            the new type
     */

    public void setType(Type type) {
        this.type = type;
    }

    /**
     * Return the variable initializer.
     *
     * @return the variable initializer.
     */

    public JExpression initializer() {
        return initializer;
    }

    /**
     * Set the variable initializer.
     *
     * @param initial
     *            initializer.
     */

    public void setInitializer(JExpression initial) {
        this.initializer = initial;
    }

    /**
     * No analysis is done here.
     *
     * @param context
     *            context in which names are resolved.
     * @return the analyzed (and possibly rewritten) AST subtree.
     */

    public JVariableDeclarator analyze(Context context) {
        // Not used. Analysis is done further up the tree.
        return this;
    }

    /**
     * No code generation is done here.
     *
     * @param output
     *            the code emitter (basically an abstraction for producing the
     *            .class file).
     */

    public void codegen(CLEmitter output) {
        // No code generation here -- possibly up the tree
        // (for initializations).
    }

    /**
     * @inheritDoc
     */

    public void writeToStdOut(PrettyPrinter p) {
        p.printf("<JVariableDeclarator line=\"%d\" name=\"%s\" "
                + "type=\"%s\">\n", line(), name, type.toString());
        p.indentRight();
        if (initializer != null) {
            p.println("<Initializer>");
            p.indentRight();
            initializer.writeToStdOut(p);
            p.indentLeft();
            p.println("</Initializer>");
        }
        p.indentLeft();
        p.println("</JVariableDeclarator>");
    }

}

```