JExpression.java

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

package jminusminus;

/**
 * The AST node for an expression. The syntax says all expressions are
 * statements, but a semantic check throws some (those without a side-effect)
 * out.
 *
 * Every expression has a type and a flag saying whether or not it's a
 * statement-expression.
 */

abstract class JExpression extends JStatement {

    /** Expression type. */
    protected Type type;

    /** Whether or not this expression is a statement. */
    protected boolean isStatementExpression;

    /**
     * Construct an AST node for an expression given its line number.
     *
     * @param line
     *            line in which the expression occurs in the source file.
     */

    protected JExpression(int line) {
        super(line);
        isStatementExpression = false; // by default
    }

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

    public Type type() {
        return type;
    }

    /**
     * Is this a statementRxpression?
     *
     * @return whether or not this is being used as a statement.
     */

    public boolean isStatementExpression() {
        return isStatementExpression;
    }

    /**
     * The analysis of any JExpression returns a JExpression. That's all this
     * (re-)declaration of analyze() says.
     *
     * @param context
     *            context in which names are resolved.
     * @return the analyzed (and possibly rewritten) AST subtree.
     */

    public abstract JExpression analyze(Context context);

    /**
     * Perform (short-circuit) code generation for a boolean expression, given
     * the code emitter, a target label, and whether we branch to that label on
     * true or on false.
     *
     * @param output
     *            the code emitter (basically an abstraction for producing the
     *            .class file).
     * @param targetLabel
     *            the label to which we should branch.
     * @param onTrue
     *            do we branch on true?
     */

    public void codegen(CLEmitter output, String targetLabel, boolean onTrue) {
        // We should never reach here, i.e., all boolean
        // (including
        // identifier) expressions must override this method.
        System.err.println("Error in code generation");
    }

}

```