CST8116 Lab Exercise 04 (22S)

# Instructions

* The five parts of the Software Development Process as presented by Cay Horstmann [1] will be used as the basis for this lab exercise.

1) Understand the problem

2) Develop and Describe an Algorithm

3) Test Algorithm with Simple Inputs

4) Translate the Algorithm into Java

5) Compile and Test Your Program

* Also refer to your textbook by Joyce Farrell [2] for help with pseucode, flowcharts, UML class diagrams.

# What is the problem to solve for this exercise?

* A tool manufacturing company needs software to verify the lengths of meter sticks as they come off the production line. The software should ask the user to enter the length of the meter stick in meters, then verify that it is within 0.0001 meters of 1 meter. (0.1 meter is decimeter, 0.01 meter is centimeter, 0.001 meter is millimeter). The program should also ask the user what type of output they wish to view, one of verifying the length of the meter stick, seeing the centimeters of the meter stick, seeing the millimetres of the meter stick, or seeing a text report that represents the meter stick. All numbers must be formatted to five decimal places for output. When verifying meter sticks the company uses a tolerance of 0.0001 meters i.e. no more than one-tenth of a millimeter off expected measure of 1.0.

# Part 1 Understand the Problem

* Review the word problem, as well as look at the starter code provided.
* Outline how the main logic would flow from inputs, to processing, to output.
* Outline how the verify method would flow from start to return statement including math
* Use a menu in method main to offer the user a selection of output choices (use constants), and use a selection structure to perform the appropriate processing before displaying the output. You may use any selection structure (select case or if then) in solving this problem.

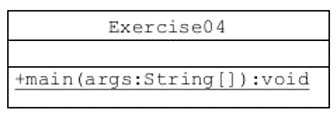
# Part 2a Pseudocode

* Create a detailed UML Class diagram that documents the provided class(s).
* Write pseudocode for the verify method of class MeterStick
* Write pseudocode for the toString method of class MeterStick
* Write pseucode for the main method

?Reminder: Please review the lecture notes for UML Class diagrams as well as your Joyce Farrell textbook [2].

* In our course typically, we will have a class with only method main, as this is a static method it should be underlined.
* Local scope variables inside method main are not to be documented in the UML class diagram in the second row.
* Expanding on that, we never document variables that are local scope, i.e. declared within constructors or methods.
* The only variables that appear in a UML Class diagram, in the second row, are class-level variables declared at a class level.
* The only local-scope variables that appear in a UML class diagram, at all, are the parameters for constructors, and methods, but these are in-lined within each constructor’s or method’s line within the third row.

E.g.



Feel free to reproduce this example as part of your own UML Class diagrams.

## Part 2b Flowchart

* Write a flowchart for the verify method of class MeterStick
* Write a flowchart for the toString method of class MeterStick
* Write a flowchart for the main method
* Place these into your MS Word document as images.

# Part 3 Test Plans for Algorithms

* Create a table in your MS Word document
* You can use the example below as a guide
* Refer to the lecture notes in week 9 regarding boundary testing, there are more than just two tests required.

## Testing verify method of class MeterStick

Rather than using input, we use values of the length field, to then document expected vs actual method returns.

Check boundary conditions just above, below, and at the EPSILON value. (Expecting 5 tests.)

|  |  |  |  |
| --- | --- | --- | --- |
| length | Expected return | Actual return | Description |
| 1.0 | Meter stick is within tolerance of 0.00010 | Meter stick is within tolerance of 0.00010 | Hand trace of method logic indicates that the method return value matches expectations. Length is in between tolerance range. |
| 1.0002 | Meter stick is not within tolerance of 0.00010 | Meter stick is not within tolerance of 0.00010 | Hand trace of method logic indicates that the method return value matches expectations. Length is above upper tolerance range. |
|  |  |  |  |

## Verifying method main

The focus here is to verify that the menu system works, if the user enters a menu option that does not match the expected options are they given an error message from your program logic before the program shuts down?

|  |  |  |  |
| --- | --- | --- | --- |
| Input | Expected output | Actual output | Description |
| 1.0  1 | Meter stick is within tolerance of 0.00010 | Meter stick is within tolerance of 0.00010 | Hand trace of program logic indicates that menu option for displaying verification report worked correctly. Numbers should be formatted to 5 decimal places |
| 1.0  2 | Centimeters: 100.00000 | Centimeters: 100.00000 | Hand trace of program logic indicates that menu option for displaying centimeters conversion worked correctly. |
|  |  |  |  |

# Part 4 Translate the Algorithm into Java

* Create an Exercise04 project in Eclipse
* Using the starter code:
  + Update all of the comment sections as per the requirements of our course, see week 1 lecture notes
  + Follow your algorithm design and code the toString method, verify method, as well as method main.

# Part 5 Compile and Test Your Program

* Compile and run your program.
* Be prepared to demonstrate your program in the lab period, in the week before the formal lab submission is due.
* Re-use your algorithm test plans and test your program code
  + Also ensure that printf is correctly formatting to five decimal places for all output.
* Test and document invalid and valid input like:
  + Lengths within and outside tolerance
  + Valid and invalid menu selection
  + Text instead of numbers for input

**Important:** Only check for invalid inputs like text in place of numbers when testing and documenting the main method. The verify method would never be reached in the case of invalid user input so there would be no program crashes to document in the verify method testing table.

# Lab Demonstration Notes

* Your lab professor will ask you to demonstrate your program in lab, typically the week before the formal submission
* Your lab professor may also ask you a brief question on your code.

# Microsoft Word Document Format

See the template example (from lab exercise 1) and use the suggested headings below:

* Understand the Problem
* Pseudocode(s) and Detailed UML Class Diagram
* Flowchart(s)
* Test Plan Algorithm
* Test Plan Program

# Submission Requirements

* You will need to submit your MS Word document and .java code files by the due date as specified in Brightspace.
* Follow your lab professor’s submission guidelines.

# Grading (8 Points)

|  |  |  |  |
| --- | --- | --- | --- |
| Criteria | Missing / Incorrect (0) | Below Expectations (0.5) | Meets Expectations (1) |
| Understand the Problem | Missing or incorrect. | Partly correct. | Briefly outlines the necessary steps, in order, as an overview. Has examples of boolean expressions needed to solve the problem. |
| Pseudocode(s) | Missing or incorrect. | Partly correct. | Correct format, steps are in correct sequence and lead to correct outputs. Demonstrates correct selection structure use in needed places.  Detailed UML class diagrams are correct. |
| Flowchart(s) | Missing or incorrect. | Partly correct. | Correct format, correct shapes used, steps are in correct sequence, matching pseudocode and lead to correct outputs.  Demonstrates correct selection structure use in needed places. |
| Test Plan for Pseudocode and Flowchart | Missing or incorrect. | Partly correct. | Has correct format as shown in the lab handout, has test values and expected and actual outputs. Has coverage for boundary checks. |
| Demo in lab period | Missing or student could not answer any questions correctly. | Student program may not compile or run correctly, student partly answers question(s) related to their program code, or answer(s) are partly correct. | Student program compiles, and runs correctly, student correctly answers question(s) related to their program code. |
| Source Code: \*.java file(s) Comments and Conventions | Missing or poorly done. | Missing a comment-header from one or more of class declaration and / or method main declaration. Code loosely follows Java coding conventions for identifiers, indentation. | File comment header with student full name is present. Class and method declarations have comment headers. Code closely follows Java coding conventions for identifiers, indentation. This includes Java conventions for indentation of nested selection structure(s). |
| Source Code:  \*.java file(s) program structure and logic. | Missing or poorly done or program does not follow from the pseudocode, and flowchart(s). | Program may have small syntax mistakes and will not compile, and / or produces incorrect output when run. Program loosely follows the student’s pseudocode and flowchart(s). | Program has correct syntax and program logic that produces correct output. Program closely follows the student’s pseudocode and flowchart(s). Program logic for selection structure(s) is correct. |
| Test Plan for Program | Missing or poorly done or is only an unchanged copy of the provided algorithm test plan. | May not have correct format, does not verify that the program outputs match expectations. | Has correct format as shown in the lab handout, verifies that the program outputs match, and documents variations in output including samples of invalid inputs including invalid input that will crash the program. |

# Appendix: Sample Program Runs

Note that the user input in the examples below was formatted with bold text and highlighted using MS Word formatting tools. Within the Eclipse IDE the default font color for user input is a light green.

Each sample is a separate run of the program from start to finish.

## Meter stick out of tolerance

Meter stick checker program.

Enter measured length in meters: **1.002**

Please select from one of the following:

1 to validate meter stick

2 to show meter stick length in centimeters

3 to show meter stick length in millimeters

4 to show meter stick report

**1**

Meter stick is not within tolerance of 0.00010

Program by Stanley Pieda

## Meter stick within tolerance

Meter stick checker program.

Enter measured length in meters: **1.0000999**

Please select from one of the following:

1 to validate meter stick

2 to show meter stick length in centimeters

3 to show meter stick length in millimeters

4 to show meter stick report

**1**

Meter stick is within tolerance of 0.00010

Program by Stanley Pieda

## Meter stick out of tolerance, centimeters conversion

Meter stick checker program.

Enter measured length in meters: **1.02**

Please select from one of the following:

1 to validate meter stick

2 to show meter stick length in centimeters

3 to show meter stick length in millimeters

4 to show meter stick report

**2**

Centimeters: 102.00000

Program by Stanley Pieda

## Meter stick out of tolerance, millimeters conversion

Meter stick checker program.

Enter measured length in meters: **1.02**

Please select from one of the following:

1 to validate meter stick

2 to show meter stick length in centimeters

3 to show meter stick length in millimeters

4 to show meter stick report

**3**

Millimeters: 1020.00000

Program by Stanley Pieda

## Meter stick out of tolerance, text representation

Meter stick checker program.

Enter measured length in meters: **1.02**

Please select from one of the following:

1 to validate meter stick

2 to show meter stick length in centimeters

3 to show meter stick length in millimeters

4 to show meter stick report

**4**

MeterStick length 1.02000

Program by Stanley Pieda

## Meter stick out of tolerance, invalid menu option entered

Meter stick checker program.

Enter measured length in meters: **1.02**

Please select from one of the following:

1 to validate meter stick

2 to show meter stick length in centimeters

3 to show meter stick length in millimeters

4 to show meter stick report

**42**

Invalid menu option selected

Program by Stanley Pieda

# References

[1] Cay Horstmann. (2019). Big Java Early Objects. 7th Ed. Wiley.

[2] Joyce Farrell. (2018). Programming Logic & Design Comprehensive. 9th Ed. Cengage Learning.

# Recommended Reading

SplashLearn. (n.d.). Metric System – Definition with examples. Last Accessed March 1, 2022 retrieved from

https://www.splashlearn.com/math-vocabulary/geometry/metric-system