Learning Activity: JUnit Testing

Back

Educational Objectives

- Gain familiarity with JUnit testing.

Instructions

JUnit is a commonly-used library for unit testing in Java. For this lab, you will create unit tests for a very simple class hierarchy. While there are some errors in the provided hierarchy, the goal of this activity is *not* correcting those errors. Instead, the goal is the creation of *test cases*, and therefore identifying the errors.

Starting point code

Download the starting point files.

- CSC143Shape.java
- CSC143Circle.java
- CSC143Rectangle.java
- CSC143Triangle.java

Some notes:

- Create three separate classes, one to test each of the concrete classes: CSC143Circle, CSC143Rectangle, CSC143Triangle.
- The test cases should be created using JUnit 4.
- Name the classes for the test cases csc143.test.junit.TestCSC143Circle, csc143.test.junit.TestCSC143Rectangle, and csc143.test.junit.TestCSC143Triangle, respectively. Notice that the unit test classes and the classes that are being tested are in different packages. This is typical. The unit testing code is generally not considered a deliverable for a project. It's for "internal use" only. Therefore the documentation needs are generally lighter.
- It is laudable, but not required that JavaDoc comments be prepared for each unit test method. However, each method should have some minimal comments. The comment can simply identify what is being testing in the given test method.
- Now, there may be times where the development and delivery of testing code is part of the contract.
- Remember, each test method should be designed to perform only a single test.
- Create an ASCII plain-text document listing any errors that your test cases found.
    - Include the constructor or method tested, the nature of the error, and the JUnit test method(s) that identified the error.
    - This file shall have the standard header for text files.

JUnit 4

- No inheritance requirement, instead annotations.
- Individual test cases: `@org.junit.Test public void` *`xxxx`*`()`
  modifier: `public`
  return type: `void`
  no parameters: `()`

- Assert methods are static within `org.junit.Assert`
- Typically, imports:
  `import org.junit.*;`
  `import static org.junit.Assert.*;`
- Test environment annotations:
  `@Before public void` *`xxxx`*`()`
  `@After public void` *`xxxx`*`()`

Submission

- One (1) Java archive file (.jar) with source and byte code for:
  - csc143/test/junit/TestCSC143Circle
  - csc143/test/junit/TestCSC143Rectangle
  - csc143/test/junit/TestCSC143Triangle
- One (1) ASCII, plain-text file listing any errors detected by your JUnit tests.

Note: There is no entry point for this code. That is, the jar file does not need to be runnable.

Evaluation Criteria

Minus:

- The jar contents are correct: file names and folder structure.
- The JUnit test files can be extracted, compiled, and run to completion (normal successful termination). (Not the same as passing the tests.)
- The JUnit test classes check all of the public functionality (constructors and methods) of the three classes to be tested.

Check:

- The JUnit test cases cover normal (at least 3 cases), error (at least 1 case), and boundary (at least one case) conditions, as applicable.
- The JUnit test classes use @Before to create common objects for the test cases. (There is nothing for @After to do.)

Plus:

- The JUnit tests correctly identify all errors in the code.

Back