CS 158 Lab 5 January 28, 2010
Exceptions and Timing

1. Open the project from lab 2 called Shapes. Copy your **Rectangle** class to a new package called **original**. The only purpose of this is to give you a copy of your original **Rectangle** class from lab 2 that will not be changed in this lab. Go to the class web site at http://faculty.valpo.edu/jcaristi/cs158 and follow the link to chapter 2 files. Copy and paste the contents of the file **Rectangle.java** into the class **Rectangle** in your project in the package with the rest of the classes (not **original** – ask if this does not make sense). This is the file the author of our text uses in Chapter 2. Read through it carefully to see what the author has changed. Run your tests. One of them should fail because the author made a mistake in one of the methods. Find and fix the error. You may also have to fix another error having to do with a clone. Just have Netbeans fix the error by adding the appropriate "throws" statement.

2. Add a main method to **Rectangle** that exercises (tests) the new exception code in **setLength**. In other words, do something to generate the exception and see what happens.

3. From the chapter 2 files, look at the author's **RectangleTester**. Open it in a web browser so that you can see the code, but do not put it into Netbeans as a class. Read Test Case 3 carefully. It has to do with testing for illegal length. Make sure you understand it. In your **RectangleTest** class, create private **Rectangle** objects **r1** and **r2** the way the author does, copy in his **TestInstantiateIllegalLength** method, and run the tests to make sure they pass.

4. From the chapter 2 files, look at **Problem_2_3.java**. Create a class for it. Study it. What does it do? Write a JUnit test that confirms your answer.

5. Read the text pages 156-158. They have to do with timing how long it takes to access elements of two-dimensional arrays. Although we have not talked about 2D arrays in Java, you don't need to know much to understand what he's talking about. The problem has to do with the way the arrays are stored in memory, the way caching takes place, and how that affects performance. Download the code from **ArrayCopyTimed.java** from the chapter 2 files into a new class in your project. Your task is to reproduce a version of the table on page 160 (your numbers will differ, and they will even differ if you re-run the program). There is one catch. This program uses "command line arguments". Remember that stuff in the parentheses in the main methods? Those are command line arguments as an array of **String**s. This program expects the array size to be passed in as a command line argument when the program is run. You could do this from the command line if you were running from a terminal window in the directory where the class files are located by typing:

   java ArrayCopyTimed 500

   The value 500 is passed in as **args[0]** and becomes the array size. However, you can do this in Netbeans by first setting the Main Project to be Shapes from the Run menu. Then right click on the project name, Shapes, choose Properties, then under the Category "Run", set the Main class as **ArrayCopyTimed** and put 500 in the slot for Arguments. Then from the Run menu you must run the Main Project