Goal: Implement type checking and JVM code generation for the programming constructs that were added to *j--* in Assignment 3 (Parsing).

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

Problem 1. (*Operators*) Add support for the following operators:

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

```
>_ ~/workspace/j--
$ ant
$ ./bin/j-- codegen/Operators.java
$ java Operators 23 3
a               : 23
b               : 3
a -= b          : 20
a *= b          : 60
a /= b          : 20
a %= b          : 2
a != b          : true
a >= b          : false
a < b           : true
a < 0 || b < 0 : false
--a             : 1
b++             : 3
```

Problem 2. (*Long and Double Basic Types*) Add support for the `long` and `double` basic types. Use `JLiteralLong` and `JLiteralDouble` as the AST representation for a `long` and `double` literal, respectively.

```
>_ ~/workspace/j--
$ ant
$ ./bin/j-- codegen/Factorial.java
$ java Factorial 7
5040
$ ./bin/j-- codegen/Quadratic.java
$ java Quadratic 1 -5 6
3.0 2.0
```

Problem 3. (*For Statement*) Add support for a for statement.

```
>_ ~/workspace/j--
$ ant
$ ./bin/j-- codegen/ForStatement.java
$ java ForStatement 100
5050
$ ./bin/j-- codege/Stats.java
$ java Stats
Mean   = 5.5
Stddev = 2.8722813232690143
```

Problem 4. (*Break Statement*) Add support for a break statement.

```
>_ ~/workspace/j--
$ ant
$ ./bin/j-- codegen/BreakStatement.java
$ java BreakStatement 1000
168
```

Problem 5. (*Continue Statement*) Add support for a continue statement.

```
>_ ~/workspace/j--
$ ant
$ ./bin/j-- codegen/ContinueStatement.java
$ java ContinueStatement 100
3.121594652591011
```

Problem 6. (*Switch Statement*) Add support for a switch statement.

```
>_ ~/workspace/j--
$ ant
$ ./bin/j-- codegen/SwitchStatement.java
$ java SwitchStatement
Queen of Hearts
$ java SwitchStatement
Jack of Spades
```

Files to Submit:

1. `JArrayExpression.java`
2. `JArrayInitializer.java`
3. `JAssignment.java`
4. `JBinaryExpression.java`
5. `JBooleanBinaryExpression.java`
6. `JBreakStatement.java`
7. `JComparisonExpression.java`
8. `JConditionalExpression.java`
9. `JConstructorDeclaration.java`
10. `JContinueStatement.java`
11. `JDoStatement.java`
12. `JForStatement.java`
13. `JLiteralDouble.java`
14. `JLiteralLong.java`
15. `JMember.java`
16. `JMethodDeclaration.java`
17. `JReturnStatement.java`
18. `JSwitchStatement.java`
19. `JUnaryExpression.java`
20. `JVariable.java`
21. `JVariableDeclaration.java`
22. `JWhileStatement.java`
23. `Parser.java`

24. `Scanner.java`
25. `TokenInfo.java`
26. `notes.txt`

Before you submit your files, make sure:

- Your code follows good programming principles (ie, it is clean and well-organized; uses meaningful variable names; and includes useful comments).
- 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, you must include in nore more than 100 words: a short, high-level description of the problem; your approach to solve it; and any issues you encountered and if/how you managed to solve them.