# CMSC 132 Week 1 Lab #1

## Notes for TAs

* 1. We do not use CS Department the laptop carts. If a student does not have a laptop, pair up students.
  2. These notes are for you (do not display them nor make them available online).
  3. Make sure that you access the lab material days before the lab (do not wait until the day before as CS servers may be down).
  4. If you see any errors in the material provided, let us know immediately.

## Project

Address any questions about the current project. Remind students to verify they can check out and submit the project.

## Debugger

The code for this exercise can be found at:

<http://www.cs.umd.edu/class/fall2018/cmsc132/labs/Week2/Debugging.zip>

Tell students there will be a quiz associated with the Eclipse debugger (a TA will sit with them and test

whether they know the debugger fundamentals). As you go over the following examples, let the students

work along with you.

1. **Utilities.java** example
   1. Show how to set line numbers by right-clicking on the left column of code page.
   2. Using the Utilities.java example, step through the code without setting any breakpoints. As you step through, show the values of variables.
   3. Set breakpoints and run the program. Show the values of variables and change the stack activation record to the caller of the current method. Display the values of variables in the stack activation records.
   4. Point out the color scheme used to identify private (red) and public (green) elements.
   5. Also illustrate how static variables (DIV in the example) are displayed.
2. **Student.java** example

Using the Student.java, set the specified breakpoints (see example) and show the contents of variables.

1. **Course.java** example

Using the Course.java set the specified breakpoints (see example) and show the contents of arrays and ArrayList objects.

## Pair-Programming Exercise

* 1. Students should practice the debugger in pairs.
  2. Do not allow a student to work by herself/himself. One goal of this exercise is for students to work together. It is fine to create groups of three students if you don’t have an even number of students.
  3. Ask students to practice the concepts discussed earlier. Also, they can write a new program and debug it.
  4. Remind students they should not wait to have problems with their code in order to learn the debugger. They should learn the debugger with code they are familiar with and that works.

## In-Lab Office Hours

Use any time left to hold office hours.