CS4272 Assignment 2

Due on 24 October 2007, 11:59 pm

1 Notes

- This is an individual assignment. Acts of plagiarism are subjected to disciplinary action by the university. Please refer to

  http://www.comp.nus.edu.sg/students/plagiarism/

  for details on plagiarism and its associated penalties.
- Submission Instructions: (Failure to follow these instructions may result in deduction of marks.)
  - Submit one file named **YourMatricNumber.pdf** or **YourMatricNumber.doc**, containing your answers to questions, to the IVLE Workbin Folder *Assignment 2 Submission*. Submit only *one* copy.

2 Questions

1. (6 marks) JUnit is a popular unit testing framework for Java programming language. It contains a main class and takes unit test cases as parameter. By invoking JUnit, all test cases are executed sequentially and the test result is reported. Usually, the developer of an application write such test cases for automatic testing. Test cases are created by extending the base class and providing a series of tests. A test case example is:

```
public class HelloWorld extends TestCase {
    public void setUp() { ... }
    public void testMultiplication {
        // Testing if 3*2 = 6
        assertEquals ("Multiplication", 6, 3*2);
    }
    public void testDivision {...}
    public void tearDown() { ... }
}
```

   where `setUp` and `tearDown` are two methods to initialize and clean up testing environment of current test case. `testMultiplication` and `testDivision` are the two tests contained in the test case.

   Ideally, test cases are independent with each other, such that they can executed selectively. For example, when a Java class in a project is modified, it may not be necessary to execute all test cases for the project. In

other words, we should be able to execute any subset of test cases and get the same result - whether tests are passed or not.

JMeter is a Java based functional and performance evaluation tool, and test cases are provided by the developer. In a particular version, the test cases provided for unit testing are not fully independent. That is, the code executed for one test case is (transitively) control/data dependent on code executed for another test case. As a dynamic slicing tool, JSlice is capable of finding out such control/data dependence during testing. To execute these test cases, we execute JUnit and instruct it to evaluate test cases of JMeter.

Use JSlice tool to find out all dependency cases between two test cases. Write in your report the **detailed approach** to find these dependencies, including but not limited to criteria selected, relevant part of slicing results, and etc.

Note:

(a) All eight test case classes are listed and labeled in `JMeterTestSuite.java` at root folder of JMeter. You are required to analyze at this test case level. The `setUp` and `tearDown` methods are part of the test case. (You should report dependence between test cases. However, for easier understanding, you may set the criterion as a single test in a test case.)

(b) Generally, a slicing criterion is (l, v) of a variable v at line l. In this assignment, we use a higher level criterion definition - a slicing criterion is a method m, such that all statements executed between entering and exiting m are considered as part of criterion. For the usage of JSlice, see section 3.

(c) Given a slicing criterion, the slicing result contains a set of statements. An entry (class name, line number) in the slicing result indicates that the criterion **is dependent on** the entry. That is, you may find the dependence between test cases by setting appropriate criteria and examining the slicing results.

2. (2 marks) A common reason for test case dependence $a \leftarrow b$ (test case b is dependent on test case a) is: a variable v is defined during execution of test a and v is then used by test case b without re-definition. In some cases, for a test case dependence $a \leftarrow b$, if we remove test case a, the behavior of b will be different. This is the real test case dependence. By inspecting the code, you will find that if you re-write the test cases or the application, such dependence can be removed.

   In some other cases, the dependence $a \leftarrow b$ is not due to the application - by removing test case a, the behavior of b is not affected. This is because both test cases use some Java library (e.g. Thread). In the first access by a, the library initializes its variable; while in the second access by b, the library just uses the variable. This is essentially the delayed initialization.

   Thus we are interested in finding out real test case dependence and then remove it to maintain independence between test cases.

   For every dependence cases found in question 1, determine whether it is a real test case dependence. Write the **detailed approach** in your report.

3. (2 marks) For all **real** test case dependence found in question 2, state the reason briefly, and suggest a solution to remove such dependence if possible. (You may need to inspect the source code of JMeter).

3 About JSlice

JSlice (http://jslice.sourceforge.net/) is a dynamic slicing tool for Java. In particular, given a Java program and a slicing criterion, it executes the Java program (for the dynamic behavior) and report a set of statements. These statements essentially affect the execution of the criterion through control dependence and/or data dependence. To use the pre-installed JSlice, you can follow the steps below:

1. You should logon to this server using SSH to access JSlice:

   modelchk.ddns.comp.nus.edu.sg

   The username and password has been sent to you.

2. Download the `jmeter.tar.gz` from IVLE workbin folder *Assignment 2*. Transfer it to your home folder on the server, and extract it in your home folder using the command

   tar zxf jmeter.tar.gz

3. Navigate into the root folder of JMeter, and specify the slicing criterion in a file (e.g. `criteria.txt`) as:
   `number_of_criteria`
   `class_name_1 method_name_1`
   `class_name_2 method_name_2`
   ...
   For example:
   1
   org.apache.jmeter.control.OnceOnlyController$Test testProcessing

4. To run JSlice, type `sh runtest.sh` *criteria slice_result*. The slice result of a set of statements will be saved in the file specified. The format in the result file is:
   `class name 1`
   `line number 1`
   `class name 2`
   `line number 2`
   ...

5. To recompile the root test file `JMeterTestSuite.java`, execute:
   /usr/local/slicing/bin/javac -classpath
   /usr/local/slicing/lib/Klasses.jar:/usr/local/slicing/lib/kjc.jar:./lib/junit.jar:./
   JMeterTestSuite.java