Assignment #4

Practice with Class Design and JUnit

CS 3354.251 and 252 Spring 2017
Instructor: Jill Seaman

Due: before class **Wednesday, 3/29/2017**
Submit a "hard copy" (hand-written or computer-generated) in class for part I.
Upload electronic copy by 10:30am for part II.

Part I Class Design

A preliminary (incomplete) version of a Library Management System is available on TRACS (lms.zip). In that code, find ONE example of each of the problems listed below AND provide new code to fix the problem.
Note: do not change the Day class.

1. Sharing Mutable References (unintentionally).
2. Not Separating Accessors and Mutators.
3. Side Effects.
4. Violating the Law of Demeter (aka "Sharing Mutable References intentionally").

Part II JUnit

Write some tests using JUnit for the Library Management System. Download and install the Library Management System into an eclipse project (or use another IDE, or the command line). Note: do not change the Day class.

Include test cases for the following:

1. Test the **method Resource.calculateFine(Day)**. You should have at least 4 test cases, including the "exceptional cases".

2. Test the **Collection class** (this is the list of Resources owned by the Library). For this assignment you need test only the addResource(Resource) method. You should include a successful and an unsuccessful add operation. Note that you are testing the collaboration of the addResource and findResource methods.

3. Test the **collaborating classes** Member and Resource as they carry out the **Resource.calculateNewDueDate()** operation. You should have 4 tests for this, one for each combination of a Member subclass and a Resource subclass.

NOTES:

This assignment may be done with a partner.

For Part II JUnit:

- Use the package "assign4" for your classes and put your files in the appropriate directory structure. Submit ONLY your test case java files!! (your files may need to import lms.*;).

- Each test case must be in its own method. More specifically,
  each @Test method may contain only ONE assert... method call!!

- Make sure your test will fail if the code is broken. Just because your test passes does not mean it is a good test. After you have written your test, consider changing the code to introduce a bug that you think your code should catch.

- Follow the style guidelines from the class website. Javadoc comments are not required, but good comments describing each test method are required.

Submit:

Please submit your *.java files in a single zip file (assign4_xxxxxx_yyyyyy.zip). The xxxxxx and yyyyyy are your TX State NetIDs (mine is js236, You may have two, one for each partner. If you are working solo, you will only have one).

Submit:
Part I: a hard copy at the beginning of class on the due date.
Part II: an electronic copy only, using the Assignments tool on the TRACS website for this class. If you are working in a pair, submit using the TRACS account of just ONE member of your partnership.