Software Testing MTAT.03.159
Lab 2: Black-box Testing

Lecturer: Dr. Dietmar Pfahl
TA: Briti Deb
TA email: deb@ut.ee
Inst. of Comp. Science, University of Tartu

April 2013

- Submission deadline: Before the start of the next lab (i.e., before start of Lab 3).
- Late Policy: 50% deducted for every 24 late hours.
- Group: Maximum two members in a group. Answers should be your own group work, in your own words.
- Maximum points = Eight.

1 Introduction

In black-box testing, the purpose is to the test the output from the component under test. There are different strategies to use in order to test efficiently. In this exercise you will use equivalence partitioning (EP) and boundary-value analysis (BVA) on a small code example. You will also get insight into the JUnit approach by using JUnit when applying the black-box techniques.

2 Learning Objectives

The exercise aims at giving an understanding of black-box testing. The specific learning goal is to gain a detailed insight into two common black-box testing techniques: the equivalence partitioning and boundary-value analysis.

3 Preparation (on paper)

Assignment 1: Read chapter 4 in [1] and the lecture slides on Black-box test techniques.
Assignment 2: Read the tutorial on using JUnit in Eclipse available at [2] and [3].
Assignment 3: Read the documentation for the program Triangle, available at the course homepage.

4 Exercise (on computer)

Assignment 4: Now test the Triangle program using equivalence partitioning and boundary-value analysis. First, specify test cases by using the EP and BVA techniques at a unit test level (specify test cases for each class method, but exclude the main method). Remember to specify test inputs, execution conditions and expected output, and make sure that the specied test cases cover both Valid equivalence classes and Invalid equivalence classes.

You should implement the test cases (see Appendix 1 for a sample test case) you prepared on paper. Add new test cases if you discover equivalence classes you missed during the preparation. Preferably, use Eclipse and JUnit. Execute the test cases when you have implemented them.

5 Report

The report should be submitted in a PDF file via the course web-page using the 'submit' button – selection: Lab 2

On the first page of your report, write the name of the lab, group member's names and email addresses. The report is expected to be 4-5 A4 pages, and should focus on conclusions from the lab session.

Record your test results carefully in your test report. Remember to specify test case ID, what is tested, description, input, expected output and any other useful information. You may also want to make room for pass and fail notes and perhaps for comments. The report **must** include the following five sections.

1. The test cases. (6 points)
2. Defects detected. An example defect report can be found in [1, p. 363]. (0.5 point)
3. Which method works best and why? (0.5 point)
4. When is each method most applicable? (0.5 point)
5. Can you name any other Black-box testing techniques other than EP and BVA techniques, if EP and BVA techinques are not used? (0.5 point)

References

[1] Burnstein, I., Practical Software Testing - A Process-Oriented Approach, Springer-Verlag, 2003.
[2] http://www.cse.chalmers.se/edu/year/2012/course/TDA566/juniteclipse.html
[3] http://www.cs.washington.edu/education/courses/cse143/11wi/eclipse-tutorial/junit.shtml

Appendix 1

Table 1: Test case

Test ID	Method Under Test	Input	Testing Technique	Expected Results	Actual Results	Comme nts
1	isScalene()	3, 4, 5	EP	scalene	rightangled	Fail

Appendix 2

Table 2: Defect Report

Test ID	Project	Location	Symptom
1	Triangle	IsScalene()	Expected scalene but returned rightAngled