# CSCI-2467 Lab 7 – JavaDoc, Unit Testing, and GitHub

## Background

Read the accompanying documentation and watch the video to understand the little Java game program you are about to download.

* Sam Kass: How to Play Rock, Paper, Scissors, Lizard, Spock - <http://www.samkass.com/theories/RPSSL.html>
* (Video) Big Bang Theory: Rock, Paper, Scissors, Lizard, Spock - <https://www.youtube.com/watch?v=Kov2G0GouBw>

## Assignment

Download the Lab7.zip starter file. Use 7zip to unzip the file using ‘Extract Here’. Open the project folder in IntelliJ.

Examine the code and try running the program. Make sure you completely understand how the program works.

### Step 1 – JavaDoc

Add documentation to the RPSLSpock.java source code file. Make sure that you document the class and all methods in the file. Be sure to use the following annotations:

@author – for the class, list yourself as the author

@param – define each method parameter

@return – explain the return values of each method

Note: these JavaDoc tags and more are documented in the Schildt book in Appendix A.

In IntelliJ, if you enter /\*\* followed by the ENTER key on a blank line preceding a method, IntelliJ will automatically generate a JavaDoc comment template for that method. Be sure and fill in additional information including purpose of the method, purpose of the parameters, and possible return values.

To verify your JavaDoc comments generate (and view) the JavaDoc documentation in an external folder. Examine the documentation and make sure it is correct. If not, fix your documentation and re-generate it. Do not turn in your assignment yet, you are not finished.

### Step 2 – Unit Testing

Look at the unit tests in the RPSLSpockTest folder. Try running the unit tests. Add four additional tests to the isValidPick test method in the RPSLSpockTest class. Add three tests for the other three possible valid method arguments (rock, paper, and scissors) that have not been tested.

The RPSLSpock.isValidPick() method should return false if you pass it an invalid argument (such as the String “banana”). Add a fourth test to verify this. Run your unit tests and make sure they work.

### Step 3 - Extra Credit (2 pts): GitHub (OPTIONAL)

1. Create a (free) account at GitHub. Go to <https://github.com/> and sign-up for an account using your Columbus State e-mail address. Be sure and record your password, as you will need it to complete this exercise.
2. If you are using your home Windows PC, install Git for Windows. See the Word document entitled: “How to Install Git on a Windows PC”.
3. Go to <https://gist.github.com/peterhurford/4d43aa5d6de114c0c741ba664c9c5ff5>

and complete Exercises 1 to 4.

1. Integrate IntelliJ with Git:
   1. Start IntelliJ and open your project.
   2. Click on VCS and then Enable Version Control Integration
   3. Select Git as your version control system from the drop-down menu and click OK.
2. Read the IntelliJ instructions on how to “Share a Project on GitHub”.

<https://www.jetbrains.com/help/idea/manage-projects-hosted-on-github.html>

1. Share your completed Lab 7 project (with your changes for JavaDoc and unit testing) to GitHub. Make sure you commit and push your completed code.
2. Login to github.com with your web browser. Click on your new repository for your Lab 7 project. Copy the URL for the repository. You’ll submit that along with a zipped copy of the Lab 7 project in the next and final step.

### Step 4 – Turn in Your Assignment

You just updated your project with JavaDoc comments and new unit tests. You have verified that the JavaDoc is correct and the new unit tests work. Zip your entire project folder and submit the zip file for grading on Blackboard. If you completed the extra credit portion of the assignment, add a comment to your Blackboard submission with the URL of your GitHub repository. The extra credit part of the assignment is worth an additional 2 points for a total of 12 points.

**Addendum: Making Changes to Your GitHub Project**

Suppose that you've shared your project on GitHub. Now you realize that you need to make some last-minute changes to it.

1) Open IntelliJ with your local copy of the project (the copy of the project that you previously shared).

2) Make and test your changes. Make sure you are satisfied with the changes before sharing them with your team (or the world).

3) Commit your changes: VCS -> Commit. Then enter a comment describing the reason for the changes. Then click on 'Commit'.

4) VCS -> Git -> Push - this will push your changes to the GitHub repository.