Assignment 2

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

Due: Thursday, March 20, 11am.

Format: Individual or in teams of two.

Testing with JUnit

The purpose of this assignment is to give you experience in creating automated test code with JUnit. 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 written report describing your testing.

Background

A task that one often has to perform in the software industry is to attempt to understand a legacy software system that needs to be updated for a variety of reasons, such as adding new features or porting to a new hardware platform. This task is often hard due to the absence of the original developers, as well as the lack of up-to-date documentation. As a result, one often is faced with hundreds of source files and no place to start.

A common solution to this problem is to employ a clustering algorithm, i.e. an algorithm that will break the system into meaningful subsystems that should be easier to understand (a typical subsystem may contain 15-30 source files). Such algorithms typically utilize dependencies between the source files, such as function calls, variable references etc. in order to decompose the software system.

For example, here is a possible set of dependencies extracted from a software system:

```
depend a.c a.h
depend a.c b.c
depend b.c c.c
depend c.c p.c
depend p.c p.h
depend c.c z.c
depend z.c z.h
```

In this example, the software system contained eight source files called `a.c`, `a.h`, `b.c`, `c.c`, `p.c`, `p.h`, `z.c`, and `z.h`.

However, it is often difficult to extract such dependencies from large software systems. For this reason, it would be helpful if one could randomly create such dependency graphs that resemble dependencies found in real software systems. In this assignment, you will test a system that creates such random dependency graphs.

What to do

Download `GraphGenerator.java` from the course website. Place it in an appropriate directory (`winter2008/a2/code`). You will also need several jar files, also provided in the course website. Make sure to include them in your CLASSPATH, or in your Java Build Path if you are using Eclipse.

For this assignment, you will create a package called `winter2008.a2.test` that will contain all your test classes. This package must contain a class called `AllTests` in a file called `AllTests.java` that will be used to run your test cases. You will probably need to import the `GraphGenerator` class in every test class with

```
import winter2008.a2.code.GraphGenerator;
```

Using this setup, create a test suite for the provided implementation. You are free to create as many test classes as you like.

Hint: Do not use absolute paths for any test files you will create, since these will not work after you submit. In order to create a file called `output.rsf` in the `test` directory, you might need to refer to it as `winter2008/a2/test/output.rsf`.

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.

Following is the specification for the implementation you will be testing:

Specification

The API for the `GraphGenerator` class contains the following two methods:

1. `public GraphGenerator (int numberOfEntities, int maxDegree, boolean isConnected, String outputFileName)`

   The first argument is the number of entities that the generated file should refer to. The second argument is the maximum degree for any entity (the degree of entity a is defined as the number of other entities a depends on).

If the boolean argument is true, then the produced dependency graph must be connected in the non-directed sense. If the boolean argument is false, then the graph may or may not be connected.

Finally, the String argument is the name of the file that will contain the produced output in the format described above (we call this the RSF format).

2. `public void createGraph() throws Exception`

   This method generates the random set of dependencies and outputs it in the designated file. Since it uses randomization it can be called multiple times, each time producing a different output (following the parameters set in the constructor of course).

   The generated graph needs to have the same number of entities as specified in the constructor. Each entity should have a degree that is less or equal to the maximum degree. No duplicate edges should appear in the dependency graph.

   If the number of entities or the maximum degree is less than 1, then a `NegativeInputException` is thrown.

   Bonus spec: One of the properties of the generated set of dependencies that this implementation guarantees is that the degrees of the nodes in the generated graph follow a power law distribution (see `http://en.wikipedia.org/wiki/Power_law`). One would like to test whether this property holds and possibly estimate the exponent of the power law distribution. Warning: Do not attempt this part of the assignment unless you are finished with everything else. It is hard!

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 a2 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(s) and

student number(s). Submit the same report electronically before the deadline as well (PDF format is preferable, but other formats will also be accepted). To submit, give:

```
submit 4313 a2 a2.pdf
```

where `a2.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%
- Coverage of your testing: 20%
- Description of your testing approach: 40%
- Quality of bug reports: 10%
- Presentation/English: 10%

Bonus marks (up to 10%) will be givenif you address the bonus spec.

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 2 Grade Sheet

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

Student Name:

Student Number:

Student Name:

Student Number:

Bug finding power .....

Coverage _____ .....

Description of testing approach .....

Quality of bug reports .....

Presentation / English .....

Bonus .....

Total .....

Letter grade .....