**Generating JavaDoc for a Java Project in Eclipse**

1. Select the Java project in the Eclipse Project Manager (only the project title needs to be highlighted)
2. Select “Project” from the Main Menu at the top of the screen. A drop-down menu will appear.
3. Select “Generate Javadoc..” from the drop-down menu.

You now get a pop-up window headed “Javadoc Generation” which allows you to select projects to create Javadoc for, and configuration options for the Javadoc contents.

The “Javadoc command:” input in this pop-up should show the location of a Javadoc.exe program in the same directory as the JDK or JRE your Eclipse is using. I have never needed to change this default but you can if you want to.

You could just select the project(s) and click “Finish” to use the defaults. But the following configuration options should be noted:-

* The radio buttons which allow you to choose one from Private, Package, Protected or Public visibility. The default is “Public” but this may not be appropriate in every case. You must decide how much developers need to know and what they might want to edit.
* The “Destination” input allows you to specify which directory the Javadoc files will be created in. The default is a subdirectory named “doc” under the main project.
* Clicking the “Next” button brings up a checkable list of external libraries referenced by the project. If you check one of these and then double-click the JAR file name, a pop-up appears which allows you to enter a URL for Javadoc related to this library. If you enter an appropriate URL, your generated Javadoc will contain hyperlinks to the Javadoc for the library.

One external Javadoc reference I particularly recommend is adding the core Java Javadoc page (e.g. <https://docs.oracle.com/en/java/javase/11/docs/api/>) to your Java runtime JAR file (which is jrt‑fs.jar in my case). This means there will be hyperlinks to standard Java classes like String, List, Map etc in your generated Javadoc. If you do not include this, your Javadoc will instead contain plain text like “java.util.String” or “java.util.List” with no hyperlink. Given how often these classes are used as method arguments and results, the hyperlinked version looks much neater and more readable.

As we develop the various Planit projects, we will probably need to include Javadoc for other libraries we use like GeoTools and Apache Commons CSV.