**Course: Software Testing**

**Lab. Report #2 – Automated Requirements-Based API Unit Testing using JUnit**

|  |  |
| --- | --- |
| Group #: |  |
| Student Names: |  |
|  |

**Table of Contents**

[1 Unit testing plan 2](#_Toc159269567)

[1.1 List of the methods under test 2](#_Toc159269568)

[1.2 The chosen black-box test technique(s) and why you have decided to choose them 2](#_Toc159269569)

[1.3 How the team plans to organize their JUnit test suites (based on Appendix C in lab doc) 2](#_Toc159269570)

[2 Designing the unit test-cases using black-box test-case design techniques 2](#_Toc159269571)

[2.1 SUT Class DataUtilities 2](#_Toc159269572)

[2.1.1 calculateColumnTotal(Values2D data, int column) 2](#_Toc159269573)

[2.1.2 calculateRowTotal(Values2D data, int row) 3](#_Toc159269574)

[2.1.3 createNumberArray(double[] data) 3](#_Toc159269575)

[2.1.4 createNumberArray2D(double[][] data) 3](#_Toc159269576)

[2.1.5 getCumulativePercentages(KeyedValues data) 3](#_Toc159269577)

[2.2 SUT Class Range 3](#_Toc159269578)

[2.2.1 Name of the method under test from Class Range 3](#_Toc159269579)

[2.2.2 Name of the method under test from Class Range 3](#_Toc159269580)

[2.2.3 Name of the method under test from Class Range 4](#_Toc159269581)

[2.2.4 Name of the method under test from Class Range 4](#_Toc159269582)

[2.2.5 Name of the method under test from Class Range 4](#_Toc159269583)

[3 Output of test suite execution: Include a screenshot of test suite execution in JUnit showing their Pass/Fail/Error status (showing the names of test methods) 4](#_Toc159269584)

[4 Based on the list of failed test cases (failures) in the previous section above, discuss the possible faults leading to those failures 5](#_Toc159269585)

[5 How the team work/effort was divided and managed 5](#_Toc159269586)

[5.1 How the team work/effort of the lab was managed and divided 5](#_Toc159269587)

[5.2 Writing the lab report 5](#_Toc159269588)

[5.3 Lessons learned from your teamwork in this lab 5](#_Toc159269589)

[6 Difficulties/ challenges encountered, overcoming them, and lessons learned 6](#_Toc159269590)

[6.1 Difficulties/ challenges encountered 6](#_Toc159269591)

[6.2 How did you overcome the above difficulties/ challenges? 6](#_Toc159269592)

[6.3 “Technical” Lessons learned 6](#_Toc159269593)

[7 Comments/feedback on the lab and lab document itself 6](#_Toc159269594)

[7.1 Did you find the lab a useful learning experience? How it helped you learn the new testing topics 6](#_Toc159269595)

[7.2 Was the lab document easy to follow? 6](#_Toc159269596)

[7.3 About time budget? (Was there too much/too little time for this lab?) 6](#_Toc159269597)

[7.4 Please provide your comments on how to improve the lab work and lab document 6](#_Toc159269598)

**URL of your project in GitHub.com**

[www.github.com/…](http://www.github.com/…)

**General advice for writing high-quality lab reports:**

* Ensure following technical writing advice and resources on the internet. Do a Google search for:
  + <https://www.google.com/search?q=technical+writing+for+software+engineers>
  + And <https://www.google.com/search?q=technical+writing+for+students>
* Avoid common mistakes of technical writing ([google.com/search?q=technical+writing+mistakes](https://www.google.com/search?q=technical+writing+mistakes)), such as very long paragraphs, etc.

# Unit testing plan

## List of the methods under test

## The chosen black-box test technique(s) and why you have decided to choose them

## How the team plans to organize their JUnit test suites (based on Appendix C in lab doc)

# Designing the unit test-cases using black-box test-case design techniques

* **Make sure to include, in your report: HOW you used the black-box test-case design techniques (such as 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. just like how we learned and practiced with them in the lectures.
* You should not include any test code in this section, but only the design of the test cases using the above methods, before coding them.

## SUT Class DataUtilities

### calculateColumnTotal(Values2D data, int column)

Details steps of the test-case design method:

**Note:** if you apply boundary-value analysis / testing (BVA) and other test-case design techniques, you should also add their details below

1-Identify the input domain (of method/function parameters):

2-Equivalence classing of method input(s): shall be presented "visually", like done in the lectures

3-Combining equivalence classes of the different inputs, using the multi-dimensional approach, i.e., Strong Equivalence-Class Testing (SECT)

### calculateRowTotal(Values2D data, int row)

1-Identify the input domain (of method/function parameters):

2-Equivalence classing of method input(s): shall be presented "visually", like done in the lectures

3-Combining equivalence classes of the different inputs, using the multi-dimensional approach, i.e., Strong Equivalence-Class Testing (SECT)

### createNumberArray(double[] data)

### createNumberArray2D(double[][] data)

### getCumulativePercentages(KeyedValues data)

## SUT Class Range

### Name of the method under test from Class Range

1-Identify the input domain (of method/function parameters):

2-Equivalence classing of method input(s): shall be presented "visually", like done in the lectures

3-Combining equivalence classes of the different inputs, using the multi-dimensional approach, i.e., Strong Equivalence-Class Testing (SECT)

### Name of the method under test from Class Range

1-Identify the input domain (of method/function parameters):

2-Equivalence classing of method input(s): shall be presented "visually", like done in the lectures

3-Combining equivalence classes of the different inputs, using the multi-dimensional approach, i.e., Strong Equivalence-Class Testing (SECT)

### Name of the method under test from Class Range

1-Identify the input domain (of method/function parameters):

2-Equivalence classing of method input(s): shall be presented "visually", like done in the lectures

3-Combining equivalence classes of the different inputs, using the multi-dimensional approach, i.e., Strong Equivalence-Class Testing (SECT)

### Name of the method under test from Class Range

1-Identify the input domain (of method/function parameters):

2-Equivalence classing of method input(s): shall be presented "visually", like done in the lectures

3-Combining equivalence classes of the different inputs, using the multi-dimensional approach, i.e., Strong Equivalence-Class Testing (SECT)

### Name of the method under test from Class Range

1-Identify the input domain (of method/function parameters):

2-Equivalence classing of method input(s): shall be presented "visually", like done in the lectures

3-Combining equivalence classes of the different inputs, using the multi-dimensional approach, i.e., Strong Equivalence-Class Testing (SECT)

# Output of test suite execution: Include a screenshot of test suite execution in JUnit showing their Pass/Fail/Error status (showing the names of test methods)

Include a screenshot of ALL test suite execution in JUnit showing their Pass/Fail/Error status, such as:

A screenshot of a computer

Description automatically generated

**(Note: This is just an example. We are NOT providing the number of test cases for you.)**

# Based on the list of failed test cases (failures) in the previous section above, discuss the possible faults leading to those failures

Hint: Use the chain of software dependability threats: errors, fault, failure, as learned in the lectures

# How the team work/effort was divided and managed

## How the team work/effort of the lab was managed and divided

* You can say for example discuss which parts of the lab-work (e.g., classes under test, etc.) was done by who…
* And also discuss the meetings that you had to plan and run the lab work
* Etc.

## Writing the lab report

Fill up the following table to specify who wrote what part of the lab document:

|  |  |
| --- | --- |
| **Lab-report section** | **Written by** |
| 1- Introduction | Student A |
| 2-.. |  |
| … |  |

## Lessons learned from your teamwork in this lab

Only include lessons learned from **your teamwork in this section**. **“Technical”** lessons learned **shall be discussed in another section below.**

# Difficulties/ challenges encountered, overcoming them, and lessons learned

## Difficulties/ challenges encountered

Text…

## How did you overcome the above difficulties/ challenges?

Text…

## “Technical” Lessons learned

Only include **“technical”** lessons learned from **in this section**. Lessons learned **your teamwork shall be discussed in another section above.**

# Comments/feedback on the lab and lab document itself

This section has the following sub-sections.

## Did you find the lab a useful learning experience? How it helped you learn the new testing topics

Text…

## Was the lab document easy to follow?

Text…

## About time budget? (Was there too much/too little time for this lab?)

Text…

## Please provide your comments on how to improve the lab work and lab document

Text…