Laboratory 14

CS 1323, Spring 2015

# Learning Objectives

1. Create a UML class diagram using umlet in eclipse[[1]](#footnote-1) (10 points).
2. Implement a class with instance data. (10 points)
3. Implement a constructor. (10 points)
4. Implement accessor and mutator methods. (10 points)
5. Write a main program that tests every method in a class and is stored in a separate class. The contents of this method may match that in objective #6. (20 points)
6. Javadoc the class that contains instance data. (20 points)

10 points will be awarded for the documentation of your program (outside of Javadoc that is assessed separately). That means using good names for variables, proper and consistent indentation of code, and meaningful use of whitespace.

Section 10: When your program is completed and running, have the teaching assistants check it to get credit for the lab. If you do not submit your assignment during the laboratory, it will be due on Wednesday, April 29 by 11:59 p.m.

Section 1: The lab is due by Wednesday, April 29 by 11:59 p.m.

The class that is developed in this laboratory will also be used in Laboratory 13.

# Description

In this laboratory we’ll come full circle to where we started the semester. We started by selling three items of makeup on an e-commerce site. We’re going to write a more complete version of the program, featuring our new found knowledge of object-oriented program. For this week, we will develop and test a class that stores items. Next week, we will combine many items into a more realistic website.

Items on the website will have a name, a price, and a quantity that are in stock. The name and price cannot be changed, although the quantity in stock can. New items might have a name and a price, or a name, price and quantity in stock. You’ll want a method that represents each item as a String, to use in testing. The name of this method should follow the standard Java convention (toString()).

Our ultimate website will allow the company to list the inventory, sell items to customers, and restock items from a fixed inventory. When you design your class to store items, make sure that you provide methods that will be necessary to make these operations work next week.

To test the methods, you should make sure that every method has been run at least once, and that the results that are returned match your expectations.

# Javadoc

If you’ve watched the Janux videos at all, you’ll notice that I’ve been creating Javadoc for all of the code for a couple of months. It’s your turn now to start creating Javadoc to fully document your programs.

At the start of your class there is a comment that describes the class and attributes authorship.

/\*\* This class does whatever it does. More is generally better when it comes to documentation.

\* @author Author’s name

\* @version 1.0

\*/

The rules for documenting methods are summarized below.

/\*\* First sentence that describes what the method does. The first sentence is the one that shows up in \* the table, so it needs to be informative.

\* @param firstParameterName Description of the first parameter.

\* @param secondParameterName Description of the second parameter.

\* @return Description of exactly what the method returns.

\*/

After you have written Javadoc, you’ll notice that eclipse will start showing the documentation of your classes the same way that it shows classes in the API. You’ll also notice that the Javadoc tab next to the Console will start showing your Javadoc.

1. The latest version of umlet has been freezing eclipse on Macs occasionally. If this happens in lab, students will be given full credit for the laboratory with a handwritten UML diagram. [↑](#footnote-ref-1)