Javadoc

Javadoc is Java's application that extracts documentation from a Java program. It is generated in html format so that it can be viewed in webpages. All the documentation for the Java 8 API classes was generated by the javadoc application and placed at this URL: https://docs.oracle.com/javase/8/docs/api/

You can see the first page of such output for our assignment A1 solution on the next page.

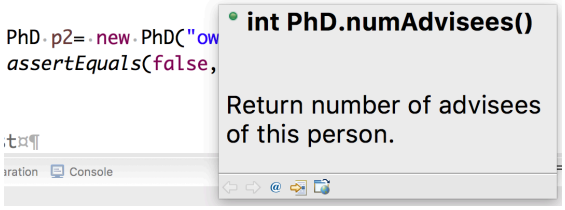
Eclipse uses javadoc comments to help *you* when you are editing Java code, and that's a *good reason* to continually keep your javadoc comments up to date. To the right is method numAdvisees. The javadoc comment on the line before it contains the method's specification.

```
/** Return number of advisees of this person. */
public int numAdvisees() {
    return advisees;
}
```

Suppose you are looking at another class in Eclipse, and you see a line like that to the right, containing a call on numAdvisees. You forgot what numAdvisees does, what its specification is. Your first inclination is to turn to that first class and look at the declaration and spec of numAdvisees. But don't that!

```
assertEquals(0, p1.numAdvisees());
```

Instead, hover your mouse over the function call and wait a second or two. A window pops up, which contains the method signature together with the specification that was in the javadoc comment!



Eclipse will give you the specification of *any* method or public field —static or non-static— as long as the spec is given appropriately in an javadoc comment.

What is a javadoc comment?

A javadoc comment is a comment that begins with "/**" and ends with "*/". If placed *before* a public method declaration (not after, as in Python), a public field declaration, or a public class declaration, it will be extracted (we show how to do this later) and placed as the comment for that method, field, or class. The reader of the javadoc-extracted material will see only that material, and not the source code itself.

What are the rules for writing javadoc comments?

Our main rule is that the javadoc comment precisely, thoroughly, and clearly define the method or variable or class that it describes. We tend to write short, simple, laconic, and clear specifications. For example, our method specs always mention the parameters of the method by name and precisely explain their use. You would do well to be aware of the format and simplicity of our specifications as you read them and try to emulate the style when writing your own specs.

Java does have some guidelines for writing specs. We don't follow them all because they lead to longer more verbose specifications, and ones that are often not as precise. You can find the guidelines on this wikipedia webpage: https://en.wikipedia.org/wiki/Javadoc. We make three remarks.

1. The first sentence of the javadoc specification for a method —defined as up to the first period "."— is expected to be a summary. Only this sentence will appear in the "Method Summaries" of the extracted output, but the whole specificaton will appear in the "Method Detail".
2. Javadoc comments are set in a browser using html. It is best to put the html tag
 at the end of each line in a javadoc comment where you want a line break.
3. Various annotations like @param, @return, @author, and @throws can be used. We don't use them because our specs are clearer and shorter without them. You may get a warning if you don't use @param or @return. Just ignore the warning —or turn such warnings off.

In Eclipse, How do I extract the javadoc specification?

In Eclipse, use menu item Project -> Generate javadoc … . A window will open, asking you for the destination where the files should be placed. Use the default one, which is folder doc in the current project. Click *Finish*.

Javadoc

Look in folder doc in the Eclipse project. Double-click on file index.html, and it will open in the same pane in which java programs appear. You can also look for it in a window on your harddrive and click on it, and it will open in your favorite (default) browser.

When we double-click on file index.html within Eclipse, the window shown below opens: