Homework 3: Name Checking

For this homework, you will extend your RoboLang parser to better handle name checking.

Step 1. Check out the project

In your Subversion repository (from previous homeworks), I have added an Eclipse project named **roboc** (short for Robo Compiler). Using Eclipse, checkout the project.

Within the project are several files:

- The folder **src** – Contains Java source code for the parser. The **RoboLang.g4** file is found with the **edu.memphis.cs.roboc.parser** package files. This g4 file contains only place-holder code.
- A number of ***X*.robo** files – These are input files to use to test your parser. Be warned that I may test your parser on different input when I grade it! Feel free to create additional input files.

To get the project up and running, you must do two things:

1. Do a **Maven -> Update Project** on the project.
2. Copy your hw2 **RoboLang** code into this project's **RoboLang.g4** file.
3. Use ANTLRWorks to generate the Lexer, Parser, and Listener (not Visitor) Java source files.

Step 2. Modify Grammar

Modify your RoboLang.g4 grammar to allow variable declarations within the bodies of robot-behavior declarations.

You may also want to modify your grammar to facilitate the next step.

Step 3. Extend Your Parser (by writing Java code)

Make your parser check for uses of undeclared identifiers, and report errors.

- Identifiers must be declared before they can be used.
- Use block scoping, with the robot-behavior bodies and the bodies of loops and branching conditionals being blocks.
- Do not allow name shadowing. Report any such redeclarations as an error.

To implement this feature, you must use a treewalker similar to the ones we studied in class, a symbol table to keep track of identifier names and their declared scopes, and a scope stack to keep track of the current referencing environment as you walk the tree.

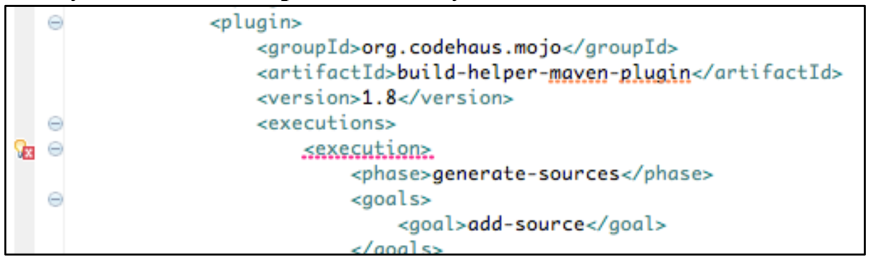
Note: There are no "closed" scopes.

Step 4. Submit your work

To submit, simply commit your files to the SVN repository. Feel free to add/commit your test files as well.

Appendix A: Generating Java Code Using Maven

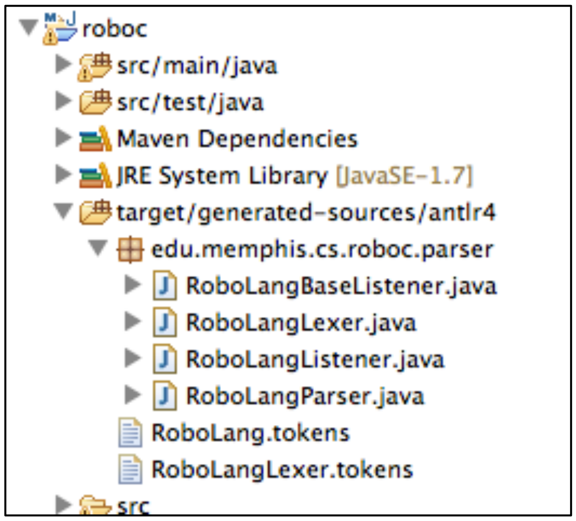
As an additional requirement for this homework, you should switch so that your project uses Maven to generate the Parser, Lexer, and Listener Java files (instead of using ANTLRWorks 2.1). To accomplish this reconfiguration, do as follows:

1. If you generated files using ANTLRWorks 2.1, then delete those files. In particular, these are the files to delete: (1) **RoboLangBaseListener.java**, (2) **RoboLangLexer.java**, (3) **RoboLangListener.java**, (4) **RoboLangParser.java**, (5) **RoboLang.tokens**, (6) **RoboLangLexer.tokens**. (Also, delete any visitor-related files, if you accidentally generated them as well.)
2. Replace your **pom.xml** file with the one at: http://www.cs.memphis.edu/~sdf/comp4040/homeworks/hw3-resources/pom.xml. I noticed that if you open this file in Chrome, Chrome adds some additional text and formatting characters to it. To be on the safe side, don't copy/paste from your browser. Instead, use **File -> Save Page As…**
3. After you save the new **pom.xml** file, you will notice that it contains an error on this line:


```
            <plugin>
                <groupId>org.codehaus.mojo</groupId>
                <artifactId>build-helper-maven-plugin</artifactId>
                <version>1.8</version>
                <executions>
                    <execution>
                        <phase>generate-sources</phase>
                        <goals>
                            <goal>add-source</goal>
                        </goals>
```

   Hover your mouse over the red-underlined word **execution**, and list of quick fixes will appear. Click the **Discover new m2e connections** option. This should cause an **m2e Marketplace** window to appear.
4. Click **Finish**, which selects the **buildhelper** connector by Sonatype, Inc. to be installed. Doing this causes an **Install** wizard to appear.
5. Step through the wizard, and restarting Eclipse at the end.
6. Finally, run **Maven -> Update Project** on your project. Your project should build successfully (assuming it did to begin with).

Notice that the generated code is being placed in a different folder than before – it's now in **target/generated-sources/antlr4/**, as shown in this picture:



```
roboc
  src/main/java
  src/test/java
  Maven Dependencies
  JRE System Library [JavaSE-1.7]
  target/generated-sources/antlr4
    edu.memphis.cs.roboc.parser
      RoboLangBaseListener.java
      RoboLangLexer.java
      RoboLangListener.java
      RoboLangParser.java
    RoboLang.tokens
    RoboLangLexer.tokens
  src
```

Whenever you modify your (.g4) grammar file and want to update the generated Java code, just run **Maven -> Update Project** on your project. Note that editing/saving your grammar file will not automagically cause the generated code to be updated. You have to run the update command manually.

Be aware that the **target/** and **bin/** folders are like temporary scratch folders for Eclipse, where generated files are dropped. These folders should never be committed to the svn repository. Fortunately, your svn repository is currently configured to ignore those folders.

Appendix B: Error messages

If your program detects a name error, it should:

1. Print an error message to standard output (e.g., with **System.out.print()**) that includes
    a. the offending name and
    b. the line number it appears on, and
2. Terminate the program (e.g., with **System.exit(0)**).

Hint on the line-number part: ANTLR's **Token** class has a method **getLine()**. ANTLR's **ParserRuleContext** class (which all the generated ***X*Context** subclasses inherit) has a method **getStart()**, which returns the first **Token** of the context.