CST8116 Lab Exercise 01 (22W)

# Instructions

The 5 parts of the Software Development Process as by Cay Horstmann [1] form 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

1. What do we mean by understand the problem?

Solving the problem on your own in a general sense; in the context of how would a computer program work.

* + What information do you (or the computer program) need to solve the problem (inputs)?
  + How would you perform a calculation (processing) (what calculations will the computer need to perform)?
  + What is the desired result (outputs) that the program has to provide?

1. What is an Algorithm?

A step-by-step solution to a problem. The algorithm will be planned and documented in a few ways in our course:

Pseudocode – English-like yet close to program code syntax, each separate line is a command for the computer

Flowchart(s) – documents the same steps as pseudocode, but flowcharts are visual with shapes and arrows

Later in the course (week 3) we will also introduce UML Class diagrams as well, to document the structure of classes used to define objects we use to solve problems.

1. Test Plan for Algorithm

We use a test-plan table (starter examples in lecture notes and below) to check our step-by-step logic to verify that inputs are processed to produce correct outputs, if they do not we revisit the algorithm logic to make changes (fixing bugs in the program logic)

1. Translate the Algorithm into Java

Using your pseudocode and flowchart (and later UML Class diagrams) create a Java program, you will need to follow Java syntax requirements as well as select specific data types, e.g. instead of num use either int or double and so on.

1. Test Plan for Java Program

The algorithm test plan can be re-used, but not copied exactly as the tests need be used on the completed program to ensure no mistakes were made in step 4 when the program was written, based on the algorithm.

# What is the problem to solve for this exercise?

* “A person would like a computer program that outputs a tree using ASCII art, you can use the example provided here or modify it however it must be your own work.” Include your name on screen as part of the output, replacing “your name” with your actual name as it appears in ACSIS. To be clear, your program output must depict a tree.

~~~~~~~

~~~~~~~~~~~

~~~~~~~~~~~~~~~~

~~~~ ~~~~~

|| || ~~~

|| || ~~~~~

|| | ||

| |====||

| |

|O|

| |

| |

|| ||

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

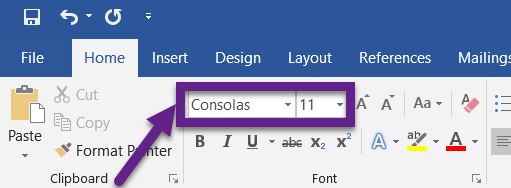
# Program by Your Name

# Part 1 Understand the Problem

* How will the computer output the needed characters?
* Tip: What are the outputs, line by line?
* Tip2: Can you draw the tree using a program like Notepad, before even working on the next parts?

# Part 2a Pseudocode

* Write pseudocode to plan and document your algorithm to solve the problem. Place this into your MS Word document (see below for notes on the format of the MS Word document).
* Note: When documenting your output statements use a Font-Face that is a mono-type-font, this means every character has the same width making it easier to line the letters up. Recommended font-face: Consolas. The screen shot below was taken from Microsoft Word.



* + In MS Word: Select the text to change the font for first. Then use the Home Tab in the ribbon, the font selection drop-down-list, and select Consolas.

# Part 2b Flowchart

* Use a flowchart to plan and document your algorithm. Place this into your MS Word document as an image.
* You can draw a flowchart by hand (use a ruler so lines are straight, and print legibly). Use a scanner or mobile-phone to capture the flowchart as an image then email it to yourself if needed and then copy and paste the image into your MS Word document so it is visible.
* Start learning how to use the recommended software for flowcharts for our course which is Diagrams.net (See week 1 lecture materials for handout on using Diagrams.net). If you are comfortable with Diagrams.net you can use it now rather than drawing by hand.

# Part 3 Test Plan

* Create a table in your MS Word document with headings: inputs, expected output, actual output, description.
* inputs: for this lab exercise there are no inputs to the program, leave this column blank.
* expected output: based on your initial problem solution what is the expected output
* actual output: start at “start” in your algorithm, and follow the steps one at a time to arrive at the output, ending at “stop”, verify that the actual output matches what you would expect (check pseudocode and flowchart)

Here is an example of the table format, before the algorithm is tested.

|  |  |  |  |
| --- | --- | --- | --- |
| Input | Expected Output | Actual Output | Description |
|  | ~~~~~~~  ~~~~~~~~~~~  ~~~~~~~~~~~~~~~~  ~~~~ ~~~~~  || || ~~~  || || ~~~~~  || | ||  | |====||  | |  |O|  | |  | |  || ||  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Program by Your Name |  |  |

# Part 4 Translate the Algorithm into Java

* Following your pseudocode and flowchart create a simple Java program consisting of one class, e.g. Exercise01.java and the main method.
* See the lecture notes from week 1 and Hybrid 1 for help as well as the required readings from your textbooks.

# 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.
* Copy the Algorithm Test Plan table, and paste a second copy into your MS Word document, add notes to verify that the Java program is working as intended.
* You must have two separate test plan tables in your submission, one for the pseudocode/flowchart and one for the Java program. If you only provide one table you will not earn full marks. If both tables are exactly the same you will not earn full marks either.

# 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 and use the suggested headings below:
* Understand the Problem
* Pseudocode
* Flowchart
* Test Plan for Pseudocode and Flowchart
* Program Test Plan

# Submission Requirements

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

# Appendix: Sample Program Output by Professor Stanley Pieda

~~~~~~~

~~~~~~~~~~~

~~~~~~~~~~~~~~~~

~~~~ ~~~~~

|| || ~~~

|| || ~~~~~

|| | ||

| |====||

| |

|O|

| |

| |

|| ||

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Program by Stanley Pieda

# Grading (8 Points)

|  |  |  |  |
| --- | --- | --- | --- |
| Criteria | Missing / Incorrect (0) | Below Expectations (0.1) | Meets Expectations (1) |
| Understand the Problem | Missing or incorrect. | Partly correct. | Briefly outlines the necessary steps, in order, as an overview. Has example of math needed to process input into output, if applicable to the problem statement. |
| Pseudocode(s) | Missing or incorrect. | Partly correct. | Correct format, steps are in correct sequence and lead to correct outputs. |
| Flowchart(s) | Missing or incorrect. | Partly correct. | Correct format, correct shapes used, steps are in correct sequence, matching pseudocode and lead to correct outputs. |
| 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. |
| 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. |
| 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). |
| 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. |

# References

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

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

# Optional Further Reading: ASCII Art

* These are some websites for information and examples of ASCII Art.
* <https://en.wikipedia.org/wiki/ASCII_art>
* <https://www.asciiart.eu/faq>