CS3354 – Fall 2017 – Assignment 4

Due date: Thursday, Nov. 2, 2017 at 11:55 pm.

Goal:

The goal of this assignment is to help students understand the use of JUnit to test Java code.

Description:

In this assignment you will create a set of unit tests, to test the behavior of the code written for Assignment 1.

To keep things consistent, please use the solution to Assignment 1 provided by the instructor. You can find the solution on TRACS, under *Resources > Assignment-related material > Assignment 1 - solution*.

***Note**: Updated version of Assignment 1 solution has been uploaded to make testing easier. Use the updated version.*

A unit test is an automated piece of code that invokes a unit of work in the system and then checks a single assumption about the behavior of that unit of work. A unit of work is a single logical functional use case in the system that can be invoked by some public interface (in most cases). A unit of work can span a single method, a whole class or multiple classes working together to achieve one single logical purpose that can be verified.

Think of unit testing as a way to test the behavior of the code (or parts of the code) written, without actually having to run the program. For example, in the case of assignment 1, assume that the front-end (MainApp console user interface) part of the program and the back-end part of the program (ShippingStore database management) are written by two different developers. How would the developer of the back-end be able to ensure that the code he/she has written works correctly without having access to the front-end?

In this assignment, you are asked to create JUnit tests to test the class ShippingStore, including all its methods, written for Assignment 1.

First you should consider testing the behavior of the class/methods under normal operation scenarios.

For example, to test the method findPackageOrder(String trackingNumber) of the class ShippingStore, you may need to create a test method, which creates a mock PackageOrder object and adds it to the packageOrerList before the method findPackageOrder can be called to search for it. To ensure that everything worked as planned, you can then search for the package using its trackingNumber and see if the correct package is found. Mock objects can be created either in the same test method or before any test methods are run, using the @BeforeClass annotation.

Subsequently, you can consider creating test cases for unusual scenarios, e.g. when a certain input or behavior is expected to cause an exception to be thrown, or when a user input is not as expected.

At the end create a TestRunner class, which has a main method that runs the unit tests that you have created, and prints out the test results.

Do not modify the code of the solution to Assignment 1 provided.

Tasks:

1. Implement the JUnit tests to test only the class **ShippingStore.java**, including all its methods. Try to be creative by coming up with test cases that can test as many different situations as possible. *You don't need to test the classes PackageOrder.java and MainApp.java.*

2. Use a standard Java coding style to improve your program's visual appearance and make it more readable. I suggest the BlueJ coding style: http://www.bluej.org/objects-first/styleguide.html

3. Use Javadoc to document your code.

Logistics:

This assignment will be done and submitted **individually** by each student.
Submit your test code in a single file (assign4_xxxxxx.zip). The xxxxxx is your TX State NetID.

Submit an electronic copy only, using the Assignments tool on the TRACS website for this class. Do NOT include executable or .class files in your submission.