Assignment 1 - To be performed INDIVIDUALLY.

Automated Requirements-Based API Unit Testing using JUnit

1. System Under Test

JFreeChart is an open source Java framework for chart calculation, creation and display. This framework supports many different (graphical) chart types, including pie charts, bar charts, line charts, histograms, and several other chart types. Note that the versions of JFreeChart distributed for this lab do not correspond with actual releases of JFreeChart. The versions have been modified for the purposes of this assignment.

The JFreeChart framework is intended to be integrated into other systems as a quick and simple way to add charting functionality to other Java applications. With this in mind, the API for JFreeChart is required to be relatively simple to understand, as it is intended to be used by many developers as an open source off-the-shelf framework. A snapshot of four different types of charts drawn using JFreeChart is shown in Figure 1.

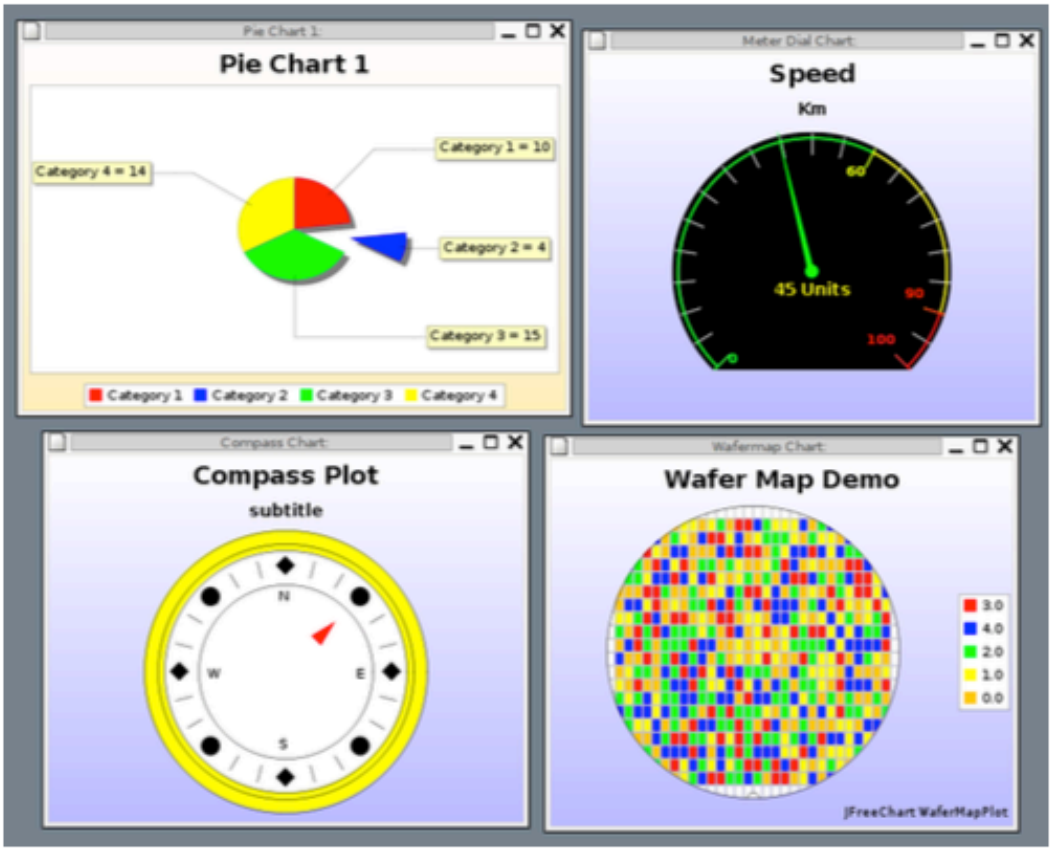

Figure 1 - A snapshot of four different types of charts drawn using JFreeChart.

While the JFreeChart system is not technically a stand-alone application, the developers of JFreeChart have created several demo classes which can be executed to show some of the capabilities of the system. These demo classes have *Demo* appended to the class name. For the purpose of this lab, full knowledge of the usage of the JFreeChart API is not particularly necessary.

The framework is grouped into two main packages, (1) `org.jfree.chart` and (2) `org.jfree.data`. Each of these two packages is also divided into several other smaller packages. For the purpose of testing in this lab, we will be focusing on the `org.jfree.data` package.

2. Provided Artifacts

An Eclipse Java project for JFreeChart. The project includes:

a. a jar file for a modified version of JFreeChart for this assignment (JFreeChart.jar within JFreeChartJar folder).
b. A set of needed libraries for JFreeChart jar file to compile properly.
c. A sample test file RangeTest.java providing a sample test case. This test case passes.
d. A Modifed JavaDocs zip file including the Java Docs for the JFreeChart application.

3. Development of Unit Test Code

3.1 Test Requirements (Classes to be tested)

The class `org.jfree.data.Range` (in the package `org.jfree.data`): has 15 methods, and the class `org.jfree.data.DataUtilities` has 5 methods. You will be required to create unit tests for 5 methods from `org.jfree.data.Range` class **and** to create unit tests for all methods from `org.jfree.data.DataUtilities` class, testing all the selected methods against their specifications as per the provided Java Docs.

3.2 Test-Case Design

- In a written report, you should discuss how you are designing the test cases (recall the "test-case design" lectures from the class). Since you are given the requirements only, you should apply black-box test-case

design techniques such equivalence classes, and boundary value analysis. When applying these techniques, make sure to explicitly follow the steps discussed in the class, e.g., first derive the domain for each input variable, then the equivalence classes, etc. You should ensure that the requirements are adequately tested.

- Create your test-cases on paper (your written report) first.

3.3 Write your Test Code based on your Test-case Design

- The next step is to code your test code in the JUnit framework based on the list of test cases you have designed on paper. Each test method should include one test case only.

  For example: `testPositiveValuesForMethodX()` and `testNegativeValuesForMethodX()`, instead of a single `testMethodX()`. This will help to keep test cases consistent, and make analysis of test case impact simpler later on.

- Execute the test suite you have created on JFreeChart v1.0.zip. Note that the classes have random defects in them intentionally, and thus several of your tests should fail. Therefore, to write your test methods, you need to follow the specifications, not the actual results.

4. Deliverables

1. Written Lab Report: You can make it hand-written, scan it, and submit the scanned copy, as long as your hand writing is readable.

2. JUnit Test Suite: Submit a TWO tests files. One file would include your tests for Range class and another file would include your tests for the DataUtilitles class. Your JUnit test cases need to have appropriate test method naming convention as explained above. Each test files should include the tests related to its corresponding source code file being tested.

5. Assignment Submission Notes

- You should put the **delivered written lab report** and the **TWO test files** in a folder named CS496_FirstAssignment_StudendID and compress them to a .zip file with the same folder name. The compressed file would be the file to be delivered. Failing to abide by the naming convention would result in REDUCED grades.

- In case of cheating, the student will be get –N grades for an assignment that is worth N grades. ALL participants in cheating will get –N.

5. ACKNOWLEDGEMENTS

The original version of this lab package was developed by Prof. Vahid Garousi at the University of Calgary and is part of a large software testing laboratory courseware available under a Creative Commons license for other testing educators.