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

package jminusminus;

/**
 * The AST node for a formal parameter declaration. All analysis and code
 * generation is done in a parent AST.
 */

class JFormalParameter extends JAST {

    /** Parameter name. */
    private String name;

    /** Parameter type. */
    private Type type;

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

    public JFormalParameter(int line, String name, Type type) {
        super(line);
        this.name = name;
        this.type = type;
    }

    /**
     * Return the parameter's name.
     *
     * @return the parameter's name.
     */

    public String name() {
        return name;
    }

    /**
     * Return the parameter's type.
     *
     * @return the parameter's type.
     */

    public Type type() {
        return type;
    }

    /**
     * Set the type to the specified type.
     *
     * @param newType
     *            the new type.
     * @return return the new type.
     */

    public Type setType(Type newType) {
        return type = newType;
    }
```

```
67
68      /**
69       * No analysis done here.
70       *
71       * @param context
72       *            context in which names are resolved.
73       * @return the analyzed (and possibly rewritten) AST subtree.
74       */
75
76      public JAST analyze(Context context) {
77          // Nothing to do
78          return this;
79      }
80
81      /**
82       * No code generated here.
83       *
84       * @param output
85       *            the code emitter (basically an abstraction for producing the
86       *            .class file).
87       */
88
89      public void codegen(CLEmitter output) {
90          // Nothing to do
91      }
92
93      /**
94       * @inheritDoc
95       */
96
97      public void writeToStdOut(PrettyPrinter p) {
98          p.printf("<JFormalParameter line=\"%d\" name=\"%s\" "
99                  + "type=\"%s\"/>\n", line(), name, (type == null) ? "" : type
100                 .toString());
101     }
102
103 }
104
```