The goal of this project is to add more Java tokens to the *j--* language. You will only be supporting the scanning of these tokens, which means that the only program files you will be modifying under `$j/j--/src/jminusminus` are `TokenInfo.java` and `Scanner.java`. To compile (just scan for now) your *j--* programs, you need to run the `j--` command as follows:

```
$ $j/j--/bin/j-- -t P.java
```

which only scans `P.java` and prints the tokens in the program along with the line number where each token appears.

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

```
$ $j/j--/bin/j-- -t tests/MultiLineComment.java
5         : public = public
5         : class = class
5         : <IDENTIFIER> = MultiLineComment
5         : { = {
9         : public = public
9         : static = static
9         : void = void
9         : <IDENTIFIER> = main
9         : ( = (
9         : <IDENTIFIER> = String
9         : [ = [
9         : ] = ]
9         : <IDENTIFIER> = args
9         : ) = )
9         : { = {
13        : } = }
14        : } = }
15        : <EOF> = <EOF>
```

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

```
abstract     const        finally        int          public          this
boolean      continue     float          interface    return          throw
break        default      for            long         short           throws
byte         do           goto           native       static          transient
case         double       if             new          strictfp        try
catch        else         implements     package      super           void
char         extends      import         private      switch          volatile
class        final        instanceof     protected    synchronized    while
```

```
$ $j/j--/bin/j-- -t tests/ReservedWords.java
1         : public = public
1         : class = class
1         : <IDENTIFIER> = ReservedWords
1         : extends = extends
1         : <IDENTIFIER> = SomeClass
1         : implements = implements
1         : <IDENTIFIER> = SomeInterface
1         : { = {
2         : public = public
2         : static = static
2         : void = void
2         : <IDENTIFIER> = main
2         : ( = (
2         : <IDENTIFIER> = String
2         : [ = [
2         : ] = ]
2         : <IDENTIFIER> = args
2         : ) = )
2         : { = {
3         : do = do
3         : { = {
5         : } = }
5         : while = while
```

```
5         : ( = (
5         : true = true
5         : ) = )
5         : ; = ;
6         : for = for
6         : ( = (
6         : ; = ;
6         : ; = ;
6         : ) = )
6         : { = {
7         : try = try
7         : { = {
8         : if = if
8         : ( = (
8         : true = true
8         : ) = )
8         : { = {
8         : continue = continue
8         : ; = ;
8         : } = }
9         : } = }
10        : catch = catch
10        : ( = (
10        : <IDENTIFIER> = SomeException
10        : <IDENTIFIER> = e
10        : ) = )
10        : { = {
12        : } = }
13        : } = }
14        : final = final
14        : int = int
14        : <IDENTIFIER> = x
14        : ; = ;
15        : } = }
16        : } = }
17        : <EOF> = <EOF>
```

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

```
?     =       ==     !      ~     !=     /      /=      +       +=       ++     -
-=    --      *      *=     %     %=     >>     >>=     >>>     >>>=     >=     >
<<    <<=     <=     <      ^     ^=     |      |=      ||      &        &=     &&
```

```
$ $j/j--/bin/j-- -t tests/Operators.java
1         : public = public
1         : class = class
1         : <IDENTIFIER> = Operators
1         : { = {
2         : public = public
2         : static = static
2         : void = void
2         : <IDENTIFIER> = main
2         : ( = (
2         : <IDENTIFIER> = String
2         : [ = [
2         : ] = ]
2         : <IDENTIFIER> = args
2         : ) = )
2         : { = {
3         : int = int
3         : <IDENTIFIER> = x
3         : = = =
3         : <INT_LITERAL> = 100
3         : ; = ;
4         : <IDENTIFIER> = x
4         : -= = -=
4         : <INT_LITERAL> = 1
```

```
4         : ; = ;
5         : <IDENTIFIER> = x
5         : %= = %=
5         : <INT_LITERAL> = 7
5         : ; = ;
6         : boolean = boolean
6         : <IDENTIFIER> = y
6         : = = =
6         : <IDENTIFIER> = x
6         : >= = >=
6         : <INT_LITERAL> = 10
6         : || = ||
6         : <IDENTIFIER> = False
6         : ; = ;
7         : int = int
7         : <IDENTIFIER> = z
7         : = = =
7         : <IDENTIFIER> = y
7         : ? = ?
7         : <INT_LITERAL> = 2
7         : : = :
7         : <INT_LITERAL> = 0
7         : ; = ;
8         : } = }
9         : } = }
10        : <EOF> = <EOF>
```

Problem 4. (*Separators*) Add support for the following separators.

```
,    .    [    {    (    )    }    ]    ;    :
```

```
$ $j/j--/bin/j-- -t tests/Separators.java
1         : public = public
1         : class = class
1         : <IDENTIFIER> = Separators
1         : { = {
2         : public = public
2         : static = static
2         : void = void
2         : <IDENTIFIER> = main
2         : ( = (
2         : <IDENTIFIER> = String
2         : [ = [
2         : ] = ]
2         : <IDENTIFIER> = args
2         : ) = )
2         : { = {
3         : switch = switch
3         : ( = (
3         : <IDENTIFIER> = args
3         : [ = [
3         : <INT_LITERAL> = 0
3         : ] = ]
3         : ) = )
3         : { = {
4         : case = case
4         : <STRING_LITERAL> = "1"
4         : : = :
6         : break = break
6         : ; = ;
7         : case = case
7         : <STRING_LITERAL> = "2"
7         : : = :
8         : for = for
8         : ( = (
8         : <IDENTIFIER> = String
8         : <IDENTIFIER> = x
```

```
8         : : = :
8         : <IDENTIFIER> = args
8         : ) = )
8         : { = {
10        : } = }
11        : break = break
11        : ; = ;
12        : default = default
12        : : = :
14        : } = }
15        : } = }
16        : } = }
17        : <EOF> = <EOF>
```

Problem 5. (*Literals*) Add support for (just decimal for now) int, long, float, and double literals.

```
<int_literal> = 0|(1-9) {0-9} // decimal

<long_literal> = <int_literal> (l|L)

<float_literal> = (0-9) {0-9} . {0-9} [(e|E) [+|-] (0-9) {0-9}] (f|F)
                | . {0-9} [(e|E) [+|-] (0-9) {0-9}] (f|F)
                | (0-9) {0-9} [(e|E) [+|-] (0-9) {0-9}] (f|F)
                | (0-9) {0-9} ((e|E) ([+|-] (0-9) {0-9}) (f|F)

<double_literal> = {0-9} [[ . ] {0-9} [(e|E) [+|-] (0-9) {0-9}]] [d|D]
```

```
$ $j/j--/bin/j-- -t tests/Literals.java
1         : public = public
1         : class = class
1         : <IDENTIFIER> = Literals
1         : { = {
2         : public = public
2         : static = static
2         : void = void
2         : <IDENTIFIER> = main
2         : ( = (
2         : <IDENTIFIER> = String
2         : [ = [
2         : ] = ]
2         : <IDENTIFIER> = args
2         : ) = )
2         : { = {
3         : int = int
3         : <IDENTIFIER> = a
3         : = = =
3         : <INT_LITERAL> = 372
3         : ; = ;
4         : long = long
4         : <IDENTIFIER> = b
4         : = = =
4         : <LONG_LITERAL> = 777L
4         : ; = ;
5         : float = float
5         : <IDENTIFIER> = c
5         : = = =
5         : <FLOAT_LITERAL> = 3.14f
5         : ; = ;
6         : double = double
6         : <IDENTIFIER> = d
6         : = = =
6         : <DOUBLE_LITERAL> = 1e-9d
6         : ; = ;
7         : } = }
8         : } = }
9         : <EOF> = <EOF>
```

Files to Submit

1. `j--.zip` (*j*-- source tree as a single zip file)
2. `report.txt` (project report)

Before you submit:

- Make sure you create the zip file `j--.zip` such that it only includes the source files and not the binaries, which can be done on the terminal as follows:

```
$ cd $j/j--
$ ant clean
$ cd ..
$ tar -cvf j--.tar j--/*
$ gzip j--.tar
```

- Make sure the files `$j/j--/lexicalgrammar` and `$j/j--/grammar` are updated with the syntactic changes you have made to the *j*-- language.
- Make sure your report isn't too verbose, doesn't contain lines that exceed 80 characters, and doesn't contain spelling/grammatical mistakes