Goal: Modify the handcrafted scanner to support multiline comments; additional tokens (reserved words and operators); and `long` and `double` literals in *j--*.

Zip File: Download and unzip the zip file for the assignment under `$j/j--`.

***Java Lite*:** Consult the *Java Lite* Language Specification for the lexical rules that you must follow when you make changes to the *j--* language described in the problems below.

Problem 1. (*Multiline Comment*) Add support for multiline comment, where all the text from the ASCII characters `/*` to the ASCII characters `*/` is ignored.

```
× ~/workspace/j--
$ ant
$ ./bin/j-- -t scanning/MultilineComment.java
3         : "import" = import
3         : <IDENTIFIER> = java
...
19        : "}" = }
21        : <EOF> = <end of file>
```

Compare your output with the reference output in `scanning/MultilineComment.tokens`.

Problem 2. (*Reserved Words*) Add support for the following reserved words:

`break`	`case`	`continue`	`default`
`double`	`for`	`long`	`switch`

```
× ~/workspace/j--
$ ant
$ ./bin/j-- -t scanning/Keywords.java
1         : "abstract" = abstract
2         : "boolean" = boolean
...
40        : "while" = while
41        : <EOF> = <end of file>
```

Compare your output with the reference output in `scanning/Keywords.tokens`.

Problem 3. (*Operators*) Add support for the following operators:

`-=` `*=` `/=` `%=` `!=` `>=` `<` `||`

```
× ~/workspace/j--
$ ant
$ ./bin/j-- -t scanning/Operators.java
1         : "=" = =
2         : ":" = :
...
24        : "*=" = *=
```

```
25        : <EOF> = <end of file>
```

Compare your output with the reference output in `scanning/Operators.tokens`.

Problem 4. (*Literals*) Add support for long and double literals. You are *not* allowed to use any regular expression library to scan literals.

```
× ~/workspace/j--
$ ant
$ ./bin/j-- -t scanning/IntLiterals.java
1         : <INT_LITERAL> = 0
2         : <INT_LITERAL> = 9
...
5         : <INT_LITERAL> = 1234567890
6         : <EOF> = <end of file>
$ ./bin/j-- -t scanning/LongLiterals.java
1         : <LONG_LITERAL> = 1l
2         : <LONG_LITERAL> = 9L
...
6         : <LONG_LITERAL> = 1234567890L
7         : <EOF> = <end of file>
$ ./bin/j-- -t scanning/DoubleLiterals1.java
1         : <DOUBLE_LITERAL> = 0.
2         : <DOUBLE_LITERAL> = 1.
...
74        : <DOUBLE_LITERAL> = 123456789.e-135D
75        : <EOF> = <end of file>
$ ./bin/j-- -t scanning/DoubleLiterals2.java
1         : <DOUBLE_LITERAL> = .0
2         : <DOUBLE_LITERAL> = .1
...
32        : <DOUBLE_LITERAL> = .098765e-135
33        : <EOF> = <end of file>
$ ./bin/j-- -t scanning/DoubleLiterals3.java
1         : <DOUBLE_LITERAL> = 0e2
2         : <DOUBLE_LITERAL> = 9e9
...
21        : <DOUBLE_LITERAL> = 246e-13D
22        : <EOF> = <end of file>
$ ./bin/j-- -t scanning/DoubleLiterals4.java
1         : <DOUBLE_LITERAL> = 0d
2         : <DOUBLE_LITERAL> = 0D
...
6         : <DOUBLE_LITERAL> = 0987654321D
7         : <EOF> = <end of file>
```

Compare your output with the reference output in `scanning/IntLiterals.tokens`, `scanning/LongLiterals.tokens`, and `scanning/DoubleLiterals*.tokens`.

Files to Submit:

1. `JBinaryExpression.java`
2. `JUnaryExpression.java`

3. `JConditionalExpression.java`
4. `JDoStatement.java`
5. `Parser.java`
6. `Scanner.java`
7. `TokenInfo.java`
8. `notes.txt`

Before you submit your files, make sure:

- Your code is clean, well-organized, uses meaningful variable names, includes useful comments, and is efficient.
- You edit the sections (`#1` mandatory, `#2` if applicable, and `#3` optional) in the given `notes.txt` file as appropriate. In section `#1`, for each problem, state its goal in your own words and describe your approach to solve the problem along with any issues you encountered and if/how you managed to solve those issues.