Assignment 3

COSC 4313 3.0 Software Engineering Testing, Winter 2008, Section M

Due: Monday, April 7th, 2008, 11:59pm.

Format: Individual or in teams of two.

Testing GUI applications

For this assignment, you are required to apply any of the techniques we examined in the course to test a GUI application. The implementation to test will be provided on the course website. Your task will be to create a test suite in JUnit, produce bug reports, and submit a hard copy report describing your testing.

The implementation you will be working on comes with the following specification:

Specification

Final exams in certain courses consist of multiple-choice questions. Students indicate their answers on Scantron bubble sheets. Thus, each student's answers can be represented as a line:

```
0123456782333-32442444444124125534252552-41-3511441412-532-44-3-2-22
```

The line consists of a student number (9 digits), and a vector of answers. The numbers 1 to 9 map to answers A to I respectively. The symbol `-` indicates that the student did not answer the question. The symbol `?` indicates that scantron could not read the answer on the student's bubble sheet.

Correspondingly, the answer key can be represented the same way:

```
answerkey2231x22442443444124335551515322x31x33354x31451113xx532xx554
```

The symbol `x` indicates the question was dropped from the marking scheme. The instructor provides the answer key to CNS, who then runs Scantron on all of the exams and sends back a mark for each student, as well as a file containing the scanned data.

In the case where academic dishonesty is suspected, corroborating evidence can be obtained from the scantron data. If a student copies the answers from someone sitting nearby, the two answer vectors may appear suspiciously similar.

One approach is to calculate the similarity measure (number of same answers divided by the total number of questions). While this measure can be suggestive (suppose two students provide the same answer, and the answer is not only incorrect, but also incorrect in a way that differs from most of the other students who had an incorrect answer), it can never be conclusive. Why? Because the similarity of two answer vectors is also affected by how many answers are correct. For instance, two students who both get a mark of 90% will have a high likelihood of having highly similar answer vectors.

To address this problem, a measure called the kappa statistic has been developed. It tells us what the likelihood is that the similarity between two answer vectors is due to chance. For instance, suppose we suspect student X. We calculate the kappa statistic between X and all other students in the class. Suppose we find for student Y a similarity measure of 0.82, and the kappa statistic tells us that the likelihood that X and Y's similarity measure of 0.82 is due to random chance is 0.001%. This is strong evidence for cheating and can be used to corroborate eye witness accounts. Note that the measure cannot be used to tell whether X cheated from Y or vice versa. The kappa statistic should only ever be used to corroborate eye witness evidence.

There is one other aspect of the kappa statistic. The statistic requires that, for each question, the probability distribution be derived over all of the possible answers. The default is that all of the student answers be used to build the probability distribution. But this can be changed.

Suppose the suspected student had x answers correct. A more refined statistic would be to consider only those other students who had a similar number of correct answers, such as $x \pm 2$. The value 2 is referred to as the conditioning window size. It can be a bit tricky to choose a good value. Choose a window size too small, and there will be too few student answer vectors to provide a good probability distribution. Choose a window size too large, and the level of similarity required to satisfy the statistical threshold gets very high.

The GUI application you are testing in this assignment implements all this functionality. It allows the user to select files that contain the answer key and the student answers, as well as choose the size of the conditioning window. It can then generate a similarity report for any student.

What to do

Download `a3.tar` from the course website and untar it in the `winter2008` directory from Assignment 2. Make sure to include it in your CLASSPATH, or in your Java Build Path if you are using Eclipse. It includes two packages:

- `driverApps`. This package contains the main method (in class `KappaApplicationDriver`) as well as the class that extends `JFrame` (`KappaView`).
- `kappaStatistic`. Classes in this package provide services for the calculation of the kappa statistic.

For this assignment, you will create a package called `window2008.a3.test` that will contain all your test classes. The main method that will be used to run your test cases must be included in a class called `AllTests.java`. You will probably need to import the `KappaView` class in every test class with

```
import winter2008.a3.code.driverApps.KappaView;
```

Using this setup, create a test suite for the provided implementation. Your tests should test both the GUI behaviour as well as the system's functionality. You are free to create as many test classes as you like. You may use any testing strategy you believe is appropriate including the ones discussed in class. If you believe that the implementation fails to meet the specification in some way, create bug reports. The bug reports will be part of your written report, so you can create them in whatever format you prefer.

Note that you are not required to do unit testing of all the classes in this system. Test the system's functionality through the GUI only. A paper that describes the kappa statistic in detail is posted on the course website. This is only for reference, you do not need to understand the math in the paper for this assignment.

The report that the software generates is rather complicated. For the purposes of this assignment, we are only interested in the third column (the similarity measure) and the last column (the likelihood the similarity is due to chance). You do not need to test any of the other numbers.

What to Submit

Before the deadline, submit electronically the test code package you created. To submit, navigate to the directory that contains the `winter2008` package, and give a command like the following:

```
submit 4313 a3 winter2008
```

You also need to submit a written report. This report must include:

- A description of the testing strategies you applied, as well as the test cases you created. The marker will not read your code in order to see what you tested. You have to describe it.
- The bug reports you created.

Submit a hard copy of the written report to the 4313 drop box by the deadline. Attach the last page of this handout as the first page of the hard copy submission. Fill in your name and student number. Submit the same report electronically before the deadline as well (PDF format is preferrable, but other formats will also be accepted). To submit, give:

```
submit 4313 a3 a3.pdf
```

where `a3.pdf` is your report. The marker will mark the hard copy submission. Electronic submissions will be used for remarking purposes.

Grading

This assignment will be marked out of 100. The marks will be divided as follows:

- Bug finding power of your testing: 20%
- Description of GUI testing: 30%
- Description of functionality testing: 30%
- Bug reports: 10%
- Presentation/English: 10%

Presenting your thought processes in English is an important skill for a software professional. If you have trouble writing English, have somebody proof-read and correct your prose. You might find the services offered by The English as a Second Language Open Learning Centre useful: `http://www.yorku.ca/eslolc`

Assignment 3 Grade Sheet

COSC 4313 3.0 Software Engineering Testing, Section M, Winter 2008

Student Name:

Student Number:

Student Name:

Student Number:

Bug finding power of your testing	
Description of GUI testing	
Description of functionality testing	
Bug reports	
Presentation / English	
Total	
Letter grade	