Setting Up:

1. Make sure mercurial is installed, try “hg version”
2. Create jsr308 directory
3. Make sure java is set up correctly. Use the directions on this page to set up java. However, instead of using OpenJDK use one from Oracle. Don’t set up the code until step 4.
    a. http://types.cs.washington.edu/checker-framework/current/README-jsr308.html#installing-source
    b. I used JDK 7 update 10, I think from this page: http://jdk7.java.net/download.html
4. Follow instructions to build checker-framework from source
    a. http://types.cs.washington.edu/checker-framework/current/checkers-manual.html#build-source
5. Checkout the checker-inference framework into your jsr308 directory
    a. http://code.google.com/p/checker-framework-inference/source/checkout
6. checkout plume-lib into jsr308
    a. http://code.google.com/p/plume-lib/source/checkout
    b. Follow directions in plume-lib/README.html to build the project
7. Check out the verigames repository into jsr308

Opening the code in Eclipse:

1. Create a directory ~/workspaces/verigames
2. Open Eclipse and select the verigames workspace
3. File -> Import
4. General -> Existing Projects into Workspace
5. Choose ~jsr308/verigames/java and click finish. You should see projects for verigames-generation and verigames-translation. There will be build path errors. Right click on verigames-generation -> Build Path -> Configure build path. You should see jsr308-langtools, checker-framework-inference, and checkers listed as missing.
6. Import ~/jsr308/checker-framework-inference into the workspace
7. Import ~/jsr308/jsr308-langtools into the workspace
8. Import ~/jsr308/checker-framework/checkers into the workspace
9. Import ~/jsr308/checker-framework/javaparser into the workspace
10. Ok, now verigames-[generation,translation] are free from build path errors, but checkers and checker-framework-inference still have some. The checker inference framework is missing the annotation-file-utilites and plume-lib
11. Import ~/jsr308/annotation-file-utilites into the workspace, which should contain projects annotation-file-utilites, asmx, and scene-lib
12. Import ~/jsr308/plume-lib. If you see errors, you may need to fix the build path.
    a. Right click ->build path -> configure build path.
    b. Go to the ‘Libraries’ tab.
    c. Add External JARs

    d. select ~/jsr308/jdk1.7.0/lib/tools.jar
    e. Remove plume-lib/jdk/tools.jar
13. At this point there should be no more build path errors.
14. Follow the directions in this video to use the Eclipse Scala plug-in
    a. http://scala-ide.org/download/current.html
15. Select Project -> Build All to be sure everything compiles together correctly

To use Gradle:

At the moment you MUST have the checker-framework built from source as described in:
http://types.cs.washington.edu/checker-framework/current/checkers-manual.html#build-source
You must have the JSR308 and CHECKERS environment variables set as described in the manual.

Download from: http://www.gradle.org/
Note: I have a version with read/execute permissions available at:
/homes/gws/jburke/gradle/current
The executable is merely a script that runs a java jar so you should only have to download, unzip, and add the bin directory to your path.

Please read the comments at the top of the build scripts for more information on running them but a quick description appears below. Gradle works much like Ant with "Tasks" replacing "Targets" and Groovy replacing xml (plus a lot of nice features like dependency resolution).
Note: The exact behavior of these scripts will likely be tweaked in the future. They currently do not support tests.

Checker Framework Inference
gradle dist - copies jars from Langtools, Annotation Tools, and Checker-framework to the dist directory. Builds checker-framework-inference.jar and puts it into the dist directory.

gradle clean - removes all built files including jars

Verigames:
gradle dist - copies jars from checker-framework-inference/dist into verigames/java/dist. Builds verigames.jar and places it in java/dist. Copies the scala jars used to build verigames.jar into dist/scala_lib.

gradle clean - removes built files including jars

gradle infer - Run an inference checker and generate a game world.

-P will pass through a parameter to the Gradle project (NOT the JVM or the task itself).

infChecker - indicates the checker to use (the trusted checker in this case) This parameter currently supports the values "nullness" and "trusted". To add another checker you need only to add it to a map that is laid out in the java/Generation/build.gradle script (it's usage will be obvious).

infArgs - at this point is just a string that is appended to the invocation of TTIRun (which launches the checker). Currently you should put all the files you want checked in this argument in the following manner: "file1.java file2.java file3.java", the spaces and quotations are necessary. In the future it would be nice to change this to a sourcepath argument or something more sophisticated but this will get you up and running for now.

debugInf - That will cause TTIRun to be executed with the variables:
'-Xrunjdwp:transport=dt_socket,server=y,suspend=y,address=5005'
This will let you hook a debugger to port 5005 in order to debug code executed by verigames. When I debug I generally use the following command:

```
gradle --daemon --stacktrace infer -P debugInf=true -P infChecker=trusted
    -P infArgs="/Users/jburke/Documents/tmp/Test.java"
```

Note: gradle -daemon will save you a fair bit of time. I don't use the gradle --debug flag because that adds a bunch of gradle logging that we generally don't want to see.

The build.gradle files in Generation and Translation also has similar tasks.

The build.gradle files in Generation and Translation also has similar tasks.