JWhileStatement.java

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

package jminusminus;

import static jminusminus.CLConstants.*;

/**
 * The AST node for a while-statement.
 */

class JWhileStatement extends JStatement {

    /** Test expression. */
    private JExpression condition;

    /** The body. */
    private JStatement body;

    /**
     * Construct an AST node for a while-statement given its line number, the
     * test expression, and the body.
     *
     * @param line
     *            line in which the while-statement occurs in the source file.
     * @param condition
     *            test expression.
     * @param body
     *            the body.
     */

    public JWhileStatement(int line, JExpression condition, JStatement body) {
        super(line);
        this.condition = condition;
        this.body = body;
    }

    /**
     * Analysis involves analyzing the test, checking its type and analyzing the
     * body statement.
     *
     * @param context
     *            context in which names are resolved.
     * @return the analyzed (and possibly rewritten) AST subtree.
     */

    public JWhileStatement analyze(Context context) {
        condition = condition.analyze(context);
        condition.type().mustMatchExpected(line(), Type.BOOLEAN);
        body = (JStatement) body.analyze(context);
        return this;
    }

    /**
     * Generate code for the while loop.
     *
     * @param output
     *            the code emitter (basically an abstraction for producing the
     *            .class file).
     */

    public void codegen(CLEmitter output) {
        // Need two labels
        String test = output.createLabel();
        String out = output.createLabel();

        // Branch out of the loop on the test condition
```

```java
            // being false
            output.addLabel(test);
            condition.codegen(output, out, false);

            // Codegen body
            body.codegen(output);

            // Unconditional jump back up to test
            output.addBranchInstruction(GOTO, test);

            // The label below and outside the loop
            output.addLabel(out);
        }

        /**
         * @inheritDoc
         */

        public void writeToStdOut(PrettyPrinter p) {
            p.printf("<JWhileStatement line=\"%d\">\n", line());
            p.indentRight();
            p.printf("<TestExpression>\n");
            p.indentRight();
            condition.writeToStdOut(p);
            p.indentLeft();
            p.printf("</TestExpression>\n");
            p.printf("<Body>\n");
            p.indentRight();
            body.writeToStdOut(p);
            p.indentLeft();
            p.printf("</Body>\n");
            p.indentLeft();
            p.printf("</JWhileStatement>\n");
        }

    }
```