IC211 Source Code Documentation Requirements

Rules:

1. Comments must conform to the Javadoc standard format are required for all classes, public methods, and public constructors.
2. Comment blocks open with `/**` and close with `*/`
3. The initial comment following each opening `/**` must be formatted in complete English sentences with proper spelling, grammar, and punctuation. It must contain a basic description of the class, method, or constructor and must include sufficient detail to inform the reader of:
   - What happens within the method or constructor.
   - What the class represents and/or the purpose for the class.
4. Any `import` statements must go above the initial Javadoc comment for the class
5. Comments are interpreted as HTML, so you may optionally include HTML formatting if it helps to communicate with the reader. Tips/hints for using HTML in Javadoc:
   - You must use `&lt;`, `&gt;`, and `&amp;`, for "<", ">", and "&" (and other similar items)
   - Other HTML tags work too, like <a>, <em>, <strong>, <u>
   - Lists and tables are also OK, but don't use heading tags
6. The 4 basic @ tags that you are required to use when applicable are:

   - @author
     - Only applies to the class comment. Must be near/at the bottom of the comment block and in the following format:
       `@author LastName, FirstName`

   - @param
     - Applicable for all public constructors and methods that accept parameters as input.
     - An `@param` tag is required for each parameter (even when the description is obvious), formatted as follows:
       `@param parameterName description of parameter`

   - @return
     - Required for all public methods that do not return void, even if it is redundant with the method description. (Whenever possible, find something non-redundant (ideally, more specific) to use for the tag comment), formatted as follows:
       `@return description of the item returned`

   - @throws
     - Required for all public methods whose prototypes end in:
       `throws TypeOfThrowable`, formatted as follows:
       `@throws TypeOfThrowable condition that causes it`

Making your own Javadoc:

To generate the Javadoc from your source code, navigate to the directory where your source code is kept and do the following:

```
javadoc -d doc -link https://docs.oracle.com/en/java/javase/11/docs/api -Xdoclint:all *.java
```

The `-d` option creates a directory named `doc` with the generated HTML files.
The `-link` option creates hyperlinks in the generated HTML files to the Java API documentation.
The `-Xdoclint:all` option ensures that javadoc errors and warnings are output to the user.

Examples:

If you need inspiration or examples for what to say in your comments, you are encouraged to look up the Javadoc for any of the standard Java classes that we've been using all semester.
For example: See the link to the String class.
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/String.html

Other Helpful links:

Javadoc 5.0 Tool homepage
https://docs.oracle.com/javase/1.5.0/docs/guide/javadoc/index.html

Javadoc Tool Reference Pages
https://docs.oracle.com/javase/1.5.0/docs/tooldocs/solaris/javadoc.html

Javadoc style guide
http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html#styleguide