Lab Report for SQAT

|  |  |
| --- | --- |
| **Student ID:** | L201726630108 |
| **Student Name:** | Tamraoui Oussama |

**Before submitting your labwork, please DO read the submission instructions carefully** [**How-to-submit-your-labwork.docx**](https://github.com/mordeky/SoftwareTesting/blob/master/Labwork/How-to-submit-your-labwork.docx) **!**

# Lab01: Testing Principles

## Exercise 01: failure, fault, or error

Having a certain terminology helps testers to explain the problems they have with a program or in their software. Below is a small conversation. Fill each of the caps with: failure, fault, or error.

**Mark**: Hey, Jane, I just observed a (1) \_ \_ \_ \_ \_ \_ in our software: if the user has multiple surnames, our software doesn’t allow them to sign in.

**Jane**: Oh, that’s awful. Let me debug the code so that I can find the (2) \_ \_ \_ \_ \_ \_.

*(a few minutes later)*

**Jane**: Mark, I found it! It was my (3) \_ \_ \_ \_ \_ \_. I programmed that part, but never thought of this case.

**Mark**: No worries, Jane! Thanks for fixing it!

## Exercise 02: testing principle

Kelly, a very experienced software tester, visits Books!, a social network focused on matching people based on books they read. Users do not report bugs so often; Books! developers have strong testing practices in place. However, users do say that the software is not really delivering what it promises. What testing principle applies to this problem?

Your Answer is ?

## Exercise 03: testing principle

Suzanne, a junior software testing, just joined a very large online payment company. As a first task, Suzanne analyzed their past two years of bug reports. Suzanne observes that more than 50% of bugs have been happening in the ‘International payments’ module.

Suzanne then promises her manager that she will design test cases that will completely cover the ‘International payments’ module, and thus, find all the bugs.

Which of the following testing principles might explain why this is not possible?

1. Pesticide paradox.
2. Exhaustive testing.
3. Test early.
4. Defect clustering.

Your Answer is ?

## Exercise 04: only unit testing?

John strongly believes in unit testing. In fact, this is the only type of testing he actually does at any project he’s in. All the testing principles below, but one, might help in convincing John that he should also focus on different types of testing.

Which of the following is the least related related to help John in moving away from his ‘only unit testing’ approach?

1. Pesticide paradox.
2. Tests are context-dependent.
3. Absence-of-errors fallacy.
4. Test early.

Your Answer is ?

# Lab02: JUnit for Unit Test

## Target

* To be familiar with the IDE: [Eclipse](https://www.eclipse.org/) / [IntelliJ IDEA](https://www.jetbrains.com/idea/)
* To understand Java's **annotations**
* To understand the basic concept of **Unit Test**
* To be familiar with Unit Test skills with [**JUnit**](https://junit.org/junit4/) **4/5**:
* Assert Functions, e.g., assertTrue, assertFalse.
* JUnit 4/5 Configuration in your IDE(Integrated Development Environment), e.g., [Eclipse](https://www.eclipse.org/), [IntelliJ IDEA](https://www.jetbrains.com/idea/) (Although Eclipse is very popular, I ***strongly*** suggested you to use IDEA. It’s really ***Excellent*** & ***Fascinating***!).
* To know about **Maven** and Maven-based Project
* To understand how to **add dependencies** by Maven
* To understand **Timeout Testing** and **Exception Test**
* To understand **Parameterized Test**

## Tools

* IDE: [Eclipse](https://www.eclipse.org/) / [IntelliJ IDEA](https://www.jetbrains.com/idea/) / any IDE you’d like to use
* Programming Language: Java, **JDK 16 or higher**

## Tasks

### Task 01: Terminology Illustration

Please READ [junit.pdf](https://github.com/mordeky/SoftwareTesting/blob/master/Lectures/Lec03-JUnit/junit.pdf) illustrate the following Terminology about Unit Test:

* + - 1. What’s test suite?

Your Response:

* + - 1. What’s test case?

Your Response:

* + - 1. What’s unit test?

Your Response:

* + - 1. What’s test fixture?

Your Response:

* + - 1. Annotations. JUnit 4.0 uses annotations rather than special names for setting up, tearing down and testing. Please list the mainly used annotations in JUnit 4.0.

Your Response:

* + - 1. Assert Statement. There are two forms of the assert statement. Please illustrate what they are.

Your Response:

### Task 02: Config JUnit 4/5 in a Maven Project

Please read [*JUnit-Maven.docx*](https://github.com/mordeky/SoftwareTesting/blob/master/Lectures/Lec03-JUnit/JUnit-Maven.docx) carefully. According to the instructions and steps illustrated in this document, do the following subtasks:

Configure your own maven project by adding JUnit 4 and JUnit 5 dependencies.

Open the [MySTTPCode project](https://github.com/mordeky/SoftwareTesting/tree/master/Lectures/code) with IntelliJ IDEA and then copy all of the Java class in the lec03.junit package in the [MySTTPCode project](https://github.com/mordeky/SoftwareTesting/tree/master/Lectures/code) to your own project.

Run all of the test methods and snapshot the test results.

Your Response:

NOTE: You should attach your maven project in your source code package, that is, “code/Lab02”.

A Tip: You can refer to the [MySTTPCode project](https://github.com/mordeky/SoftwareTesting/tree/master/Lectures/code) to build your own maven project.

Now, please illustrate why we manage JUnit 4 and JUnit 5 dependencies by maven, rather than manually?

Your Response:

**Assert** is very important for Making Tests. Please read and run the test code, [CalTest](https://github.com/mordeky/SoftwareTesting/tree/master/Labwork/code/Lab02/asserts/CalTest.java), and illustrate why? That is, if we don’t use Assert functions, what will happen in our testing code?

Your Response:

### Task 03: Parameterized Test.

[This video](https://youtu.be/srJ91NRpT_w) introduced the roman numeral problem. We provide its implementation in ***RomanNumeral.java*** and its test class in ***RomanNumeralTest.java***. The two Java classes can be found [here](https://github.com/mordeky/SoftwareTesting/tree/master/Labwork/code/Lab02/roman/) (i.e., Labwork/code/Lab02/roman /).

The method *singleDigit* in *RomanNumeralTest.java* tries to test if the 7 roman numerals, I, V, X, L, C, D, and M, can be correctly mapped to their corresponding Arabic numbers, 1, 5, 10, 50, 100, 500, and 1000. However, the testing code seems very repetitive. It’s clearly not suitable to perform large-scale test cases in this way.

Fortunately, we can greatly simplify the testing codes by using ***Parameterized Test*** provided by JUnit 5. Please refer to the user guide of JUnit 5 [here](https://junit.org/junit5/docs/current/user-guide/#writing-tests-parameterized-tests), and then

1. write a new method ‘singleDigitParameterizedTest’, which implements the test function of *singleDigit* by using ***Parameterized Test***;
2. paste ***the snapshot*** of the running result of your singleDigitParameterizedTest.

**Note:** ***First,*** there are several annotations, such as @ValueSource, @MethoSource, and @CsvSource, used to define the argument source. You should first choose a suitable one. ***Second,*** You also need to study how the arguments defined the source annotations can be fed to the test method.

Your Response:

### Task 04: Test your own code with JUnit 5.0

Basic Employee Compensation Problem. For each week, hourly employees are paid a standard wage per hour, e.g., $20, for the first 40 hours worked, 1.5 times their wage for each hour after the first 40 hours, and 2 times their wage for each hour worked on Sundays and Holidays. Table 1 gives some test cases of this.

Please do the following tasks:

1. write a Java class, WageCalculator, to solve the wage problem in the following;
2. write a test class WageCalculatorTest to test your code by using the test cases in Table 1; [Note: for simplicity and efficacy, it’s better to use Parameterized Test]
3. paste the snapshot of your test result.

Table 1 Test Cases for Basic Employee Compensation Problem

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | | | **Expected** | **Test result** |
| HourlyWage | StandardHours | HolidayHours |
| $20 | 40 | 0 | $800 | Pass |
| $20 | 45 | 0 | $950 | Pass |
| $20 | 48 | 8 | $1280 | **Fail**  Actual: $1360 |

**Your Response:**

# Lab03: Specification-Based Testing

Please do the labwork according to the following steps:

1. Do all of the 8 exercises [here](https://sttp.site/chapters/testing-techniques/specification-based-testing.html) of this lecture; Note that you DO NOT need to write your answers of these exercises into this labwork report.
2. Check if your answers are reasonable or right by referring to [the answers](https://sttp.site/chapters/appendix/answers.html#specification-based-testing) given by the authors, and think about why.
3. Do the following tasks listed in the following subsections.

## Task 01

In this lecture, we introduced two partition methods: Equivalence-Partition Method and Category-Partition Method. Please compare Exercises 03 and 04, and tell me the relations and differences between the two methods. Please also illustrate when it is suitable to use Equivalence-Partition Method and when it is suitable to use Category-Partition Method.

Your Response:

## Task 02

As we learn from the section of Category-Partition Method, we know that there are two principles guiding us to add constraints: identifying invalid combinations and finding exceptional behaviors. We briefly call these two principles as constraint principles. Do you think which constraint principle is usually more difficult to use? According to the experience you had when you do the exercises, give me an example to illustrate your choice.

Your Response:

## Task 03

Please judge the following statements are true or false and illustrate your answer with an example.

1. The specification does not specify any details about some input parameter, and thus, experience should be used to partition it.
2. In an object-oriented language, besides using the method's input parameters to explore partitions, we should also consider the internal state of the object (i.e., the class's attributes), as it can also affect the behavior of the method.

Your Response:

## Task 04

Please do Exercise 07 carefully and answer the following questions.

1. Why should we treat file names 'no-filename with this name' and 'omitted' as exceptional?
2. Why should we constrain the options in the 'occurrences in a single line' category to happen only if 'occurrences in the file' are either exactly one or more than one? Why does it not make sense to have none occurrences in a file and one pattern in a line? Which constraint principle do we follow here?

Your Response:

## Task 05

For Exercise 08, why do we constrain isFull == true rather than isFull == false? Which constraint principle do we follow here to constrain isFull == true?

Your Response:

# Lab04: Boundary Testing

Please do the labwork according to the following steps:

1. Do all of the 8 exercises [here](https://sttp.site/chapters/testing-techniques/boundary-testing.html) of this lecture; you DO NOT need to write your answers of these exercises into this labwork report.
2. Check if your answers are reasonable by referring to [the answers](https://sttp.site/chapters/appendix/answers.html#specification-based-testing), and think about why.
3. Do the following tasks I listed in the following subsections.

## Task 01

Is it possible to have many on-points for a specific condition? If yes, please give an example.

Your Response:

## Task 02

In Exercise 4, if we replace “boundary analysis of inequalities (e.g.,  a < 10 )” with “boundary analysis of equalities (e.g.,  a == 10 )”, which of the following statements is true? Why?

1. There can only be a single on-point which always makes the condition true.
2. There can be multiple on-points for a given condition which may or may not make the condition true.
3. There can only be a single off-point which may or may not make the condition false.
4. There can be multiple off-points for a given condition which always make the condition false.

Your Response:

## Task 03

For Exercise 1, we have the following tasks:

1. The ability to read programs is very important for unit testing. Please read the program in Exercise 1 without running it and give your description about the function of the “sameEnds” function.
2. Revise the program and make it more efficient according to the following tips:
   1. Is it really necessary to run half iterations?
   2. Can we directly use the result in the “left” variable?
3. Also, write test code with at least 2 test cases in JUnit to evaluate if your revised sameEnds is right. One of your test cases should be able to demonstrate the efficacy of your revised program. You’re suggested to use @ParameterizedTest and @ValueSource to write the test code.

Your Response:

## Task 04

Please read the “The CORRECT way” section of this lecture and watch the [teaching video](https://youtu.be/oxNEUYqEvzM), and consider the connection between the CORRECT and the boundary conditions. Why “Conformance”, “Reference”, “Time” are considered as boundary conditions? Why did the authors use the word “Cardinality” to describe the boundary condition in a loop?

Your Response:

# Lab05: Structural-Based Testing (I)

Note that the tasks prefixed with \* (e.g., \* Task 04) are Optional Tasks. You’re encouraged to do these optional tasks to improve your understanding of our lectures. However, only doing tasks without \* is OK. Dy perfectly doing tasks without \* , you can still get 100 score.

## Task 01: Exercises 7 and 8

Actually, You can do Exercises 7 and 8 with or without control-flow graph (CFG). Please try to give your answer without CFG.

Your Response:

## Task 02: Exercise 9

Note that the first question “What is the branch+condition coverage these test cases give combined?” is confusing. It should be “What is the branch+condition coverage when the test cases given above are combined?”

Your Response:

## Task 03: Exercises 11 and 12

Your Response:

## \*Task 04: Condition Coverage vs. Branch Coverage

Why condition coverage is very important compared to branch coverage? Please give your answer by analyzing why 100% branch coverage caused by the following two test cases is not yet enough for testing count method of CountLetters.

T1: str = "cats|dogs"

T2: str = "cats|dog"

/\*\*

\* Given a sentence, the program counts the number of words that

\* end with either an "s" or an "r".

\* A word ends when a non-letter appears.

\*/

public class CountLetters {

public int count(String str) {

1. int words = 0;

2. char last = ' ';

3. for (int i = 0; i < str.length(); i++) {

4. if (!Character.isLetter(str.charAt(i))

5. && (last == 's' || last == 'r')) {

6. words++;

7. }

8. last = str.charAt(i);

9. }

10. if (last == 'r' || last == 's')

11. words++;

12. return words;

}

Your Response:

## \*Task 05: Condition Coverage vs. Branch Coverage

Does 100% condition coverage always lead to 100% branch coverage? If not, why? Please give an example to illustrate the condition under which 100% condition coverage ≠ 100% branch coverage.

Your Response:

# Lab06: Structural-Based Testing (II)

## Task 01: *the relationship between the decisions, the conditions and the paths*

Path Coverage considers the full combination of the conditions in a decision. Each of these combinations is a path. What do you think of the relationship between the decision coverage, the condition coverage and the path coverage? Please use the control flow graph (CFG) of the following program to illustrate your idea. Note that you should draw CFGs in two granularities: the Condition-wise granularity, and the Branch/Decision-wise granularity, and illustrate which granularity leads to path coverage.

void hello(int a, int b) {

if(a > 10 & b > 20) {

System.out.println("Hello");

} else {

System.out.println("Hi");

}

}

Your Response:

## Task 02: Connection between MC/DC & DC (Don’t Care)

Section “Lazy vs eager operators” introduces lazy operators to compress the truth table, while the method for achieving 100% MC/DC can also be considered as another way to compress the truth table. What’s the connection/difference between these two methods? Please illustrate your answers by doing the following two exercises with both the lazy operators and the MC/DC method.

**Exercise 5.** Consider the decision (A or C) and B with the corresponding decision table (decision table can be found in the [webpage](https://sttp.site/chapters/testing-techniques/structural-testing.html)). What is the set with the minimum number of tests needed for 100% MC/DC (Modified Condition / Decision Coverage)?

**Exercise 15.** Consider the expression ((A and B) or C). Devise a test suite that achieves 100% Modified Condition / Decision Coverage (MC/DC).

Your Response:

## Task 03: Is MC/DC always achievable?

MC/DC is not always achievable in some expressions. See

* (A and B) or (A and not B);
* A and (A or B).

Please illustrate the reason by drawing their truth tables.

Your Response:

# Lab07: Model-Based Testing: Decision Table

All of the following exercises are from the Chapter “*Model-based Testing*” and can be found [here](https://sttp.site/chapters/testing-techniques/model-based-testing.html). The reference answers can be found [here](https://sttp.site/chapters/appendix/answers.html#model-based-testing). Please first report your own answers in the corresponding subsections and then check with the *reference answers*. *Also, analyze the differences between your own answers and the reference answers and point out how you learn from these differences.*

Note that the tasks prefixed with \* (e.g., \* Task 04) are Optional Tasks.

## Task 01: Exercise 08

Please refer to [the reference answer](https://sttp.site/chapters/appendix/answers.html#model-based-testing) given by the authors.

### Your Own Answer

### The differences between your Own Answer and the Reference Answer

### What do you learn from these differences?

### **\***Try to answer the following question

Mathematically speaking, N+1 is the minimum number of tests required for MC/DC coverage and 2∗N the theoretical upper bound. Here, N is the number of conditions in a decision. How many tests do we need in Exercise 08 to achieve 100% MC/DC? Please illustrate your answer by considering the following three requirements of MC/DC:

1. Each condition is at least once true and once false in the test suite;
2. Each unique action should be tested at least once;
3. Each condition should individually determine the action or outcome.

Your Response:

## \*Task 02: Exercise 14

Please refer to [the reference answer](https://sttp.site/chapters/appendix/answers.html#model-based-testing) given by the authors.

### Your Own Answer

### The differences between your Own Answer and the Reference Answer

### What do you learn from these differences?

## Task 03: Exercise 15

Please refer to [the reference answer](https://sttp.site/chapters/appendix/answers.html#model-based-testing) given by the authors.

### Your Own Answer

### The differences between your Own Answer and the Reference Answer

### What do you learn from these differences?

# Lab08: Model-Based Testing: State Machine

Note: all of the following exercises can be found [here](https://sttp.site/chapters/testing-techniques/model-based-testing.html). Their reference answers can be found [here](https://sttp.site/chapters/appendix/answers.html#model-based-testing). Please first report your own answers in the corresponding subsections and then check with the *reference answers*. *Also, analyze the differences between your own answers and the reference answers and point out how you learn from these differences.*

Note that the tasks prefixed with \* (e.g., \* Task 04) are Optional Tasks.

## Task 01: Exercise 09

Please refer to [the reference answer](https://sttp.site/chapters/appendix/answers.html#model-based-testing) given by the authors.

### Your Own Answer

### The differences between your Own Answer and the Reference Answer

### What do you learn from these differences?

## \*Task 02: Exercise 10

Please refer to [the reference answer](https://sttp.site/chapters/appendix/answers.html#model-based-testing) given by the authors.

### Your Own Answer

### The differences between your Own Answer and the Reference Answer

### What do you learn from these differences?

## Task 03: Exercise 11

Please refer to [the reference answer](https://sttp.site/chapters/appendix/answers.html#model-based-testing) given by the authors.

### Your Own Answer

### The differences between your Own Answer and the Reference Answer

### What do you learn from these differences?

## Task 04: Exercise 13

Please refer to [the reference answer](https://sttp.site/chapters/appendix/answers.html#model-based-testing) given by the authors.

### Your Own Answer

### The differences between your Own Answer and the Reference Answer

### What do you learn from these differences?

# Lab09: Design by Contracts

Note: all of the following exercises can be found [here](https://sttp.site/chapters/testing-techniques/design-by-contracts.html). Their reference answers can be found [here](https://sttp.site/chapters/appendix/answers.html#model-based-testing). Please first report your own answers in the corresponding subsections and then check with the *reference answers*. *Also, analyze the differences between your own answers and the reference answers and point out how you learn from these differences.*

Note that the tasks prefixed with \* (e.g., \* Task 04) are Optional Tasks.

Note that you really need to do all of the exercises (Exercises 01~08) to master what we learned in this lecture. However, to reduce your burden, I only select the most important ones.

## Task 01: Exercise 01

Please refer to [the reference answer](https://sttp.site/chapters/appendix/answers.html#design-by-contracts) given by the authors.

### Your Own Answer

### The differences between your Own Answer and the Reference Answer

### What do you learn from these differences?

## Task 02: Exercise 08

Please refer to [the reference answer](https://sttp.site/chapters/appendix/answers.html#design-by-contracts) given by the authors.

### Your Own Answer

### The differences between your Own Answer and the Reference Answer

### What do you learn from these differences?

## Task 03: Liskov Substitution Principle (LSP)

Please check two Java classes, Rectangle and Square, [here](https://github.com/mordeky/SoftwareTesting/tree/master/Labwork/code/Lab09/shape). LSP is violated by the two classes. Please run the test methods test1 and test2 in RectangleTest and report what goes wrong during running and also explain the reason. To avoid errors like this, please reimplement the two classes, namely Rectangle and Square, in the shape\_revised package according to LSP.

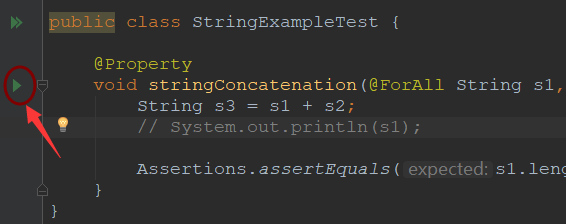
Your Response:

# Lab10: Property-Based Testing (PBT)

|  |
| --- |
| **We have the following two targets in this labowrk:**   1. **Review & Reuse Maven.** We use [jqwik](https://jqwik.net/) to perform Property-Based Testing (PBT). Since jqwik is not supported by standard JDK, we need to download the jqwik package by ourselves. However, a better choice is using Maven. 2. **Config our own project for PBT.** Note that you can download all of example codes from the [code repository](https://www.github.com/sttp-book/code-examples) provided the authors. However, what you really need to master is how to use these codes in your own project. |

Do the following subtasks and record the problems you met and the ways you used to resolve these methods.

1. Please check the PBT project I created for you in [Lab10](https://github.com/mordeky/SoftwareTesting/tree/master/Labwork/code/Lab10); And then, open it in your own IntelliJ IDEA.
2. Create a new empty Maven project in your IntelliJ IDEA.
3. Copy Java codes and maven dependencies from the downloaded PBT project to your newly created Maven project.
4. To ensure if your configuration is correct, please try to run the test method stringConcatenation in the class StringExampleTest by clicking the left green button as follows.



**During the whole procedure, what are the problems you met and the ways you used to solve these problems?**

Your Response:

1. Snapshot the running result.

Your Response:

1. For PBT, generators used to generate random values for the parameters of the test method are very important. The existing generators are often not enough when we want to test one of our own classes; in these cases, we can create a custom generator. Please find an example that uses a custom generator from the codes I provided and paste the code of the test method and the generator.

Your Response:

1. Assert functions are very important. Please compare the assertEquals function used in PassingGradeConceptsTest and the assertThat function used in PassingGradesPBTest. And then illustrate their differences.

Your Response: