Removal of Multiply Defined .dummy Symbols Emitted by ecj1

Richard J. Mathar[*]
Max-Planck Institute of Astronomy, Heidelberg, Germany
(Dated: July 15, 2014)

In the ongoing fight against bug https://gcc.gnu.org/bugzilla/show_bug.cgi?id=42143 of the `gcj` compiler, the source code of `GCCMain.java` has been ported back to its state before a dummy insertion in the ZIP entries was introduced, and rebundled with an (obsolete) 3.2 version of the eclipse library.

This back-porting has already been done in some Fedora bundles, but is apparently not making its way into the `gcj` itself—although it should.

I. SOURCES

The source code of the `ecj` library is available from the `eclipse` platform under http://download.eclipse.org/eclipse/downloads/. Because the class `GCCMain.java` which we will patch is not part of that library but a single piece of code from elsewhere, we actually need to move back to version 3.2 of the `eclipse` package because `GCCMain.java` uses a class derived (in the objec-oriented sense) from `Main` of the `eclipse` project, and because the versions 3.3 up to the current 4.4 of `eclipse` use a `SimpleJavaFileObject` class which is not known in the current `gcj` libraries.

We only need the `JDT Core Batch Compiler` that was (then 2006) bundled in `ecjsrc.jar`, taken from http://archive.eclipse.org/eclipse/downloads/drops/R-3.2-200606291905/ and which can be decomposed with

```
jar xf ecjsrc.jar
```

The source code of `GCCMain.java` is from https://github.com/sjnewbury/multilib-overlay/blob/master/dev-java/eclipse-ecj/files/eclipse-ecj-gcj-3.3.0.patch or http://ecj.sourcearchive.com/documentation/3.5.1-1/GCCMain_8java-source.html
and added to the `org/eclipse/jdt/internal/compiler/batch` directory.

Then the file `org/eclipse/jdt/core/JDTCompiler*` is removed because it needs parts of the Apache library which we do not need to deal with.

II. PATCHING GCCMAIN

- The master patch is that the roughly two times ten lines in `GCCMain.java` that argue that the JDK needs at least one entry in the zip stream are removed.
- Because its base class in the old `eclipse` library does not have the `handleWaringToken` member function, the associated call in `GCCMain` is commented out.
- Because its base class in the old `eclipse` library does not have the `disableAll` member function, the call in `GCCMain` is replaced by a `disableWarnings` call.
- Because there is no `setDestinationPath` function in `GCCMain` or its parent class, this is replaced by a simple assignment to `destinationPath`. In a similar manner, the vector `destinationPaths` is just reduced to a single variable, effectively skipping a loop of assignments in `GCCMain`.
- Because there is no variable `maxRepetition` in the base class, the variable `repetitions` is used instead.
- Because the interfae to `AccessRuleSet` differs, the `GCCMain` now uses a simple dummy second argument of `null` at one place.
- Finally, in a simple hack to avoid launching an exception higher up into some tool chain, a try-catch block is cast around a `getCompilationUnit` of the super class.

[*]URL: http://www.mpia.de/~mathar

III. RECOMPILATION

In the top directory, then a JDK compiler is called with

```
javac -cp . org/eclipse/jdt/*/*/*.java org/eclipse/jdt/*/*/*/*.java org/eclipse/jdt/*/*/*/*/*.java
```

and everything is packed with

```
jar cf ecj-3.2.1.jar org/eclipse
```

and made available in http://www.mpia-hd.mpg.de/homes/mathar/progs/ecj-3.2.1.jar.

The patch to the `gcj` then is to move this `ecj-3.2.1.jar` into

```
${HOME}/share/java/ecj-3.2.1.jar
```

to make it available as a "standard" `ecj.jar` with

```
ln -s ecj-3.2.1.jar ecj.jar
```

and to recompile the local `ecj1` with

```
gcj -o$HOME/bin/ecj1 --main=org.eclipse.jdt.internal.compiler.batch.GCCMain $HOME/share/java/ecj.jar
```

supposing that `$HOME/bin` is in the search path while compiling with `gcj`.