Developer Manual

For more information about the checker framework in general, read the manual at
http://types.cs.washington.edu/checker-framework/current/checker-framework-manual.html
or
http://types.cs.washington.edu/checker-framework/current/checker-framework-manual.pdf

Our development website can be found at
http://students.washington.edu/gyhughes/cse403/TeamWebsite/

Our repository can be found at
https://github.com/gyhughes/checker-framework

The rest of our checker documents can be found via our drive folder, at
https://drive.google.com/folderview?id=0ByNy04xRyVnXZWtCUEdPQ05BLU0&usp=sharing

Obtaining Source Code & First Time Building

Obtaining the source code for this project is similar to obtaining the source for the checker-framework project with one difference:

Instead of cloning typetools' checker framework (the 2nd clone command as of 5/12/16), clone our checker instead with the command

```
git clone https://github.com/gyhughes/checker-framework.git
```

After this, just follow the remaining build instructions in the checker-framework manual. Instructions from the checker-framework manual can be found in the
http://types.cs.washington.edu/checker-framework/current/checker-framework-manual.html#build-source

Note: the current version of the checker manual requires you to use mecurial to check out the jsr308-langtools repository. If you do not wish to use mecurial, you can use git instead with a mirror by using the command:
```
git clone https://github.com/wmdietl/jsr308-langtools.git
```
If you have errors building or running, please attempt to clone using mercurial instead.

Re-Build Instructions

If you are using the developer's version of javac as described in the checker-framework's manual, then source files can be compiled using command ant build in the second level checker directory ~/checker-framework/checker/src/org/checkerframework/checker. This can be accomplished with the commands:

```
cd $JSR308/checker-framework/checker/src/org/checkerframework/checker
ant build
```

For Eclipse or other IDE support, see the checker manual's support section here: http://types.cs.washington.edu/checker-framework/current/checker-framework-manual.html#external-tools

Test Instructions

To run the index checker, first make sure that it is added to your PATH variable via first time build instructions, then use the command:

```
javac -processor org.checkerframework.checker.index.IndexChecker <JavaFileName>.java
```

If the checker is not found, try replacing IndexChecker with org.checkerframework.checker.index.IndexChecker

Tests can be run by going into the checker folder of the checker framework and running ant index-tests for indexChecker specific tests or all-tests for the entire test suite. Specifically:

```
cd $JSR308/checker-framework/checker
ant all-tests
(or)
ant index-tests
```

Our test directory can be found at checkerframework/checker/tests/index
To add tests to the suite, simply put a JUnit test in the folder corresponding to the rule(s) you are testing. Any added tests will also be run by the test suite.

Writing a test is simple. Simply write a class that you wish to type check. Then, before lines that should be generating an error or warning, declare what error or warning should exist using the following comment syntax:
//:: error: <*error type*> or //:: warning: <*warning message*>
The tests will not pass if an undeclared error/warning is issued or either a different or no error/warning is generated after an error/warning declaration.

For more information on testing, see the documentation in checker-framework/checker/tests/README

If you have forked our repository, you may integrate Travis-CI automated builds using the .travis.yml script already present in the repository. More information can be found here: https://docs.travis-ci.com/user/getting-started/#To-get-started-with-Travis-CI%3A

Directory Structure

The directory structure of the checker framework is very large, but we are only adding a very small part to it in the form of a new checker. The new source code we are adding can be found in the directory checker-framework/checker/src in the package org.checkerframework.checker.index

The checker's tests can be found in the directory checker-framework/checker/tests. Our tests are located in the folders index-general, index-hierarchy, index-introduction, and index-type

The checker framework's default checkers can be found in the packages adjacent to our index checker. Specifically, in the packages org.checkerframework.checker.<checkerNameHere>. We recommend looking at the classic regex checker as a simple, but nontrivial example of a checker's structure.

If you would like to dive into how the checker itself is working, the checker framework source files can be found in the directory checker-framework/framework/src in the package org.checkerframework

New Version Releases
Our beta release will start at version 0.5, and work its way up by 0.1 until our full release to 1.0 From there, every release will increment by .1 until another major release, to which we will round up to the next whole number.

For developers forking our project, please add an additional . after our number followed by a number to represent your version. For instance if you fork version 0.6, name your first version 0.6.1, then 0.6.2, etc.

Bug Tracking/Resolving
Our issue tracker can be found at https://github.com/gyhughes/checker-framework/issues.
Here, bugs and other issues can be added, tracked, and resolved.
More info on gits bug tracker can be found at https://guides.github.com/features/issues/.

Please include information in the report as detailed by the Checker Framework manual here:
http://types.cs.washington.edu/checker-framework/current/checker-framework-manual.html#reporting-bugs

Coverage Reports
Inside our repository is a coverage report on the Index Checker named jacoco.exec. This coverage report was created using EclEmma's JaCoCo coverage tool. JaCoCo documentation can be found at there website here: http://eclemma.org/

Currently our Travis-CI build is not setup to automatically create a coverage report after every commit. However, a coverage report can be found and downloaded at this link:
http://students.washington.edu/gyhughes/cse403/TeamWebsite/Coverage/

Follow the directions on JaCoCo's website (http://eclemma.org/installation.html) to find and download their Eclipse plugin to import and view our coverage report for each file. If you wish to run and create a new coverage report, information about that process can be found here (http://eclemma.org/userdoc/launching.html) and information about importing and exporting this report can be found here (http://eclemma.org/userdoc/importexport.html).